First Response
An incident handling team learns a few lessons the hard way

David Cragg

Submitted for
GCIH version 2.1
Exploit in Action

Table of Contents

Introduction

This paper documents an actual incident and response in which the author participated. A database server containing a configuration error was exploited through a buffer overflow in SSH and was controlled for approximately 48 hours before being discovered by the Information Security group for our organization.

The aftereffects of this incident are still with the organization nearly a year later. Although the exploit itself is nearly obsolete, the incident illustrates several recurring issues in incident response and security in general.

I The exploit

A Name

This exploit is commonly referred to as SSH-1 CRC-32 compensation attack. It is documented in:

CVE-2001-0144, Core SDI SSH1 CRC-32 compensation attack

CERT Incident Note IN-2001-12, Exploitation of vulnerability in SSH1 CRC-32 compensation attack detector.

To avoid confusion, the reader should note that the compensation attack, as it is generally known, derives its name from its target, the compensation attack detector. The compensation attack detector was put in place to protect against a previous theoretical attack also called a compensation attack, for different reasons. The older attack was also referred to in the literature alternatively as an insertion attack.

This paper will use the name "compensation attack" for the subject exploit, and refer to the older, theoretical attack as the "insertion attack." Although this nomenclature is not ideal, using nonstandard names would have the potential to create more confusion.

B Operating Systems Affected

This exploit affects all unix versions running on Intel x86 hardware. The exploit contains a buffer overflow attack, so is architecture specific. This portion of the code could be modified, but I have been unable to find examples of this having been done.

C Service Exploited

This exploit affects most SSH servers that have SSH v1 protocol enabled.

Vulnerable SSH versions:
SSH Communications SSH1 - version 1.2.25 and later
FSecure SSH1 - version 1.3.5 and later
OpenSSH - all versions previous to 2.3.0

Not vulnerable:
Any SSH running only v2 protocol
OpenSSH version 2.3.0 and later contain a fixed version of the SSH v1 protocol.

D Brief Description of Attack

The SSH1 CRC32 compensation attack exploits a buffer overflow vulnerability located in the CRC32 compensation attack detector. This allows the attacker to execute arbitrary commands on the victim machine with root privilege. The exploit uses its elevated privilege to open a root shell on the victim machine, listening for commands on a port of the attacker's choosing.

A typical attack begins with a scan for vulnerable servers on a network. When a machine is found, the exploit is launched. Several attempts may be required to achieve a successful compromise.

The exploit is usually combined with tools that install backdoors or other software on the compromised machine. The exploit software does not include any such tools.

The CRC32 compensation attack detector was implemented to protect against a weakness in the CRC32 checksum code, which would allow an attacker to alter packets, then update the CRC32 checksum. This attack was effective even if the packets were encrypted, although not all ciphers are vulnerable. [Barrett p102]

E Variants

There are no reported variants at this time. This is a fairly old exploit, nearly a full year old at the time of this writing, and only affects the SSH v1 protocol. Use of the SSH v1 protocol has been mostly discontinued in favor of the SSH v2 protocol. It is unlikely that there will be any further development of this exploit.

F References

The CERT advisory for this vulnerability is at www.cert.org/incident_notes/IN-2001-12.html

It includes system log signatures for the exploit.

A more detailed advisory releases by SuSE is at

http://www.linuxsecurity.com/advisories/suse_advisory-1154.html

Source code for the exploit is available at

www.packetstormsecurity.org/0103-exploits/openssh-2.2.0-exp.tgz

A technical explanation of the vulnerability and the exploit are at

http://razor.bindview.com/publish/advisories/adv_ssh1crc.html

They provide all of the gory details of the implementation error that led to the vulnerability in the SSH server code.

An excellent analysis of the exploit in action is at

http://www.linuxsecurity.com/articles/intrusion_detection_article-4002.html

The author provides extensive detail on the behavior of the exploit, and a long list of references.

II The Attack

A Network description

The network for the facility is a switched model, sub-netted by laboratory. The T3 link to the internet enters the campus, and passes through a border router. Off of the border router, are links to the DMZ and the firewall.

The DMZ contains the primary web server for the institution. Additional servers are planned, including an anonymous ftp server.

The firewall is a Lucent firewall brick. The ruleset permits all connections to port 22, (SSH) both incoming and outgoing. The use of SSH is promoted as an alternative to unrestricted telnet and ftp access on all machines.

From the firewall, the link goes to the core router. The core router ties together switches for each subnet on the campus.

The core router mirrors the traffic from the incoming link, to a port used by the IDS sensor. This is the primary sensor for the IDS. The IDS is the ISS SafeSuite RealSecure Network Sensor. Host sensors are used on the main servers. The victim of the attack, referred to here simply as victim, had no host sensor installed.

The IDS was periodically set to monitor SSH servers operating on the network, to detect outdated versions. At the time of the attack, there was no signature available to identify an actual compensation attack.

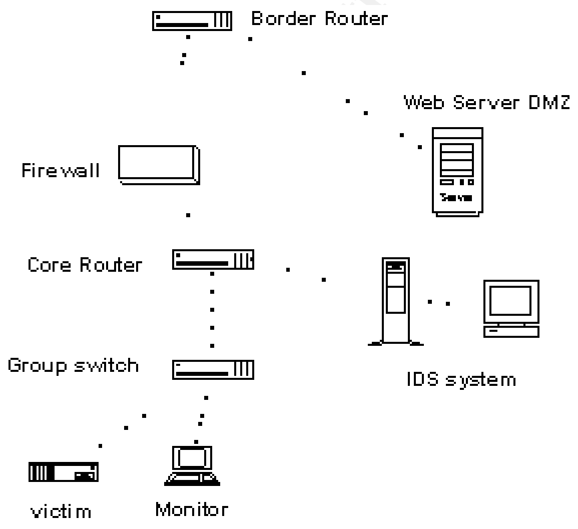


Figure 1 - Network diagram

B Protocol Description

The Secure Shell (SSH) protocol is a client-server protocol to provide encrypted network connections over TCP/IP networks. Originally, it was designed to replace rsh, rlogin, and rcp, collectively known as the r-commands. Functionality has been added to some SSH products to replace telnet and ftp also.

SSH uses public key encryption algorithms for authentication and key exchange. Symmetric algorithms are used for bulk encryption. No single algorithm is specified in any case. Choice of algorithms is negotiated between an SSH client and server.

Public key algorithms use pairs of encryption keys, public and private, to exchange encrypted messages. Any message encrypted with the public key can only be read using the private key, and any message encrypted with the private key can only be read with the public key. This allows two parties to establish an encrypted channel without already having a shared secret.

Symmetric encryption algorithms are traditional ciphers that require a single shared key for encryption and decryption. They are typically faster and stronger than public key algorithms. They are used to encrypt SSH sessions after initial key exchange.

SSH provides authentication through either passwords or through public key authentication. During password authentication, the users password is sent encrypted over the link, affording some protection. During public key authentication, the client and server use a key pair that has been previously exchanged through some other channel to perform a challenge-response type of authentication. The server sends a message that the client must correctly decode and respond to. This eliminates the need to send the user's password over the channel at all, but requires key exchange ahead of time.

Two versions of the SSH protocol exist. Weaknesses in version 1 required changes to the protocol that were not backward compatible. This led to the version 2 protocol.

In particular, the two versions of the protocol use different authentication methods.
SSH v1 uses the CRC32 checksum. CRC32 uses an algorithm optimized to detect bursts of noise in an unreliable communication channel. It is mathematically proven to reliably detect any change that spans no more than 32 bits. SSH v2 uses a Message Authentication Code (MAC) algorithm for integrity checking. MAC algorithms use cryptographically strong hashes instead of checksums.

The CRC32 checksum algorithm is vulnerable to insertion of data into the channel, because it is computationally practical to calculate the change in the CRC32 checksum for a given change in the data.

The compensation attack detector is a module responsible for detecting such an attack. It contains an integer overflow, which is the target of the exploit.

C Exploit description

The exploit is, at heart, a buffer overflow. The CRC32 compensation attack detector must dynamically allocate a buffer to hold each packet before it is analyzed. For very large packets, the integer containing the buffer length defaults to a value of 65536. This value is later copied from a 32 bit integer to a 16 bit integer. The maximum value that a 16 bit integer can contain is 65535, so the value overflows the integer, resulting in a buffer length of 0. The packet would then overflow this buffer, overwriting local variables for the attack detector.

The exploit is in three parts, which work together to exploit the attack detector's vulnerable input buffer. The source code that I was able to obtain contains them in three separate files.

The first part is a patch for the SSH client source code, which is applied to packet.c. The patch is designed specifically for OpenSSH version 2.2.0. This code constructs the malicious packets containing the buffer overflow payload. This code reads several parameters for the packet from a file, including packet length and return address. It then constructs the packet, inserting the buffer overflow payload. The machine code for this payload is included in the patch. Comments in the source code indicate that the payload was re-used from some other source. The payload binds a socket to port 36864, sets itself to listen on that socket, then exec's a root shell.

The second part is source code for a separate executable, written in c. This program takes the packet parameters and the arguments for the SSH client as its input. It performs some type casting and pointer arithmetic to convert offsets to absolute pointers, then writes the results to a file for the modified SSH client. Finally, it calls the SSH client with the necessary parameters, including hostname and port.

The third part of the exploit code invokes the executable repeatedly, incrementing the offset for the return pointer each time. It allows the user to find the correct offset value by brute force. This can vary with different options used by the SSH server. [Dittrich] The code that I worked with used a perl script for this part. It pauses between attempts, to give the user time to react.

To use the exploit, the user must have telnet or netcat ready, and a host selected. If the host is running SSH on a nonstandard port, the user must amend the perl script.

To start the exploit, run the perl script. It will prompt the user for the host address, then make the first attempt. After each attempt, the script will pause, and the user can check for a shell on port 36864.

D Attack Description

The attack against victim most likely began by scanning the muppetlabs.org network for vulnerable web servers. Several scanners exist to do this quickly. Our organization uses ScanSSH for auditing the servers present on the network.

Once victim was located, the exploit was launched against it. Repeated attempts were made. Once the exploit was successful, the attackers connected to the shell on victim. The attackers copied the installer for the t0rn rootkit onto victim and ran it.

After the rootkit installation, attackers launched the exploit again. After this second assault, several back doors were installed. Now logins from all over the world began appearing in the logs. An IRC server was installed, and the party began.

The attackers installed a log scrubber as part of t0rn, but configured it incorrectly, or not at all. The system logs remained intact for the whole period that victim was compromised.

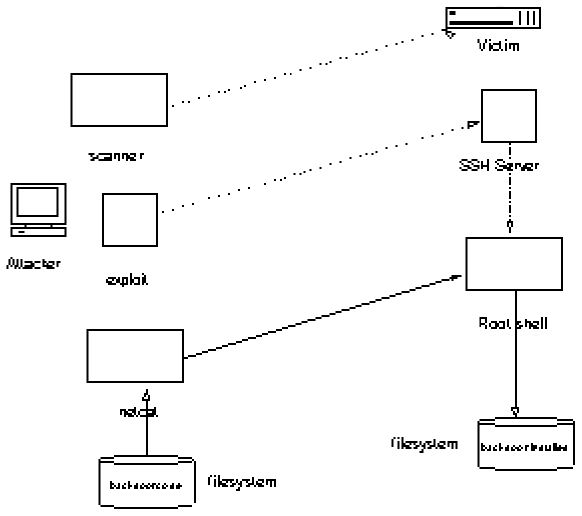


Figure 2 - attack diagram

E Exploit Testing

The exploit was tested on an isolated network between two machines running Red Hat 7.3. OpenSSH 2.2.0 was installed on both machines. The code was examined, and found to have no obvious surprises. An error in the code was fixed, and the exploit compiled and run, to capture signatures.

The exploit failed to compromise the server on several attempts. More detailed examination of the code failed to find significant errors. One of the authors of the exploit code overran a buffer, though.

It is possible that other errors remain. Comparison of the code to another implementation would help, but I have been unable to locate other implementations.

It is also possible that there is a problem with this code on Red Hat systems. Comments in the source code indicate it was tested on Mandrake Linux.

F Attack Signature

CERT reports that the attack leaves the following signature in system logs.

```
 hostname sshd[xxx]: Disconnecting: Corrupted check bytes on input.
 hostname sshd[xxx]: Disconnecting: crc32 compensation attack: network attack detected
 hostname sshd[xxx]: Disconnecting: crc32 compensation attack: network attack detected
```

This is nearly identical to the log artifacts retrieved from the server attacked at our site. Detailed excerpts from our site are listed in the identification section.

David Dittrich has packet captures from a test environment available at

http://staff.washington.edu/dittrich/misc/sshdx.dump

Snort signatures are available for this attack at

http://www.snort.org

There are four signatures in all, SID 1324-1327

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 22 (msg:"EXPLOIT
ssh CRC32 overflow /bin/sh"; flow:to_server,established;
content:"/bin/sh"; reference:bugtraq,2347;
reference:cve,CVE-2001-0144;
classtype:shellcode-detect; sid:1324; rev:3;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 22 (msg:"EXPLOIT
ssh CRC32 overflow filler"; flow:to_server,established;
content:"|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00|"; reference:bugtraq,2347;
reference:cve,CVE-2001-0144; classtype:shellcode-detect;
sid:1325; rev:3;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 22 (msg:"EXPLOIT
ssh CRC32 overflow NOOP"; flow:to_server,established;
content:"|90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90|"; reference:bugtraq,2347;
reference:cve,CVE-2001-0144; classtype:shellcode-detect;
sid:1326; rev:3;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 22 (msg:"EXPLOIT
ssh CRC32 overflow"; flow:to_server,established;
content:"|00 01 57 00 00 00 18|"; offset:0; depth:7;
content:"|FF FF FF FF 00 00|"; offset:8; depth:14;
reference:bugtraq,2347; reference:cve,CVE-2001-0144;
classtype:shellcode-detect; sid:1327; rev:3;)
```

G Protecting Against this Attack

There are two effective ways to protect against the SSH CRC32 attack. Either disable version 1 fallback for all SSH servers, or upgrade to a non-vulnerable SSH.

To remove version 1 fallback for OpenSSH, edit the /etc/sshd_config file. Find the "Protocol" line

```
Protocol 2,1
```

Modify it to read

```
Protocol 2
```

Commenting the line is insufficient, as the server will revert to default behavior, which is to enable version 1 fallback.

For FSecure SSH, remove the /usr/sbin/ssh1 executable from the system.

There are a number of SSH clients that do not support the version 2 protocol. If it is necessary to maintain version 1 SSH compatibility, then upgrade to the most recent version of OpenSSH server. Descriptions exist for patching older SSH servers, but several unrelated attacks on SSH have been discovered in the last year. Patching an older server will require patches for these other vulnerabilities as well.

The version 2 protocol is generally more secure than the version 1. Version 2 contains a number of improvements, and the use of version 1 is now recommended against. Any legacy systems using version 1 should be migrated to version 2 at the earliest opportunity.

Another possibility for legacy systems is to consider alternate solutions for secure communications. There are now a number of VLAN products available that use IPsec to encrypt network connections for secure communications. The encrypted channel is transparent to the user, and any protocol can be tunneled through it.

Part III - The incident Handling process

The handling of this incident illustrates some of the difficulties of operating in an environment where those in charge of security have, in effect, limited control over the assets they are supposed to be protecting. Although the incident was not resolved in a satisfactory manner, it could have been worse. To the extent that the intrusion was not more widespread, we were lucky.

Lack of an adequate security infrastructure also contributed greatly. More could have been done, even in the difficult environment. This incident has served for us as a good example of things to avoid in the future.

A Preparation

The site of the incident is a facility containing several research labs. The company provides network support to the institution under contract. The company was responsible for maintaining the network infrastructure for the research labs, but had no authority over the users in the labs or their machines. Owners of machines had previously been responsible for security and disaster recovery on their own.

The company had been implementing an IT security infrastructure gradually over several months. Policies had been drafted, but had not been approved by the institution.

The newly formed security group primarily maintained the security infrastructure for the network. The services provided to users and administrators outside the company were limited to mostly security advice and assistance. Liability issues prevent company personnel from touching a non-company computer without permission from the owner and authorization from management.

At the time of the incident, there was no established procedure for incident handling. The company had no experienced people available for incident response, and there were no policies in place yet. As a result, the incident was handled in a somewhat ad hoc manner. Lessons learned from this incident have since been incorporated into incident handling procedures.

The security infrastructure in place consisted of two main components, the firewall and the IDS.

The firewall was a commercial firewall server, with a packet filtering router in front of it. The rule set for the firewall was designed to allow connections by default. There were no restrictions on connections to port 22, used by SSH.

The Intrusion Detection System was a commercial product with network sensors communicating to a main console over an isolated network. The network sensor that detected the attack was located behind the firewall and the main switch. It monitored all inbound and outbound traffic, but no internal traffic. There was no network sensor to monitor internal traffic on the network segment that was attacked.

The network architecture used switches to connect most of the nodes to the backbone. Although switches were used purely for performance reasons, they provided the security benefit of minimizing the usefulness of any host based sniffer, should one be installed on a compromised machine.

The security group had three people, a LAN engineer, a security analyst, and a trainee. The LAN engineer was designated as the lead person of the security group, and was responsible for the router and firewall configuration. His time was divided between the security group and the network group.

The security analyst was responsible for operation of the IDS. She also developed policy documents for IT security. She had approximately 2 years experience, and training in Intrusion detection and incident handling.

The trainee was recently transferred from another group to the security group. His previous position was as a Unix system administrator. The trainee was the designated handler for this incident, under the guidance of the more experienced analyst.

B Identification

At approximately 0800 Monday, January 14, 2001, a web server maintained by one of the research labs was broken into. The compromise of the target machine, referred to as victim, was detected approximately 48 hours after the attack. The daily review of IDS logs revealed an IRC server operating on victim. Victim had never hosted an IRC server before. At this point, the event was recognized as an incident.

The machine was an Intel Pentium server running SuSE Linux 7.2. The kernel was modified for large file support. It hosted a web server and a relational database.

Victim was considered a machine of some concern before the incident. It hosted a web server used to deliver scientific data, so was considered an attractive target to intruders. The machine was known to have un-patched vulnerabilities. The most dangerous was believed to be the vulnerability to the SSH compensation attack. The owner of victim had requested and received instructions explaining how to obtain and install the patch, but was known to still be experiencing difficulties. He had reported being unable to understand the provided instructions, and additional instructions had been provided, with a simplified procedure. The owner of victim had reported that these instructions had failed to work correctly.

The owner of victim was contacted by telephone, and he reported unusual behavior on the system. In particular, the ps command was not working. The owner had rebooted victim, and attempted to troubleshoot the problem, but could not find anything wrong. The strange behavior persisted. He was also unaware of any IRC server running on the machine.

The security group at this point feared the possibility of a backdoor or sniffer on victim. The owner was asked to take no further action, and outside access to victim was blocked at the firewall. The security group then began containment of the incident.

Confirmation of the identity of the compensation attack was only possible after examining the system logs on victim. There appeared to be no recorded events in the IDS logs for the attack.

The annotated system logs retrieved from victim reveal several ssh connection attempts before the attack succeeded. The successful attack is shown below.

```
# Note:  breaks within each of the following blocks of data
# from /var/log/messages are denoted by ellipses (. . .).
# The breaks contain many iterations of the same error
# messages as those shown.

# Evidence of actual intrusion:  the intruder(s) apparently
# were sending buffer overrun strings of various lengths to
# the ssh daemon in an attempt to gain access:

Jan 14 08:13:07 victim sshd[14749]: log: Connection from
www.xxx.yyy.zzz port 1407
Jan 14 08:13:07 victim sshd[14749]: log: Could not reverse
map address www.xxx.yyy.zzz.
Jan 14 08:13:07 victim sshd[14750]: log: Connection from
www.xxx.yyy.zzz port 1809
Jan 14 08:13:07 victim sshd[14750]: log: Could not reverse
map address www.xxx.yyy.zzz.
Jan 14 08:13:07 victim sshd[14750]: fatal: Local: Your ssh
version is too old and is no longer supported.  Please
install a newer version.
. . .
Jan 14 08:13:54 victim sshd[14796]: log: Connection from
www.xxx.yyy.zzz port 1855
```

```
Jan 14 08:13:54 victim sshd[14796]: log: Could not reverse
map address www.xxx.yyy.zzz.
Jan 14 08:13:55 victim sshd[14796]: fatal: Local: crc32
compensation attack: network attack detected
. . .
Jan 14 08:15:00 victim sshd[14835]: log: Connection from
www.xxx.yyy.zzz port 1894
Jan 14 08:15:00 victim sshd[14835]: log: Could not reverse
map address www.xxx.yyy.zzz.

# Possible failed attempt by intruder(s) to erase their
# activity, evidenced by a restart of the system logger:
Jan 14 08:17:15 cactus syslogd 1.3-3: restart.
```

The most reliable indicator found in the logs was the report of "crc32 compensation attack: network attack detected." The crc32 compensation attack detector logs to syslog when it detects attempts to manipulate the crc32 checksum. The exploit also triggers this message to be logged. The CERT advisory notes this behavior. [CERT]

Available evidence was obtained from three sources, listed in order of importance.

The primary source was the file system of victim. Victim's file system was composed of three partitions, which were imaged, and the original disks were retained by the machine's owner. Permission was not given to save the disks for evidence purposes. The system owner feared the loss of a vital database, for which no backups existed. Because the originals could not be retained, a copy of the disk images was archived for future reference.

The second source of evidence was the IDS logs. These contained little information about the exploit itself, but were useful in determining the extent of intrusion onto the network after victim was compromised. They contained records of IRC activity on victim.

The third source of evidence was a set of notes compiled by the administrator of victim. These included records of maintenance done on the system, and troubleshooting logs from the period after the break in, before the compromise was detected.

Chain of custody for the gathered evidence was not maintained. There were several reasons for this. First, there were no procedures or infrastructure in place for secure handling of evidence. Second, the original evidence was unavailable for preservation, as previously explained. Third, the owner of victim was reluctant to draw attention to the incident, for fear of embarrassment. Finally, the time and attention necessary for anything beyond immediate restoration of systems and data was, and still is, considered an unnecessary distraction from research. This attitude is widely held at the institution.

C. Containment

The containment phase of the incident response began with the restriction of access to victim at the firewall. This was followed by a meeting of the security group. The group developed a plan for handling of the incident in the absence of established procedure. The trainee was designated as primary incident handler, and the security group made arrangements with the system administration group for support as needed.

The security group lacked the necessary hardware for forensic backups of hard disks, so the incident handler arranged for disk images to be archived over the network to one of the company operated servers at the institution, referred to as helper. The server's administrator created a TFTP server on the machine, and also prepared a boot floppy for victim with a TFTP client.

The boot floppy was based on Tom's root boot, available at (URL here)

Because of the need to access other machines on the network, victim could not be completely disconnected from the institution's intranet. It also could not be allowed to go unmonitored. To protect against victim attacking other machines, the analyst monitored all connections to and from victim using the IDS.

Once arrangements were made, the analyst contacted the owner of victim by telephone, to obtain the necessary permission to proceed. The owner's primary concern was preservation of the database on victim, and so would not release control of the original disks. He also requested that victim, and his whole subnet, be protected at the firewall with a more restrictive set of rules in the future.

During this call, the owner of victim expressed the belief that the company was at fault for the intrusion, at least in part, because the advice provided by the security group was inadequate for his level of system administration ability. He pointed out that he did not have time for the daily administrative chores of maintaining a web server.

The analyst informed him that owners of servers were responsible for administration of their machines. Such duties were beyond the scope of the company's contract. Additionally, security advice was a courtesy provided by the company for the protection of everyone, not an obligation.

Once preparations were complete, the incident handler proceeded to the site and assessed the state of victim.

A terminal session was established to victim from monitor, used to control victim and other servers located in a rack. The session was recorded in the terminal buffer, then saved to a file.

```
monitor % ssh victim.muppetlabs.org
Received signal 2.  (no core)
```

SSH was not working. This was not noticed before. Try to telnet in.

```
monitor % telnet victim.muppetlabs.org
Trying aaa.bbb.ccc.ddd...
Connected to aaa.bbb.ccc.ddd.
Escape character is '^]'.
Welcome to SuSE Linux7.2 (i386) – Kernel 2.4.4-64GB-SMP (1)

victim login: twm
password: ********

Last login: Thu Dec 13 10:04:22 from monitor
Wed Jan 16 14:45:11 EST 2002

victim % more /etc/passwd
root:x:0:0:root:/root:/bin/bash
<edited>
apached::1000:100:Apache Daemon:/sbin/apached:/bin/sh
smtpd::0:0:Sendmail:/var/spool/mail:/bin/sh
```

The final two entries were new. Neither had a password. The last one had uid 0, equivalent to root. The owner confirmed that neither entry belonged there. This confirmed that the machine had been compromised.

The incident handler at this point began gathering evidence for later analysis.

```
victim % su
Password:********

victim # mail root@helper.muppetlabs.org < /etc/passwd
victim # mail root@helper.muppetlabs.org < /etc/shadow
victim # mail root@helper.muppetlabs.org < /etc/group
victim # mail root@helper.muppetlabs.org < /etc/inetd.conf
victim # mail root@helper.muppetlabs.org < /var/log/wtemp
```

Next gather the running processes.

```
victim # ps –ef | mail root@helper.muppetlabs.org
bash: /bin/ps: No such file or directory
No message, no subject; hope that's ok
```

The owner had reported problems with ps.

```
victim # whereis ps
ps: /bin/ps /usr/share/man/man1/ps.1.gz
```

The command seems to be there in /bin but cannot be found.

```
victim # ls –l /bin
<edited>
-rwxrwxrwx    1 smtpd        root     39484 May 11 2001 ls
<edited>
-rwsrwxrwx    1 smtpd        root     31336 May 15 2001 ps
<edited>
```

The ls and ps executables are both world writeable. They were most likely replaced, but incident handler hasn't noticed yet. The ps command is clearly in /bin

```
victim # /bin/ps –ef | mail root@helper.muppetlabs.org
bash: /bin/ps: No such file or directory
victim # /bin/tcsh
victim # /bin/ps –ef
/bin/ps: Command not found.
```

The incident handler now realized that the system was seriously damaged at the very least. Uncertain whether it would work, the incident handler tried to view the processes one last time using top.

The top command revealed a process called "k", later found to be an IRC server, consuming nearly all spare CPU resources on the machine.

With compromise confirmed and a possible rootkit, the system commands could not be trusted. At this point, the system was shut down to permit imaging of the hard disks.

A keyboard and monitor were attached to victim to allow console operation. It was booted from the boot floppy.

The network was configured next. First the network card was enabled.

```
# ifconfig eth0 aaa.bbb.ccc.ddd broadcast aaa.bbb.ccc.255 \
        netmask 255.255.255.0 up
```

Then static routes were added to the routing table.

```
# route add -net aaa.bbb.ccc.0 gw aaa.bbb.ccc.ddd
# route add -net default gw aaa.bbb.ccc.1
```

Now victim could communicate over the network. The next step was to identify the disk partitions.

```
# fdisk -l
/dev/sda1       6 G
/dev/sdb1       7.5 G
/dev/hda1       8 G
```

Finally, the partitions were backed up for later analysis. The dd command was used to perform a byte for byte copy, with rsh to transport the byte stream over the network. The images were sent to helper, which had rsh enabled temporarily. Note that in the absence of a DNS client on the boot floppy, IP's are used. Also, the block size of 524288 was chosen in dd for efficiency. It is exactly one half of a megabyte.

```
# dd -if /dev/sda1 bs=524288 | rsh aaa.bbb.ccc.efg 'cat > /images/victim/ \
        victim.scsi1.img'
# dd -if /dev/sdb1 bs=524288 | rsh aaa.bbb.ccc.efg 'cat > /images/victim/ \
        victim.scsi2.img'
# dd -if /dev/hda1 bs=524288 | rsh aaa.bbb.ccc.efg 'cat > /images/victim/ \
        victim.ide.img'
```

Once the images had transferred, victim was once again shut down.

After the initial assessment of victim was complete, the incident response team focused on possible compromise of other machines from victim. Users of victim were informed that their passwords on all other systems could be compromised. The owner of victim was asked to identify any systems that trusted victim, and to consider them potentially compromised. None were identified.

Users of victim who also had accounts on machines controlled by the company, were required to change passwords. Their accounts were inspected for evidence of compromise. None was found.

The switched network limited any sniffer based on victim to observing only packets going to or from victim. Hosts on the network that did not share users with victim were considered to not be at risk unless they had been directly attacked.

D Eradication

Cursory examination of the disk images convinced the security group that victim needed to be completely rebuilt. The t0rn rootkit had been installed on the system. /bin/login had been replaced with a version containing a backdoor. OpenSSH had also been replaced with a version containing a backdoor.

The incident handler contacted the owner of victim to discuss rebuilding the system. The owner had available a new disk drive to build a fresh system on. Victim would rebuilt on the new disk, and the database recovered from the old disk.

The owner at this time requested a report of the incident response effort detailing the vulnerability used by the intruders, and any logs from the password sniffer. The incident handler reported that no logs had been found, and that all passwords on victim should be considered as potentially compromised.

The administrator for helper performed the forensic analysis of the evidence recovered from victim. This analysis showed that the intruders had used the SSH CRC32 compensation attack to gain access to victim. Analysis also revealed that FSecure SSH had been installed on victim one year previously. This install had disabled the OpenSSH server that had come with the default SuSE installation. This was the reason that patches applied to the OpenSSH server had failed to close the vulnerability.

E Recovery

After the owner of victim had completed rebuilding the machine, the analyst scanned it for known vulnerabilities, and reported it free of major problems. The LAN engineer then activated a set of new rules on the firewall specifically for victim and the other machines owned by the same lab. The new rules allowed almost no access inbound.

The incident handler prepared and submitted a report of the incident, including the findings of the forensic analysis. This concluded the investigation.

The owner of victim then requested a meeting with the manager of the network support contract, in order to voice a grievance. The incident handler was tasked to compile a report on all events regarding victim, both prior to the incident, and during the response. This report exceeded 100 pages.

At the meeting, the owner of victim demanded and received, funding for a full time Unix administrator. He also demanded and was denied, root access to the institution's firewall server. The issue of responsibility for the incident was not discussed.

The intruders were never identified. As resources were diverted from the incident response to the issue of documenting responsibility for the incident, it was decided that there was no significant benefit to the organization.

With the subnet containing victim behind a very restrictive set of firewall rules, no further action was taken to either harden the machine or ensure the safety of the rest of the subnet. Given the organizational structure, it is impractical for the Information security group to exert any pressure for action, and it has become practically impossible to offer any useful advice or assistance since the incident.

The subnet is still an area of concern, because it could be exploited from a compromised machine within the network. There is hope that as the laboratory gets its own resources, it will be able to secure its own subnet. We also watch traffic on that subnet very closely from the IDS.

F Lessons Learned

1 Avoid finger pointing

The resolution of this incident was incomplete at best. One of the problems was that the enormous amount of effort required to document the history of the incident for liability purposes. Eventually, the blame game overtook the whole investigation.

Once issues of blame are brought up, they are difficult to defer, and they can hamper the incident response process. Therefore, prevention is in order. Clear understanding of responsibilities and expectations could have minimized the problem in this case.

2 Document as much as possible

No matter how much is documented, you will wish you had more. Two main problems arose in documenting this incident.

Events that occurred over a year previously had relevance to the investigation. Since the participants did not know ahead of time what would be important, there were gaps in the paper trail.

The second problem was a lack of documentation caused when participants avoided written correspondence. Email can be preserved much more easily than telephone logs. Use written communication as much as possible. This can be difficult, if other parties refuse to cooperate. If a person knows their words can be used against them, they will avoid email.

3 Establish clear objectives for an incident response.

Not all responses will have the same objectives. Beyond cleanup and recovery, an incident response may lead to personnel action, or civil or criminal action. If parties involved have different expectations for the incident response process, friction can result. It may be required to state the goals of the incident response repeatedly, to avoid having the process diverted toward other purposes.

Level of preparation also limits what is possible for an incident response to achieve. If a particular goal, such as prosecution, is impossible or impractical, this needs to be recognized.

4 Cooperation is critical

The incident response team cannot function without cooperation. Cooperation can be fostered by establishing relationships among organizational units. Getting groups to buy in should be considered part of preparation.

5 Security and convenience do not mix

In an organization that permits users almost total local control for convenience and flexibility, high security, or anything resembling it, cannot be achieved. This is to be expected.

Users will also not understand why they don't have total security, but be unwilling to give up control to achieve security. This is also to be expected.

G Conclusion

Despite the problems encountered in handling the incident, it appears to have been successful to the extent that there have been no re-occurrences of the incident, nor further compromises of the network from it. Considering the lack of preparation and the inexperience of the Information Security group, not much more could have been achieved in the environment that existed at the time.

References

Barrett, Danial J, Silverman, Richard E, "SSH The Secure Shell, The Definitive Guide" O'Reilly and Associates, Sebastopol, CA 2001.

Caswell, Brian and Roesch, Marty, SNORT signatures
http://www.snort.org

CERT, "Exploitation of Vulnerability in SSH1 CRC-32 Compensation Attack Detector"
www.cert.org/incident_notes/IN-2001-12.html

Dittrich, David, "Analysis of SSH crc32 Compensation Attack Detector Exploit"
http://www.linuxsecurity.com/articles/intrusion_detection_article-4002.html

NeMeSiiy, SSH CRC32 compensation detector exploit code
http://packetstormsecurity.org/0103-exploits/openssh-2.2.0-exp.tgz

SuSE, "SuSE Security Announcement SuSE-SA:2001:04"
http://www.linuxsecurity.com/advisories/suse_advisory-1154.html

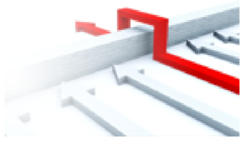
Zalewski, Michael, "Remote Vulnerability in SSH crc32 Compensation Attack Detector"
http://razor.bindview.com/publish/advisories/adv_ssh1crc.html

Last Updated: February 25th, 2020

Upcoming SANS Penetration Testing

SANS PENETRATION TESTING

Click Here to {Get Registered!}



SANS Northern VA - Reston Spring 2020	Reston, VA	Mar 02, 2020 - Mar 07, 2020	Live Event
SANS Munich March 2020	Munich, Germany	Mar 02, 2020 - Mar 07, 2020	Live Event
Northern VA - Reston Spring 2020 - SEC542: Web App Penetration Testing and Ethical Hacking	Reston, VA	Mar 02, 2020 - Mar 07, 2020	vLive
SANS Secure Japan 2020	Tokyo, Japan	Mar 02, 2020 - Mar 14, 2020	Live Event
Mentor Session @work- SEC542	Oklahoma City, OK	Mar 03, 2020 - Apr 02, 2020	Mentor
SANS St. Louis 2020	St. Louis, MO	Mar 08, 2020 - Mar 13, 2020	Live Event
Dallas 2020 - SEC504: Hacker Tools, Techniques, Exploits, and Incident Handling	Dallas, TX	Mar 09, 2020 - Mar 14, 2020	vLive
SANS Prague March 2020	Prague, Czech Republic	Mar 09, 2020 - Mar 14, 2020	Live Event
SANS Dallas 2020	Dallas, TX	Mar 09, 2020 - Mar 14, 2020	Live Event
SANS Doha March 2020	Doha, Qatar	Mar 14, 2020 - Mar 19, 2020	Live Event
Secure Singapore 2020 - SEC504: Hacker Tools, Techniques, Exploits, and Incident Handling	Singapore, Singapore	Mar 16, 2020 - Mar 21, 2020	vLive
SANS Norfolk 2020	Norfolk, VA	Mar 16, 2020 - Mar 21, 2020	Live Event
SANS Secure Singapore 2020	Singapore, Singapore	Mar 16, 2020 - Mar 28, 2020	Live Event
SANS London March 2020	London, United Kingdom	Mar 16, 2020 - Mar 21, 2020	Live Event
SANS SEC504 Nantes March 2020 (in French)	Nantes, France	Mar 16, 2020 - Mar 21, 2020	Live Event
SANS San Francisco Spring 2020	San Francisco, CA	Mar 16, 2020 - Mar 27, 2020	Live Event
SANS Kuwait March 2020	Salmiya, Kuwait	Mar 21, 2020 - Mar 26, 2020	Live Event
Secure Singapore 2020 - SEC560: Network Penetration Testing and Ethical Hacking	Singapore, Singapore	Mar 23, 2020 - Mar 28, 2020	vLive
SANS Seattle Spring 2020	Seattle, WA	Mar 23, 2020 - Mar 28, 2020	Live Event
SANS Madrid March 2020	Madrid, Spain	Mar 23, 2020 - Mar 28, 2020	Live Event
SANS Oslo March 2020	Oslo, Norway	Mar 23, 2020 - Mar 28, 2020	Live Event
Community SANS Austin SEC504 @ CISCO	Austin, TX	Mar 23, 2020 - Mar 28, 2020	Community SANS
SANS SEC560 Lyon March 2020 (In French)	Lyon, France	Mar 23, 2020 - Mar 28, 2020	Live Event
Community SANS Vancouver SEC560	Vancouver, BC	Mar 30, 2020 - Apr 04, 2020	Community SANS
Community SANS Cincinnati SEC542	Cincinnati, OH	Mar 30, 2020 - Apr 04, 2020	Community SANS
SANS Philadelphia 2020	Philadelphia, PA	Mar 30, 2020 - Apr 04, 2020	Live Event
Community SANS Ottawa SEC560	Ottawa, ON	Mar 30, 2020 - Apr 04, 2020	Community SANS
SANS Frankfurt March 2020	Frankfurt, Germany	Mar 30, 2020 - Apr 04, 2020	Live Event
Mentor Session - SEC504	Austin, TX	Apr 01, 2020 - Jun 03, 2020	Mentor
SANS 2020	Orlando, FL	Apr 03, 2020 - Apr 10, 2020	Live Event
Mentor Session - SEC504	Denver, CO	Apr 03, 2020 - Apr 24, 2020	Mentor