# Global Information Assurance Certification Paper

GIAC Certified Incident Handler (GCIH) Practical Assignment Version 3:

The Educational Facility's Network

Timothy C. Hall

Statement of Purpose

Historically, the LAN/WAN systems contained within educational facilities have exhibited a propensity towards loose network security. Many times such networks are heterogeneous, understaffed, and those who do staff them have a tremendous workload placed upon them. Security often times takes a back seat to other issues considered more pressing. This paper will show a plan of attack that is frequently used to compromise networks, such as our mock educational facility's, from the inside by exploiting the vulnerabilities often present within. The consequences of such an attack could be quite ugly. User ids and passwords for multiple types of network accounts, grades, email, instant messages, addresses, telephone numbers, and online banking or credit card information would be readily available. This says nothing of file sharing services, IRC bots, kernel level root kits, sniffers, denial of service attacks, etc. The tools and exploits used for this demo will be KeyLog V1.1 by J.Daniel Pino, and the Linux kernel ptrace/kmod exploit implemented by Wojciech Purczynski. Screenshots are also included, and knowledge of the following items is helpful:

- 'Knoppix' (http://www.knopper.net/knoppix/)
- 'PuTTY' (http://www.chiark.greenend.org.uk/~sgtatham/putty/)
- 'Lynx' (http://lynx.browser.org/)
- Experience using a text editor in Linux (vi, pico, emacs)
- How to create and modify users in Linux is also a plus.

**Note:** Scenarios described in this paper, i.e. network attacks, and the network diagram, are fictional. Any resemblance to actual networks or attacks is purely coincidental. If such activities were to be carried out for testing and evaluation purposes, explicit written permission from all parties owning equipment, running services, and having responsibility over said equipment and services would have to be obtained in advance. Protect yourself!

Tools and Exploits

1. KeyLog Version 1.1

   By J. Daniel Pino (Daniel_2ar<at>hotmail.com)
   CVE: N/A
   Bugtraq ID: N/A
   Operating Systems Affected: Windows 9x/NT/2000/XP
   Protocols/Services/Applications Affected:
   - System keys and text input via the keyboard for most applications, client login screens, web forms, etc. are logged to a file
   - Which application is currently in focus is logged to a file
   - Application launch time and date are logged to a file

   Variants: (too many to list, this is a small amount)
   Hardware-based
   - 'KEYKatcher' at http://www.keykatcher.com
   - 'Keylogger'™ at http://www.amecisco.com

Tools and Exploits (continued)

Software-based

- Mikkotec's 'KeyKey 2002 Professional' at http://www.mikkotech.com/keykey.html
- 'sklog' by red0xd at http://skl0g.cjb.net/

Description and Operation

KeyLog Version 1.1 has been seen distributed in at least two different downloadable archives. One archive is a *.zip file containing the application binary, the *.dll file it needs to work, and an informational text file describing KeyLog Version 1.1's features and usage. The other archive contains the binary, *.dll, usage and features information, and the assembly language source code with compilation instructions. KeyLog Version 1.1 is a software-based keylogger.

The program is relatively simple to install and is written in MASM32 according to its 'Readme.txt.' It records standard and system keys depressed as text is input into login clients, web forms, word processors, chat programs, email applications, etc. In addition, it logs which application is given focus on the system, and the time at which any of the above events occurred. A sample key capture file is located in the 'Appendix' of this paper.

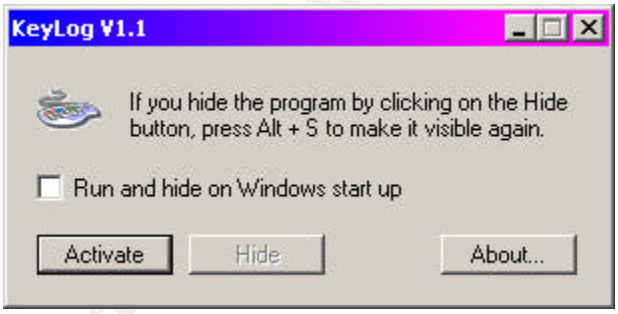



Figure 1: The KeyLog V1.1 program interface.

Installing

To install KeyLog V1.1, simply extract the files from either archive to a location on a computer's hard disk. To run it, browse to the location where 'KeyLog.exe' was extracted and double-click on it. You will see the user interface shown in 'Figure 1.'

Activating and Deactivating

To begin logging keystrokes, click the button labeled 'Activate.' 'KeyLog.exe' will begin logging information it gathers into a file called 'Klgf.txt' located at the root of the system drive. The output is logged to this file regardless of where 'KeyLog.exe' is run from in the non-hidden mode (see below). The information contained in the log file can only be examined if 'KeyLog.exe' is not running. The file is locked while the program is running. You will also notice that the 'Activate' button now reads 'Deactivate.' To stop the logging process, click the 'Deactivate' button.

## Tools and Exploits (continued)

### Running Hidden

Once KeyLog is running, the user interface can be hidden from view. This is accomplished by clicking the 'Hide' button. Once the 'Hide' button has been depressed the program disappears from view; however, it can easily be recalled by typing 'Alt-S.' There is also a check box that enables you to cause KeyLog to be launched and run hidden each time Windows is restarted. Once 'Run and hide on Windows start up' has been checked the 'Hide' button must be depressed in order for the keylogger to run in this manner. This behavior is accomplished by means of a registry entry being made on the targeted computer. On Win9x machines, this is normally a trivial issue; less so with NT/2K/XP systems. A key is added in the 'HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run' section of the Windows registry where programs that need to be launched upon startup are placed. The key contains the installation path to the KeyLog executable file as detected by the program when it was launched with the 'Run and hide on Windows start up' option checked.

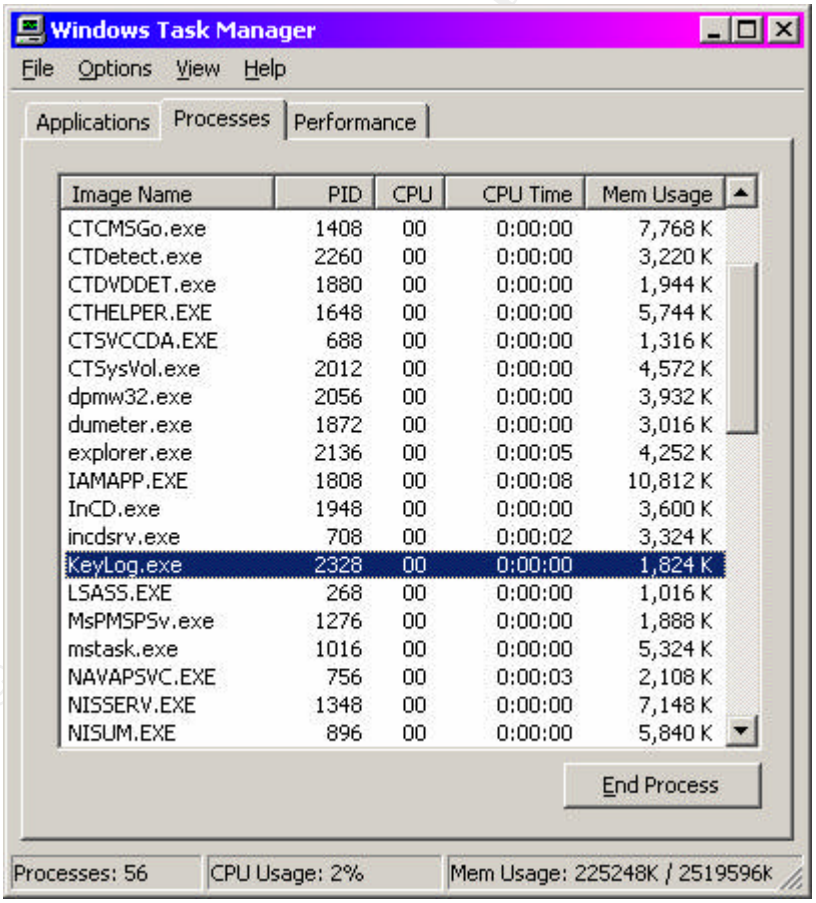



Figure 2: Task Manager with 'KeyLog.exe' running.

Tools and Exploits (continued)

'Hide' will also hide KeyLog's running process from the task manager on Windows 9x machines. This is not true with Windows NT/2K/XP machines. The running application can be seen listed as shown in 'Figure 2.' Consequently, it is easy to detect and kill in these operating systems. If the program is moved, it must be restarted with the run hidden option selected again in order to cause KeyLog to update the registry key with the new location.

About

Let us not forget about the 'About' button! Clicking on this button will tell you who wrote this little program, and where they may be contacted. See 'Figure 3' below to view the information.


Figure 3: About KeyLog Version 1.1

Signs of a KeyLog V1.1 Installation:

If a computer has any of the files listed below with the delineated specifications, there is a good possibility that KeyLogV1.1 is installed on it.

- The program 'Keylog.exe' listed in the Windows Task Manager display on NT/2K/XP machines is another sign of installation and log activity.
- Another indication is the presence of the file 'Klgf.txt.' at the root of the system drive. This file will be locked if the keylogger is running and the data it contains will not be available for inspection unless the keylogging applications process is killed. If a registry key is found in 'HKLM\SOFTWARE\Microsoft\Windows\Current-Version\Run' that reflects the installation path of the 'KeyLog.exe' file similar to the picture shown in Figure 4, then the machine has KeyLog V1.1 installed. Note the key labeled as 'KeyLogRegEntry.' The 'Data' section of the highlighted key contains the path and command line options to launch KeyLog and run it hidden.

## Tools and Exploits (continued)

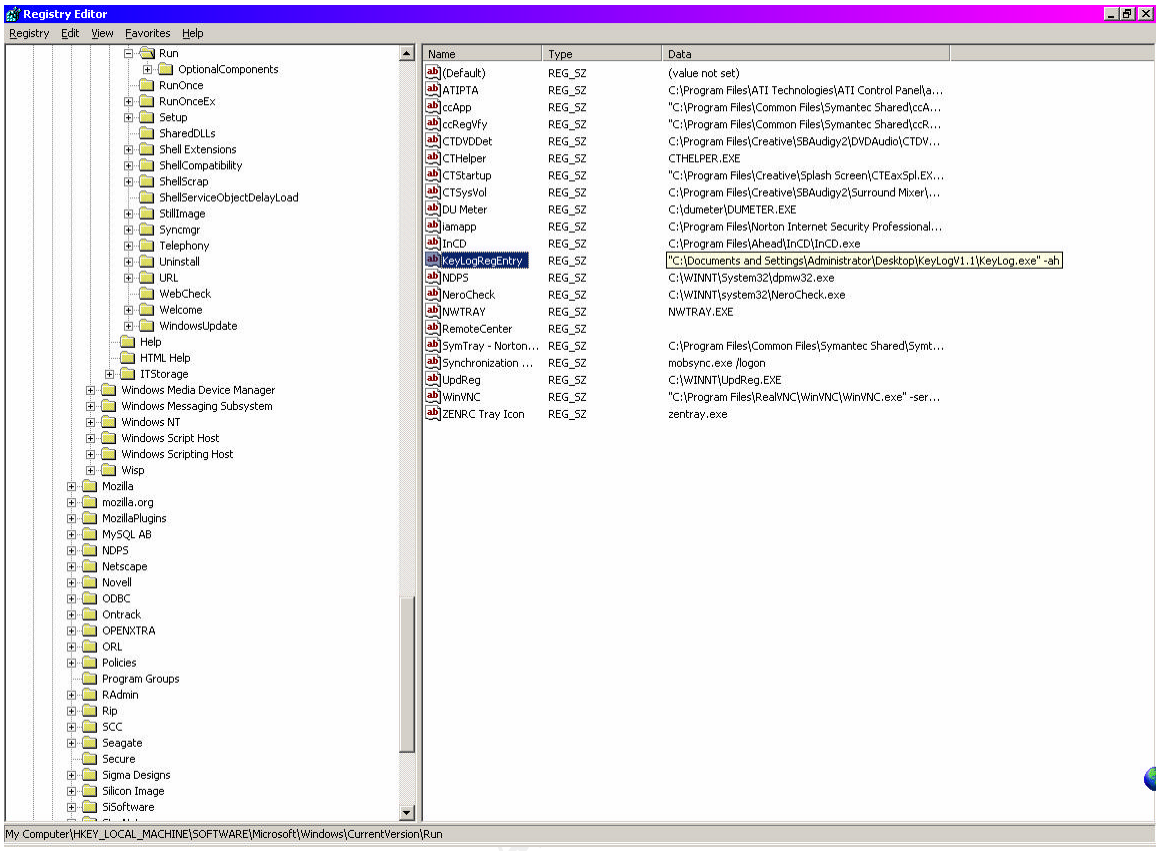



Figure 4: The key reflecting the installation path to KeyLog is shown here.

### Signatures of KeyLog V1.1

- File Specifications: While any of the files listed below could be found on a machine with KeyLog installed, two are critical to its operation. They are 'KeyLog.exe' and 'HKL.dll. Both files need to be placed in the same directory for the program to run properly.

Tools and Exploits (continued)

- File A (contents below):'KeyLogV1-1_source_code.zip' 36,864 bytes
- MD5 Sum: 85d305d43c8a992b3f68e98eac8eda67

| File Name | File Size | Date/Time Modified |
|---|---|---|
| 'KeyLogV11_source_code' (folder) | N/A | N/A |
| 'HKL.dll' | 3,072 bytes | 01/16/2002@ 11:59 AM |
| 'KeyLog.exe' | 13,824 bytes | 07/22/2002@ 02:32 AM |
| 'Readme.txt' | 286 bytes | 07/19/2002@ 06:22 AM |
| 'Build.bat' | 175 bytes | 07/22/2002@ 02:09 AM |
| 'KeyLog Reference.txt' | 1,679 bytes | 07/03/2002@ 05:11 AM |
| 'KeyLog.asm' | 59,283 bytes | 07/22/2002@ 02:32 AM |
| 'KeyLog.ico' | 2,238 bytes | 07/02/2002@ 10:09 PM |
| 'keylog.obj' | 14,557 bytes | 07/22/2002@ 02:32 AM |
| 'KeyLog.rc' | 1,509 bytes | 07/03/2002@ 11:08 AM |
| 'KeyLog.RES' | 3,496 bytes | 07/22/2002@ 02:32 AM |
| 'resource.h' | 1,183 bytes | 05/18/2001@ 01:45 AM |
| 'HKL_DLL' (folder) | N/A | N/A |
| 'Build.bat' | 146 bytes | 07/22/2002@ 02:08 AM |
| 'HKL.asm' | 12,413 bytes | 07/22/2002@ 02:09 AM |
| 'HKL.def' | 128 bytes | 01/09/2002@ 01:36 PM |
| 'HKL.exp' | 1,045 bytes | 02/22/2002@ 02:09 AM |
| 'HKL.inc' | 83 bytes | 12/06/2001@ 01:44 AM |
| 'HKL.lib' | 2,554 bytes | 02/22/2002@ 02:09 AM |
| 'HKL.obj' | 2,013 bytes | 02/22/2002@ 02:09 AM |

- File B (contents below): 'KeyLogV1.1.zip' 8,192 bytes
  MD5 Sum: 0e9e7ad0b3a0382ae78ea26eac5eb8b2

| File Name | File Size | Date/Time Modified |
|---|---|---|
| 'HKL.dll' | 3,072 bytes | 01/16/2002@ 11:59 AM |
| 'KeyLog.exe' | 13,824 bytes | 07/22/2002@ 02:32 AM |
| 'Readme.txt' | 1,679 bytes | 07/03/2002@ 05:11 AM |

- File C: 'Klgf.txt'

This file is of variable size and content, and is where KeyLog writes what it is able to capture. It is located at the root of the system drive. The file will have content similar to the example shown below in 'Figure 4a. Note that system keys are also shown along with which application got focus at what time.

Tools and Exploits (continued)

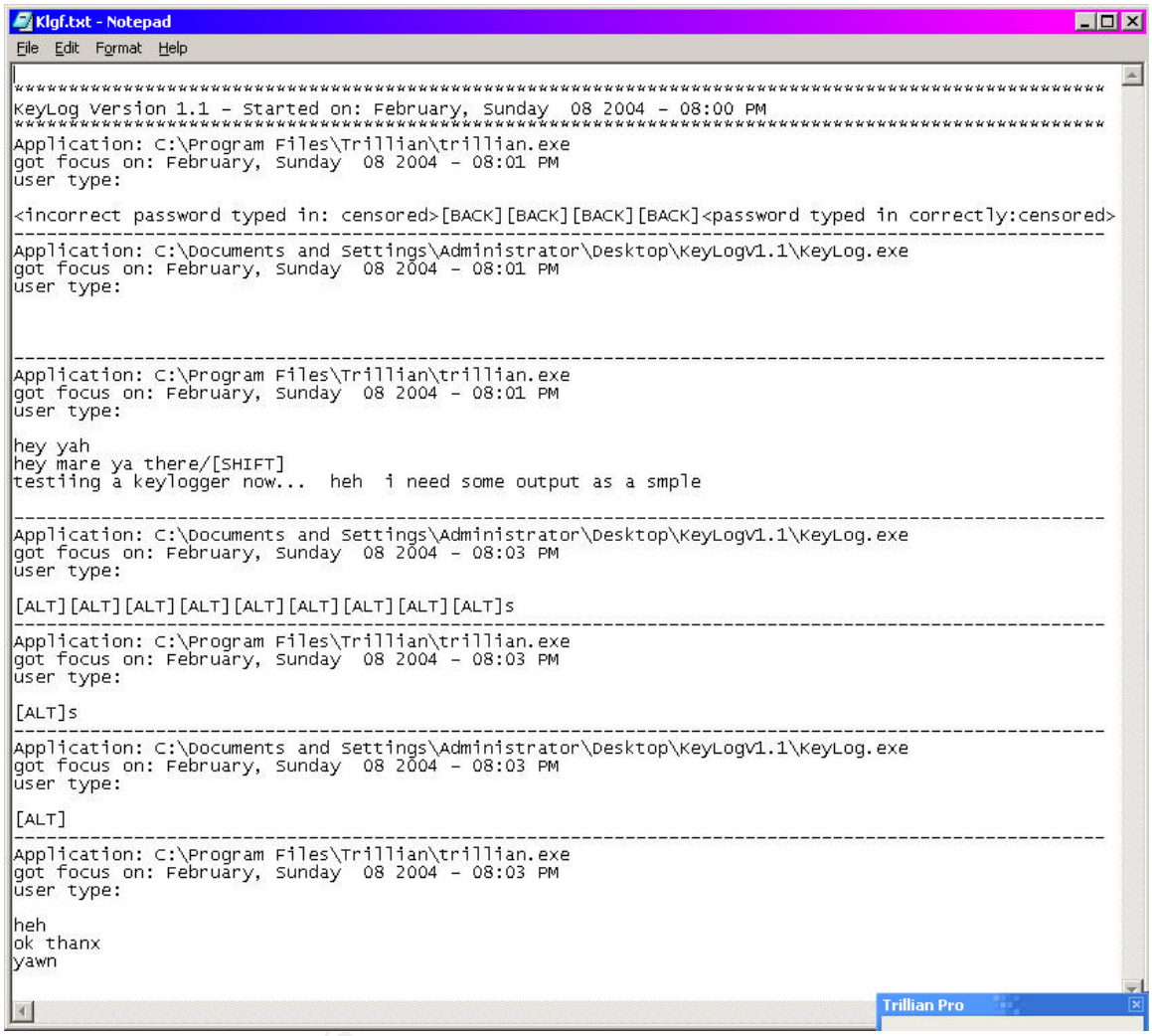



Figure 4a: The contents of 'Klgf.txt.'

2. The Linux Kernel Ptrace/Kmod Exploit Script 'ptrace_kmod.c'
   By Wojciech Purczynski of http://www.isec.pl/news.html (cliph<at>isec.pl)
   Discovered by Andrzej Szombierski (anszom<at>v-lo.krakow.pl)
   CVE: CAN-2003-0127 (under review):
   http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0127
   Bugtraq ID 7112
   http://www.securityfocus.com/bid/7112

Tools and Exploits (continued)

Operating Systems Vulnerable:

In general, Linux systems running the following kernels are vulnerable:

- Linux kernel 2.2.x up to 2.2.24
- Linux kernel 2.4.21 pre1
- RedHat Enterprise Linux AS 2.1 IA64 kernel
- RedHat Linux Advanced Work Station 2.1 kernel
- Some IA64 architecture kernels other than Red Hat's listed above

Note: Since so many different versions of the Linux kernel have been affected with this vulnerability, and because more reports have been made since the vulnerability went public March 17, 2003, it would be a good idea to consult the article at the URL above to keep current with what specific distributions and kernel versions are vulnerable. When this paper was written the Bugtraq article referred to above was updated as recently as January 15, 2004.

Protocols/Services/Applications Affected: Virtually any protocols, services, or applications running on a vulnerable Linux machine can be affected by this exploit. There is also the possibility that other Linux hosts the affected system has access to could also be exploited. An important pre-condition that must met is that the user MUST have a local account on the machine that is to be exploited. It is not remotely exploitable, except under conditions where remote logins are allowed, such as SSH or other remote protocols.

Variants

There are variants of this particular exploit according to the SecurityFocus Vulnerability Database listing at http://www.securityfocus.com/bid/7112/exploit/.

- 'km3.c' by Andrzej Szombierski (anszom<at>v-lo.krakow.pl)
- 'kmexp.c-bugtraq.c' by anonymous KuRaK
- 'myptrace.c' by snooq
- Combined with other vulnerabilities as in 'OpenFuck.c' by (SPAX<at>zon-h.org). See also http://packetstormsecurity.nl/0303-exploits/OpenFuck.c

Description and Operation

To understand how this ptrace exploit works, what ptrace is supposed to do should first be examined. According to the Linux 'man' page on ptrace displayed at http://www.die.net/doc/linux/man/man2/ptrace.2.html,

> 'The ptrace system call provides a means by which a parent process may observe and control the execution of another process… It is primarily used to implement breakpoint debugging and system call tracing…'

Under normal conditions, ptrace can spawn a child process from a parent process. At first, this process will have the effective userid and effective groupid of 0, which is root.

Tools and Exploits (continued)

Then, ptrace is supposed to change the process' effective user and group ids to those of the user running the program that caused ptrace to be needed. Once this change in access permissions happens it is safe for the child process to begin doing some work, and everyone's happy.

In a vulnerable kernel, the ptrace system call can be used with the kernel's module loading program (modprobe) and cause an unwanted gain in the privileges of a local user to those of root. This is due to a mistake that was made in the way 'kmod.c,' the code for the kernel's module loader, was written. A problem known as a race condition occurs. A race condition can loosely be defined as a timing issue where one process attempts to carry out its work on another process at precisely the right time for something unpredictable to happen. Chapter 3 of the 'The FreeBSD Developers' Handbook' located at http://www.freebsd.org/doc/en_US.ISO8859-1/books/developers-handbook/secure-race-conditions.html has this to say about race conditions:

> 'A race condition is anomalous behavior caused by the unexpected dependence on the relative timing of events. In other words, a programmer incorrectly assumed that a particular event would always happen before another.'

Therefore, a flaw in the programming code for the Linux kernel's module loader causes a race condition in ptrace. This, in turn, causes ptrace to have bad timing at a critical task it needs to perform: the changing of access permissions for the child process spawned. Due to the timing issue, at the point where ptrace should be changing the permissions of that child process to those of the program running the parent process that spawned the child, we can write shell code into the memory area where the child process runs from; and whatever is there will be executed with the privileges of w00t! YIKES! The shell code for launching a command prompt can be put in that area of memory and we get a root command shell! Ouch! The exploit is what drops the shell code into the memory area at the correct time. A copy of the code can be found at http://www.securiteam.com/exploits/5CP0Q0U9FY.html.

The ptrace exploit is easy to compile and use, as we will see. After the script has been on a machine that has 'gcc,' it can be compiled into a binary executable file and run as shown in 'Figure 5' on the next page. This was done using an SSH session to a bootable CD distro of Linux known as 'Knoppix 3.1.' The homepage for Knoppix is located at http://www.knopper.net/knoppix/. Knoppix 3.2, and 3.3 are not vulnerable to this. However, it is probably safe to assume that this (and other exploits) could be used on other bootable distributions as well since they are all static in nature.

Tools and Exploits (continued)

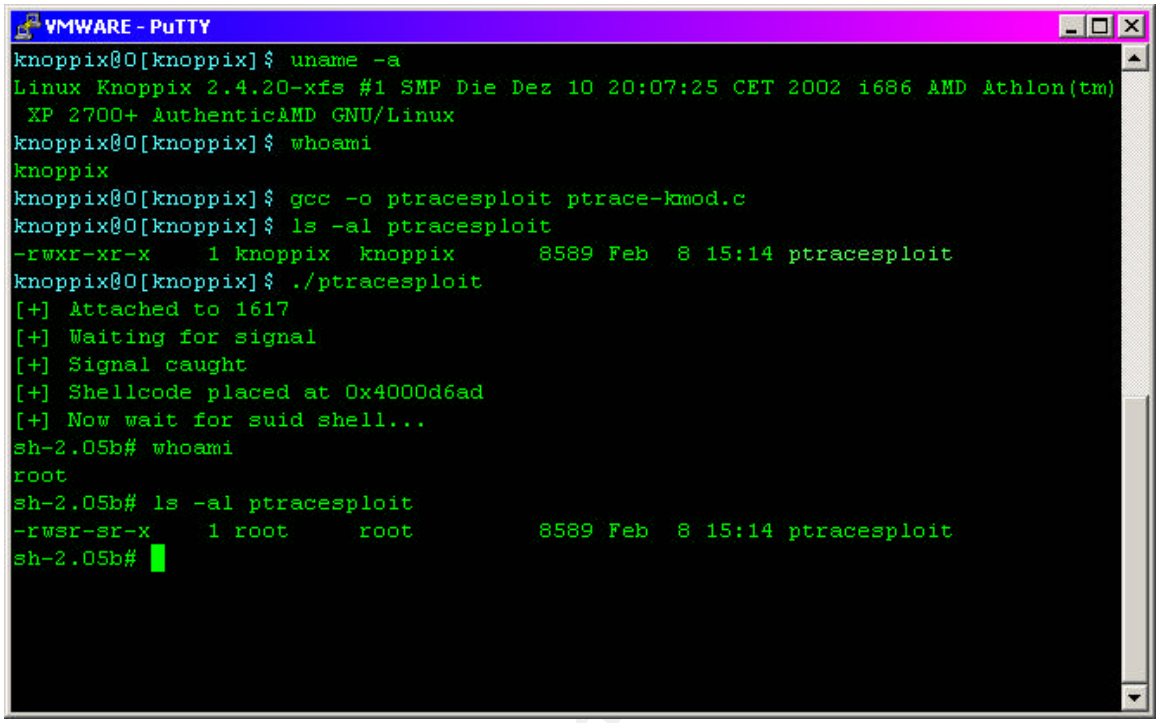



Figure 5: Here, the exploit obtained from compiling 'ptrace_kmod.c' can be seen in action. Note how the resulting binary's file permissions undergo a change that should not be possible for the regular user 'knoppix.'

Here is a breakdown of the steps taken on the vulnerable machine to exploit it. Note that if this were an actual exploit the perpetrator would probably want to keep it simple and fast. They would not want to take the time of compiling the executable on the victim machine; they would simply upload one that had been pre-compiled. In addition, they may not worry about trying to see who they were logged in as until AFTER running the exploit. Those steps have been included for the purpose of demonstration only. What is not shown is actually downloading the exploit.

- The user uses the GNU C Compiler to compile the attack script 'ptrace-kmod.c' into a program called 'ptracesploit.' The command syntax for that was 'gcc –o ptracesploit ptrace-kmod.c.'
- The user then runs the exploit with the command './ptracesploit' and the script achieves the desired goal of launching a command shell as root.
- The user then checks to see if they are indeed root with the command 'whoami. Own@Ge!!! We are now w00t!

Tools and Exploits (continued)

Signatures of the Ptrace/Kmod Exploit: The ptrace/kmod exploit has a signature that is low profile in nature. For our example compromise, the original C code filename of 'ptrace-kmod.c' was kept. The name we chose to give the exploit was 'ptracesploit.' These names could vary widely in an actual situation.

In 'Figure 6' we see a breakdown of each file and its characteristics. Notice how the user and group ownership, as well as the file permissions change after the exploit has been executed. The file originally had the file permissions of '-rwxr-xr-x' and it was owned by 'knoppix' and could be executed by members of the group 'knoppix.' After 'ptracesploit' has been executed, the file permissions change to '-rwsr-sr-x' and its group membership and owner have changed to 'root.' Now, when executed, 'ptracesploit' will run with the permissions of root. It can be used repeatedly and the same result will occur. However, it appears to carry out different actions after it has been run once. The messages that were printed to the screen in 'Figure 5' are not shown the second time the exploit is run. Instead, the user is dumped directly into a shell. 'Figure 7' shows '/bin/sh,' the process started by the exploit. Note also, that the process is running as root.

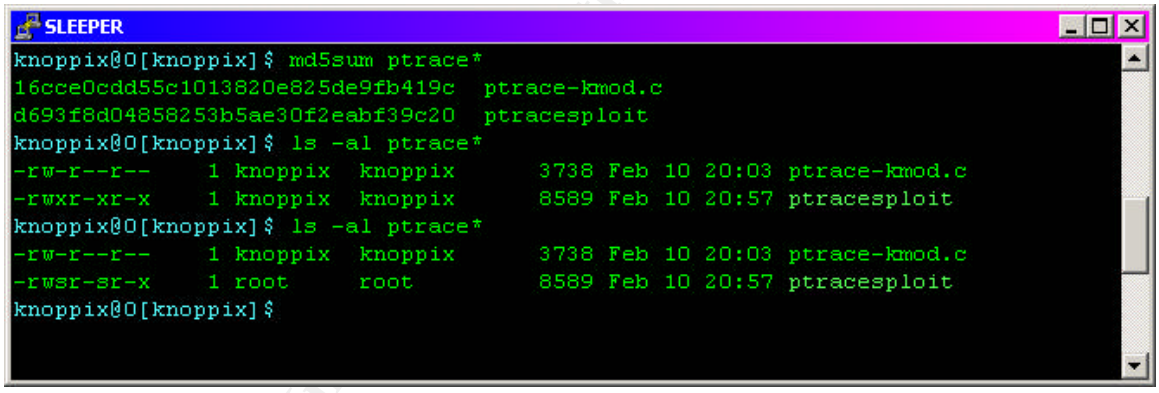



Figure 6.

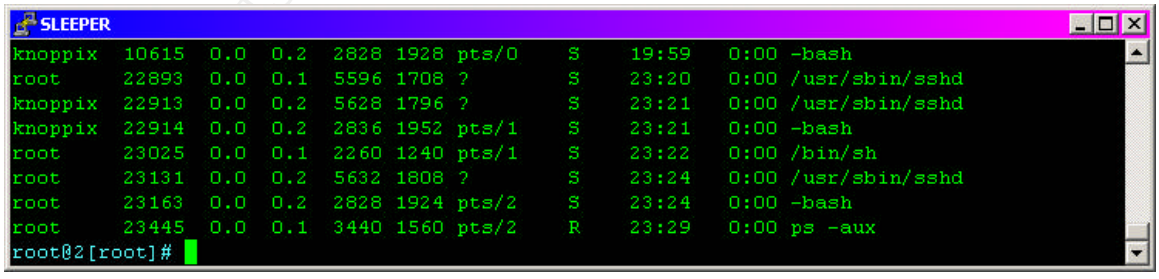



Figure 7.

## The Platforms and Environments

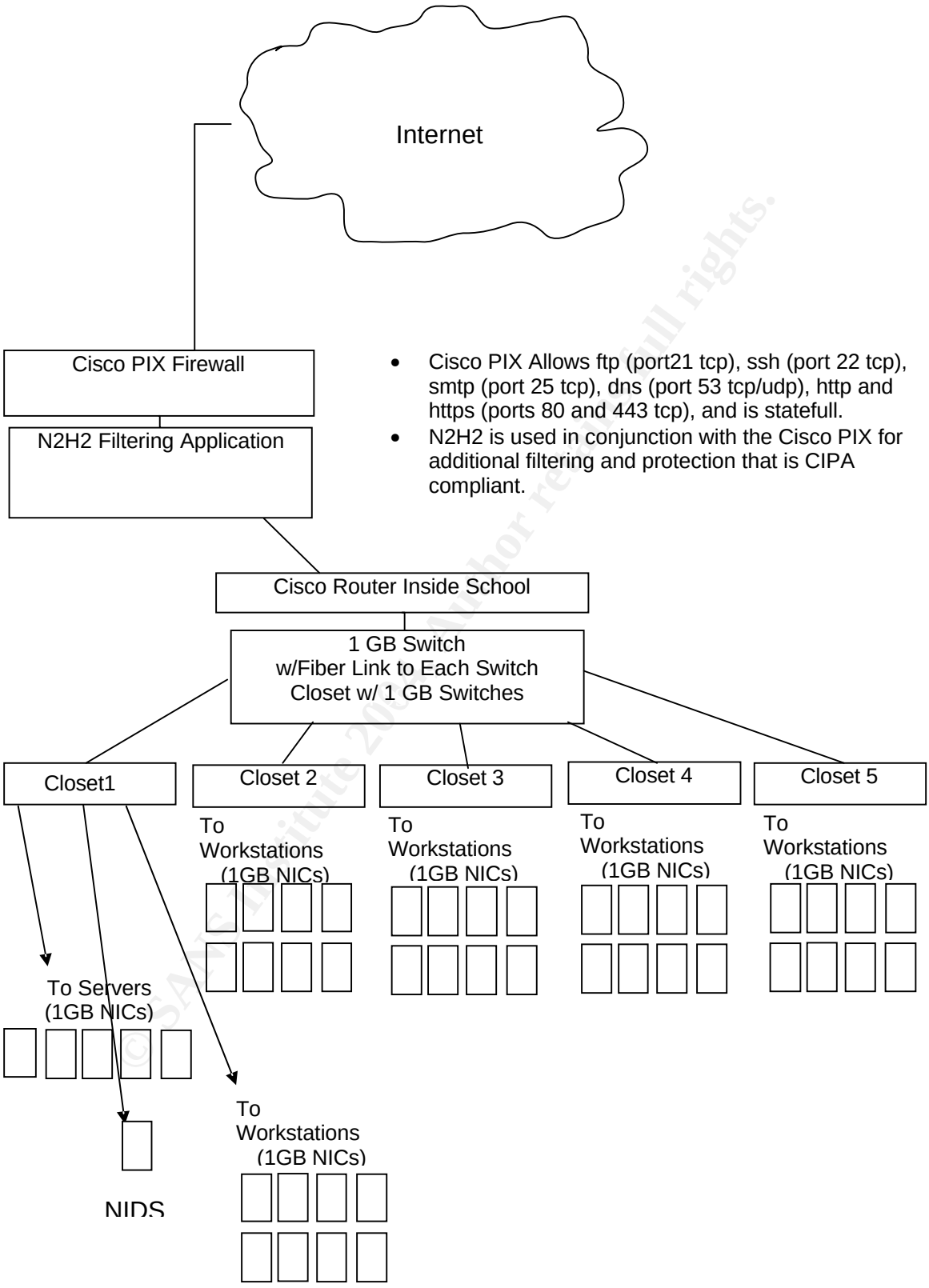
Victim's Platforms: Workstations: The sample school's network contains six hundred Windows 98SE workstations with all the applications and the operating system fully patched at levels current for the time, the Novell Netware client for Windows 98 3.3.2, PuTTY Client, MS Office 2000 Professional, and IE 6. In addition, there are ten Windows 2000 Professional workstations with all applications and the operating system fully patched at levels current for the time, Novell Netware client for Windows NT/2K/XP 4.8, PuTTY Client, MS Office 2000 Professional, and IE 6. These workstations all have Internet access and utilize private IP addresses. They also have antivirus protection which is kept current through a scripting process.

Servers: Main file storage services are facilitated via four server-class machines running Netware 5.1 with all NLMs and the operating system components at service pack levels recommended for the time. The Netware servers are also running the IPX/SPX protocol, IP protocol, and Novell Zenworks 3.2. These servers are not visible to the Internet and have multiple private IP addresses bound to the network interface card. A firewall prevents any traffic from being routed between these servers and the Internet. In addition, one Red Hat Linux 7.3 server with: C/C++ and JAVA development packages installed; FTP services; OpenSSH Protocol 1 services; Apache HTTPD with OpenSSL, PHP, MySQL; Sendmail services; and Telnet services is also used on the network. This server has Internet access and is behind a firewall that has the appropriate ports allowed with regards to the services running on it. Another Linux system is installed on the same network segment as the servers. The traffic to and from the servers has been mirrored to the switchport that this machine uses and a NIDS has been installed. Since the NIDS puts the network card in promiscuous mode for sniffing, it will be able to sample all of this traffic.

Firewalls and Filtering: All traffic from the Internet is firewalled using a Cisco PIX firewall. Filtering, so as to be compliant with the CIPA, is done with the 'N2H2' filtering application. See http://www.ifea.net/cipa.html for more information on the CIPA.

Source and Target Networks: The sources of the attacks on this network originated from several Windows 98SE workstations located inside of our educational facility, and from a machine located overseas. The data on the overseas network was not available. The intended target was the Red Hat server located on the school's premises. A diagram of the network is included on the next page.

Network Layout




- Cisco PIX Allows ftp (port21 tcp), ssh (port 22 tcp), smtp (port 25 tcp), dns (port 53 tcp/udp), http and https (ports 80 and 443 tcp), and is statefull.
- N2H2 is used in conjunction with the Cisco PIX for additional filtering and protection that is CIPA compliant.

## Stages of the Attack

### Background

The risk of an attack on our sample LAN is greatest from inside of the educational facility as there are large numbers of students and other individuals that utilize the network resources available therein. The attacker, in this case, is a fictitious student taking C/C++ programming and Cisco training. We'll assume the role of the fictitious student in this situation, so try to follow. *Events are listed roughly in order*...

"In orientation I was told that for my courses of study, I would be furnished with the following things…"

- A userid and password to access Windows 98SE/Windows 2000 Professional workstations that attach to Novell file servers. Various applications are run from the Novell servers, and I have a personal home directory located there. In addition, all workstations have standardized installations of MS Office 2000 Pro, PuTTY telnet/SSH client, and various educational applications.
- Another login using the same userid and password was created for me on the Linux server. I also have a home directory here, in addition to access to C/C++ compilers and a host of other applications.
- The same userid and password is used to login to the school website that I can consult for homework, administrative information, class forums and message boards, the school newspaper, communicate with instructors, etc.

"Not bad! I already know how to use some of these things…"

### Reconnaissance and Scanning

Reconnoitering needs to be done so a plan for attacking the network can be developed and carried out. If an attack were going to be executed from outside of the school's network, different means of reconnaissance would be used than what is described here. Instead of being able to make firsthand observations regarding the targeted network, one would have to use means that utilize whois database queries, searches of public records regarding the company or school, domain name registration, or http://www.samdspade.org. Any of these sources can reveal plenty of information from an'outsiders' perspective of the network if we don't have the benefit of an insider's viewpoint.

From the inside, there are many useful things to be observed. For example, there are several labs with thirty computers, and there is at least one computer in each classroom for teacher use only. The school media center is the largest lab, having over fifty computers. The offices of the particular departments have several computers each. All are Windows workstations. Teachers have Internet access, take attendance over the web, and have access to internal/external email. Many teachers and students are

Reconnaissance and Scanning (continued)

lax about leaving their network accounts logged in while they are away from the computer. Many students lose their student ID badges, or have them stolen. Error pages on the school website say that the server is an Apache server running Red Hat Linux as shown in 'Figure 8' below.

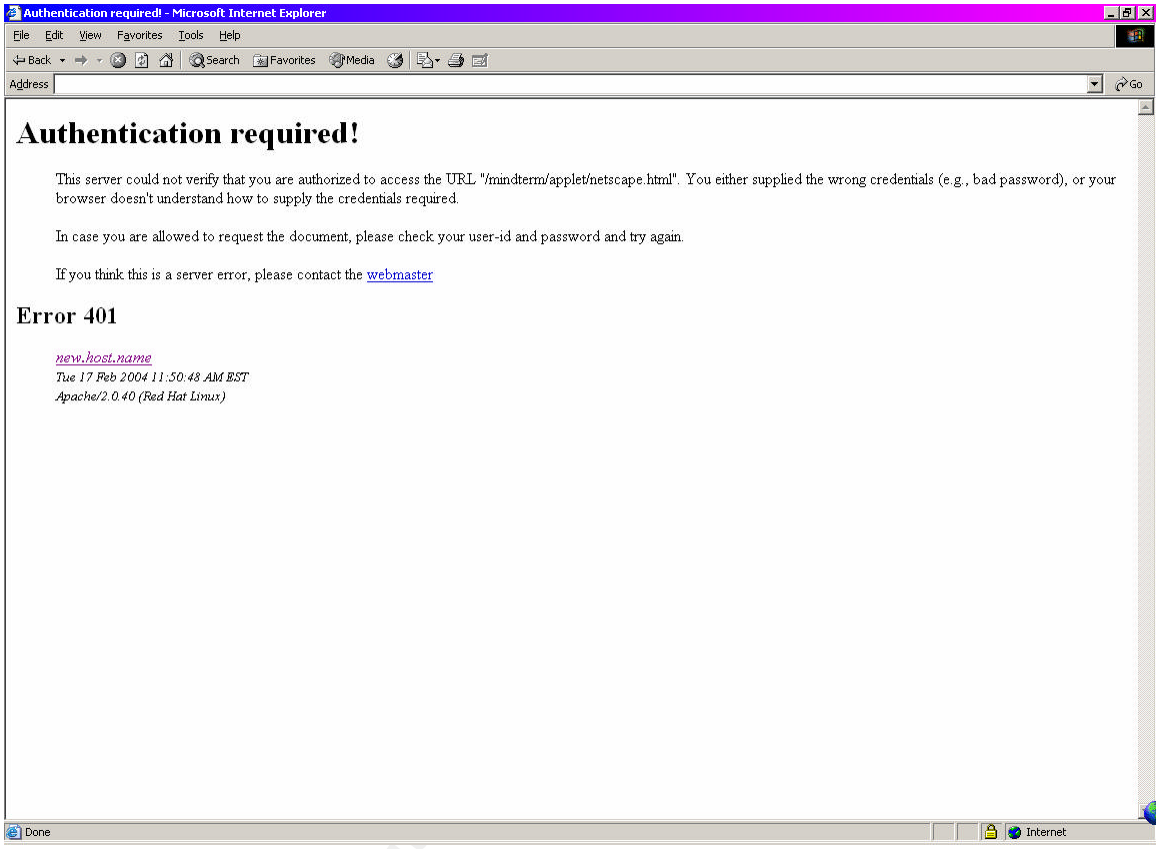



Figure 8: An authentication error '401' page. Note that the web server version and the operating system are sent back to the user if they enter an incorrect login on the site.

Each of the points stated above can present the right conditions for an attack to happen by virtue of the fact that potentially sensitive information is discovered through observation, and through a bit of experimentation. We didn't even need a port scanner.

For the next step, I'll use the computers in our fictitious school's media center. Since they are running Win2k Pro, we can now install a keylogger via a USB drive. This is done over a few days time to different machines that we check out for use. Teachers will often use machines in the media center to check email, do attendance, grades, upload things to the school's website, and print large projects. Since this area gets used quite often, a few strategically placed keyloggers will be enough. Maybe the media center staff or a systems administrator will log on to fix something. A keylogger here could yield some juicy userids and passwords.

Reconnaissance and Scanning (continued)

Next, we install the keylogger on a Win98SE machine used in programming class. It is the same computer we use each day. Maybe a person with more privileges like the system administrator or a network intern will log on using this machine.

"Now I'll wait a bit and let's find out what the keyloggers reveal…"

A Quick Synopsis of Where We Are

Our attacker has used information available to them to through their orientation with the school, experimentation with the school website, and some observations made about the way things seem to operate inside the educational facility to develop the beginnings of an attack plan. Thus far they already have access to the inside of the network as a regular user, they know some of the different operating system platforms in use on the school network, and they know what kind of web server is running on the Red Hat Linux server. In addition, keyloggers have been installed to try to gather some userids and passwords to use.

Exploiting the System Part One: KeyLogV1.1

After a bit of time has passed I return to the machines in the media center and harvest the log files made by the keyloggers. Eventually, I locate a userid and password for a teacher in the log file. They were using the PuTTY telnet/ssh client. The log entry is shown below in 'Figure 9.' The teacher was probably logging in to their shell account somewhere. Maybe even here on the school's web server where my shell account is. It's possible that I can try to login using this teacher's userid and password. I try login in under SSH as it is encrypted and telnet isn't. DONE! I have access as this teacher. I better logout now though. I don't need to be discovered. After a bit more reading of log files later, I find the login and password of a person who assists with maintaining the network. I recognize that userid from watching them work on computers before. They too were using PuTTy. Nice! With this userid and password, I'm able to clean up the keyloggers from the machines in the media center. It seems to have additional rights to do things to the workstations. I have to be careful doing this so I am not caught computer-hopping or using this account. Good thing I have this USB drive to store the log files and my other toys on!

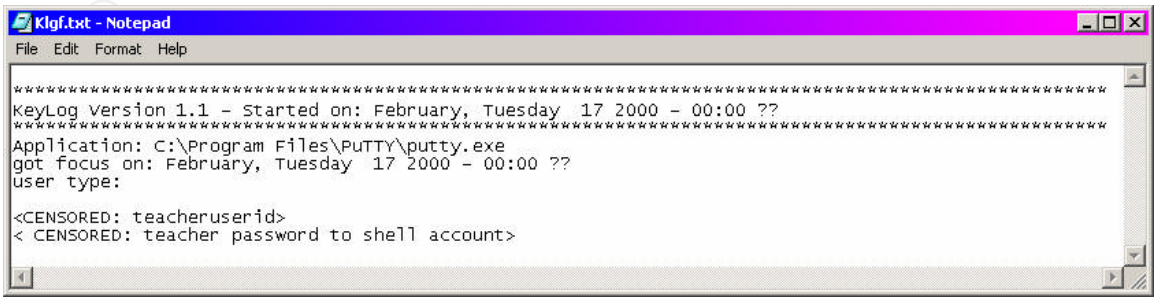



Figure 9: This shows what an entry in the log file for KeyLog V1.1 looks like.

Exploiting the System Part Two: The 'Ptrace-kmod.c' Exploit

Since I now have a valid login to the Linux server as someone other than me, I should try my next idea. If this works, I'll have root access. The steps are shown below:

- Using PuTTY, I login to the Linux server with the teacher's userid, and run 'lynx http://www.securityfocus.com/bid/7112/exploits/' to begin a text-based web session where I can download an exploit which will give me root access privileges; if it works. I'm preparing to downloading the file in 'Figure 10' below. To start the download I need to hit the letter 'd' and 'Enter" two times. The file is then saved in my current directory. To quit 'lynx' I hit 'q' then 'y.' Note that if I was blocked from performing this download by the school's firewall or filter, I could use other methods of obtaining it. Perhaps a site that the filter misses will have it.

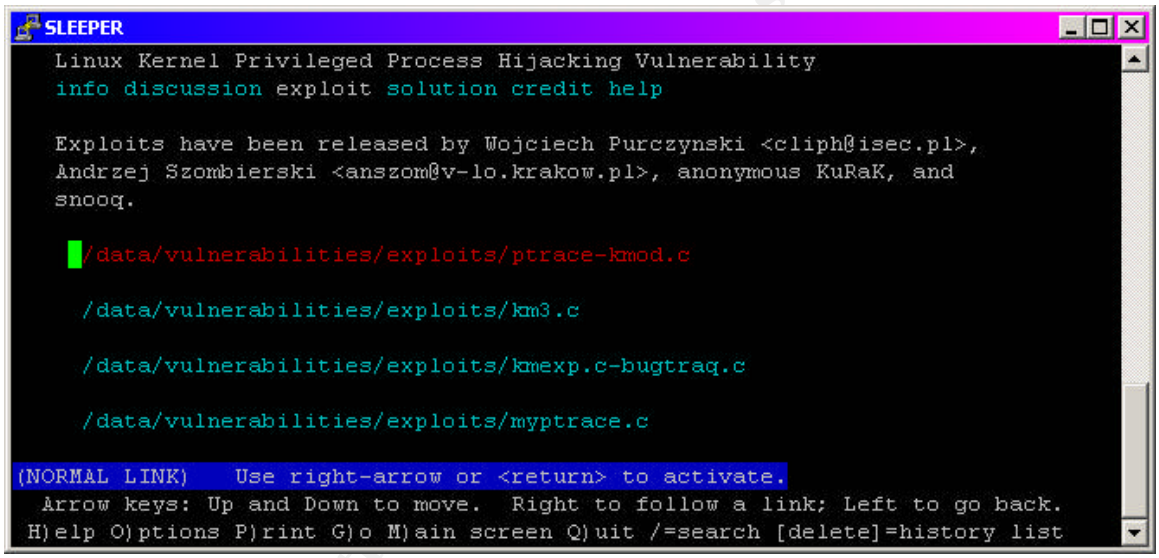



Figure 10: This shows me selecting a file to download; in this case, the C code for the ptrace exploit.

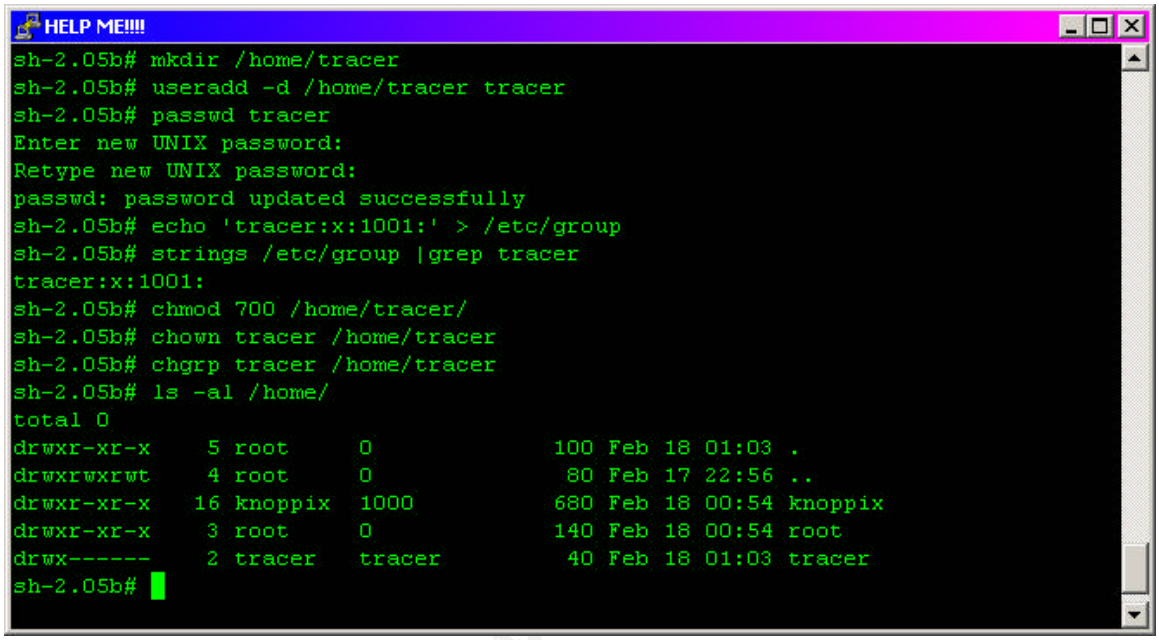
- Next, I compile the C code for the exploit using the command 'gcc –o ptracesploit ptrace-kmod.c.' This will create a program called 'ptracesploit' in my current directory.
- I test out the exploit by entering the command './ptracesploit' to execute it. Our kernel version is vulnerable! The 'whoami' command shows me as 'root'! 'Figure 5' has each of these steps shown.

Keeping Access

Now that I have root, I need to do something to keep this access level. It's too suspicious to keep signing on to the network using the teacher account. Also, if I use the computer help account, that could be easily discovered as well. People in charge tend to notice small things out of the ordinary like the last login message displayed when you successfully authenticate.

Keeping Access (continued)

That would be a bit suspicious; a login time which shows that your account was in use when you know it should not have been. I need to create myself a place to hide so I can do more when things cool off a bit.




'Figure 11': This shows the steps I go through to create a user account, home directory, and a group for myself.

After all the work I've done, I don't want to lose my access as root. Below is a breakdown of the commands used in 'Figure 11.' They are listed in order and I've included a brief explanation of the command line options used.

1. 'mkdir /home/tracer': This command makes a home directory for the user I'm going to create. Since '/home/' already exists, the directory 'tracer' will be created inside of it.
2. 'useradd –d /home/tracer tracer': 'Useradd' adds a user named 'tracer' to the machine and sets the user's home directory path to '/home/tracer.'
3. 'passwd tracer' is entered so I can set the password for my user. This will create an entry in the '/etc/passwd' and '/etc/shadow' if password shadowing is enabled.
4. 'echo 'tracer:x:1001:' > /etc/group': The echo command will print out text to the screen (known as stdout), or it can be used to put it's output elsewhere. The output is what is in quotes. If we simply typed in 'echo 'tracer:x:1001:'' the same thing would be printed on stdout (the screen). You can also see the '>' character in the above example. This symbol tells other commands to redirect the output they produce somewhere else.

Keeping Access (continued)

For the example above, we have redirected the group information for the new user into the file '/etc/group'. This file contains information on the groups used in the system.

5. 'strings /etc/group |grep tracer': This command has three parts. The 'strings' command can be used to search inside of files and display any text that it finds in four character lengths to stdout. The first part of the command, 'strings /etc/group,' would normally print out just the contents of the file '/etc/group/'. We have also used the '|' or pipe symbol in this command. The '|' can be used to string or chain together different commands. In this example, we have taken the output from the 'strings /etc/group' command and piped it to the next command, 'grep tracer'. The 'grep' command is used to search one or more files at a time for a specified pattern. In our example we examined the output of 'strings /etc/group' for any patterns matching the text 'tracer'. Why did we use such a complicated command? This command was used to verify that we did in fact create a group entry for the new user.
6. 'chmod 700 /home/tracer': We would like to have good file permissions for our user directory. We don't want just anyone being able to look around and run things inside there. With this in mind we change the file permissions using the 'chmod' command. There are two different ways to tell 'chmod' which permissions to use. I prefer using the octal method, which is depicted in the example in 'Figure 11'. For a more detailed explanation of file permissions on Linux, see http://www.linuxbeginner.org/modules.php?name=News&file=article&sid=43. We changed out file permissions so only the user I made can read, write, and execute files inside of '/home/tracer'. Of course root also will have this ability no matter what.
7. 'chown tracer /home/tracer': 'Chown' is used to change owner of a file or folder to the specified value. Here it is used to change the ownership of the folder '/home/tracer' to the user 'tracer'.
8. 'chgrp tracer /home/tracer': This will change the group ownership for files and folders. It is very similar to the 'chown' command except it works with groups.
9. 'ls –al /home/': The 'ls' command will send a listing of files in the current directory to stdout. If the '-al' option is used in conjuntion with 'ls', the output will display hidden files (files or folders with a '.' preceding the filename), file permissions, and the date and time in which they were last modified. We ran this command to check the changes we made to the user and group ownership for the user's home directory.

Covering the Tracks

Now I have a nice place where I can hide things, and an account I can use that is not traceable to my network user account. Since the system has a huge number of users, my home directory has a slim chance of being noticed. This will make it safer for me to keep my stuff on the server so I don't have to worry about uploading that script if I need it in the future. Now, I need to try to hide the script and the compiled exploit. I may need it later.

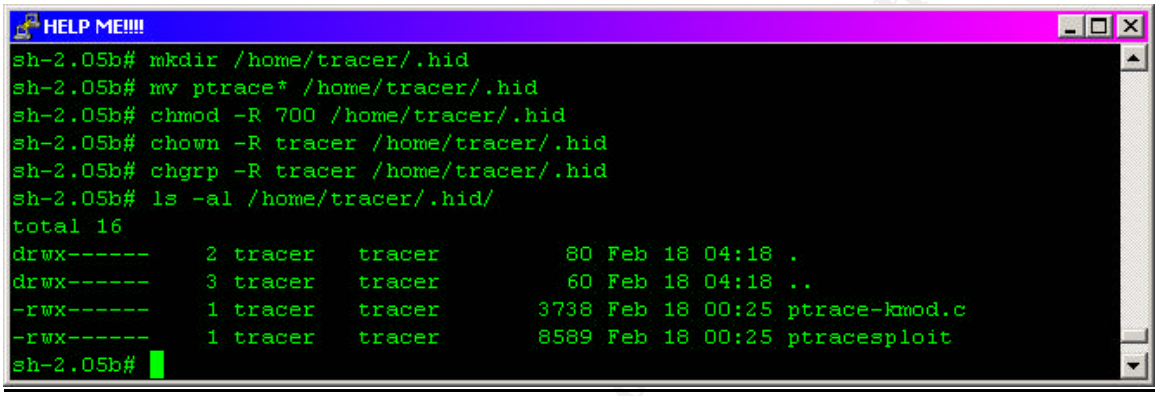



Figure 12: This shows the commands used to clean up some of the obvious traces of mischievous activities.

Some of the commands used above should look familiar, as we used five of them when we made the user account and home directory in the previous stage of the attack. The only command that is different in the above example is the 'mv' command. This command is used to move files and folders from one location to another. Since the files I want to move are located in the teacher's home directory, I need to get rid of them. 'Mv' will do that for me. By entering in 'mv ptrace* /home/tracer/.hid' I tell the computer to move all files beginning with the characters 'ptrace' and ending in anything else (this is what the '*' is for) to the directory 'home/tracer/.hid'. Also, the '-R' in 'chmod', 'chown', 'and 'chgrp' is a parameter that tells either of the three commands to do their work recursively; i.e. the folders as well as the files inside of them will have their respective attributes changed.

Now that we have hidden the evidence of our intrusion from the teacher, we need to try and get rid of any log file entries that can reveal our unauthorized account. There are five files that we should focus on for this stage of the attack. They are '/var/log/lastlog', '/var/log/wtmp', '/var/run/utmp', '/var/log/messages', and the teacher's '.bash_history' file. The files 'lastlog', 'wtmp', and 'utmp' are used to record login information, time and date changes, reboots, and information regarding what host a user logs on from. The '/var/log/messages' file is where general system event and error logging is done. The '.bash_history' file is a text file that contains a listing of commands executed by a user. It lives in the user's home directory.

Covering the Tracks (continued)

To cover up for my illicit activities, I'll need to use my new account. things up so none of the commands I enter will be put in the teacher's '.bash_history' file when I log out of their account. We will also check '/var/log/messages' to see if there is anything there we need to clean up. The process will be shown in the following series of screenshots beginning with 'Figure 13'. An explanation of the commands will be presented afterwards.

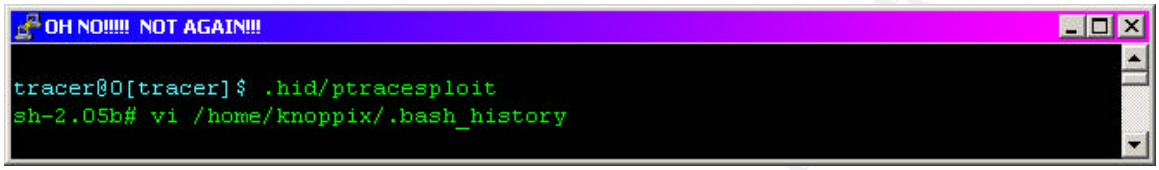



Figure 13:This shows the exploit being run again. We can also see that the compromised accounts' '.bash_history' file is about to be opened for editing.

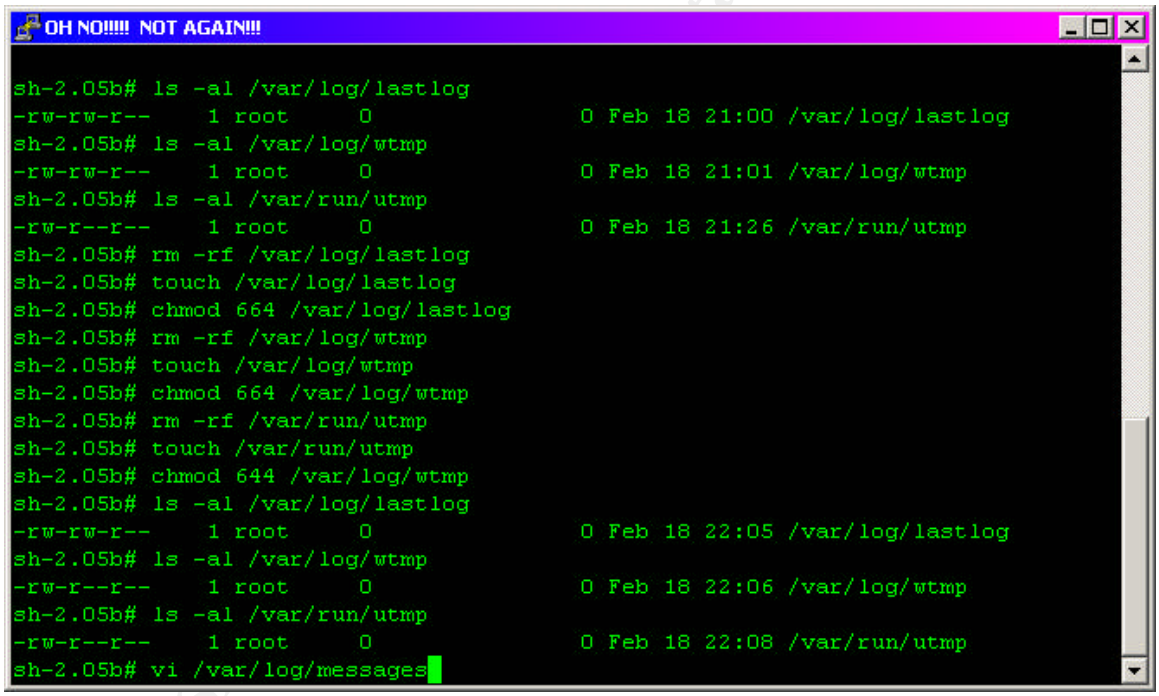



Figure 1: This shows the series of commands needed to clean up the log files that keep track of user login activity, and general system messages.

The steps taken to clean up the log files on the machine are presented below in the order they were executed. Each command, with the exception of those previously used, is also briefly explained. Steps 1 and 2 use 'Figure 13'. Steps 3 to 10 use 'Figure 14'. Step 11 uses 'Figure 15' on the next page.

Covering the Tracks (continued)

1. '.hid/ptracesploit': Since we can't do the things we need as a regular user on the system, we will need the help of the exploit. This is the syntax used to execute it from its location inside of the hidden directory we created earlier.
2. 'vi /home/knoppix/.bash_history': The 'vi' editor is a very versatile text editor found on many UNIX and Linux based systems. Here, we use 'vi' to open the teacher's command history file so we can delete any traces of our unauthorized usage of their account. Any of the commands we executed from this account are not visible now. More information on 'vi' can be found at http://unixhelp.ed.ac.uk/vi/.
3. 'ls –al /var/log/lastlog': Here, we are using 'ls' to check the permissions of 'lastlog'.
4. 'ls –al /var/log/wtmp': 'ls is used again to check the permissions of the 'wtmp file.
5. 'ls –al /var/run/utmp': The permissions of 'utmp' are examined.
6. 'rm –rf /var/log/lastlog': The 'rm' command is used to delete files or folders. The '-rf' switch tells it to delete them recursively without verification. You will not be asked if it is ok to delete the file if this switch is used. **USE 'rm –rf' WITH EXTREME CAUTION!!!** Think of it as an atomic weapon…
7. 'touch /var/log/lastlog': The 'touch' command is used to create a file. In this example, we recreated the file 'lastlog' in the '/var/log' directory.
8. 'chmod 664 /var/log/lastlog': Here we change the permissions of the newly created file to what they were originally. This is done so we don't break any logging features or arouse to much suspiscion. **Note:** Steps 6, 7, and 8 are repeated for the files'/var/log/wtmp', and '/var/run/utmp', and '/var/run/utmp' consecutively.
9. A final check is made on the files we touched using 'ls –al'.
10. 'vi /var/log/messges': Here again we use 'vi' to see if there are any weird messages created by our activities.
11. ''rm –rf .hid/ptracesploit': Since files in user areas that are marked as setuid root are a bit suspicious, we delete the exploit itself. The code is retained so we can make the program again if need be.

"Now it's time to log out, lay low for awhile and figure out, what I want to do next… I can look up more things to do at home."


Figure 15: This shows the deletion of the exploit's binary, and the shell spawned from the exploit is exited.

A Summary of the Attack

Let's recap the events leading up to the attack, and the attack itself. First, our attacker was given three login ids for use on the school network. One for the Windows workstations and Novell file servers, a login to the Linux server, and a login for the school website. Using these logins the attacker was able to do a bit of recon and determine what kinds of servers were in use at the school. The attacker also noticed that the userid and passwords given to them were the same for each network resource that they needed access to. Therefore, if one person's userid and password was captured somehow, it could be that the same userid and password would work for multiple network resources. This would allow our attacker to login to the Novell servers, The Linux server, the school website, and possibly the email aacount for the staff member or student whose login credentials were captured.

Next, the attacker installs keylogging software in some strategic places and they are able to sucessfully capture a teacher's login (among many other things). This serves to give the attacker a different account that they can be used to conduct malicious activities from. The trail would be blurred a bit, as it would appear that the teacher was doing something fishy if process accounting captured any sort of annomolous activity.

Now that a less incriminating login was captured, the attacker can try different methods of exploitation in order to elevate the privilidges they have with the different network resources and servers they have access to. Since a few tricks were known by the attacker regarding such things in Linux as opposed to Novell, a locally exploitable privilidge escalation attack for Linux was chosen. After compiling the exploit's code using 'gcc', the attacker ran the script to see if it would work. The program worked so the machine did indeed have a problem, and our attacker now had 'root' access.

Using this level of access, they were able to create another user account for themselves, move the exploit code and program out of the teacher's home directory, and put them each in a hidden directory in the new account's home directory. The teacher would never see anything that would arouse suspicion.

After authenticating under the newly made account, the attacker executes the script again and cleans up the obvious traces of their presence and activites. The incriminating log files were removed/altered, the exploit code and program were moved, and the exploit was deleted from the system. Those setuid binaries are a bit suspicious. The attacker left the code in hidden directory so the same exploit could be compiled and run again if need be.

From this point, thanks to having root access, the attacker can pretty much do whatever they want to the system. Kernel-level rootkits could be installed, network traffic sniffers could be setup and run, all manner and kind of information could potentially be compromised and used for the personal gain if the attaker or others… Many malicious misuses of the network are now possible.

## The Incident Handling Process

Every network environment is different, so the incident handling process will have some differences as a result. Our mock educational institution would not have the same incident handling policies and procedures as a military network for example. However, there are six general guidelines that should be adhered to when working with any network security incident. The steps that would be taken to handle this type incident will expounded upon below.

## Preparation

Being prepared. There is a saying regarding preparation known as the 'Seven P's' in some circles. Proper prior planning prevents piss poor performance. This adage, while a bit base, is true and should be applied to the networks in use by those in homes, educational institutions, military facillities, and all other types of network environments. It is really a matter of time before someone does something that will put your network, with its various types of services and information, to the test. Your team needs to be preprared, and they should periodically take a bit of time to test the system themselves. Team members may want to create incidents for practice using machines attached to a private, completely isolated network with no internet access.

Our mock educational institution has several ways in which it has attempted to prepare for a network security incident. It has installed, configured, and utilized some specific systems that could help with the detection of an incident. The Cisco PIX firewall, the N2H2 filtering, and the NIDS (Network Intrusion Detection System) are each helpfull in the prevention and detection of incidents. The NIDS is another system running Linux. It has Snort (http://www.snort.org) and SnortSnarf (http://www.silicondefense.com/software/snortsnarf) to assist in deciphering the alerts.

Printed matierial regarding appropriate usage of the network is distributed to all staff and students in the school via the staff and student handbooks. The students and staff are instructed that the computers are the property of the educational institution and are to be treated as such. This material specifically defines what is not considered acceptable use, and warns of the possible consequences for such actions. In addition, the document also warns the user that all of their activities on the network may be remotely observed, monitored, and logged. Furthermore, it is also clearly states that the involvement of law enforcement, legal action, termination, and or expulsion could all be consequences for inappropriate usage. Special desktop themes have been set for the different types of network users so staff will hopefully be able to identify what type of account is logged in on a computer at a given moment. Warning banners have also been posted in different areas of the building, and scripts that display the same text have

Preparation (continued)

been implemented so as a user logs on to the network, they will see the message displayed on their computer screen. A network orientation process is under development for new employees and students that will revise the currently existing orientation proceedure. A specific incident handling plan for our mock network is currently under development.

In keeping with the plan that is currently being devloped, there are seven parties who would be called upon if it were percieved that there may be a network security event occuring. Three individuals would be the top-ranking system administrators for the computers on the mock network; one would be the school network security officer; the system administrators' and network security officer's supervisor would be included, as would would the school's dean or principal. The educational facillity's central authority should also be involved.

Each of the parites on the team was chosen because of their particular skillset, and their administrative spheres of influence. The system administrators have combined experience running Linux, Windows, and Novell. In addition, they are pretty much the front line in our test environment so it is important to have them on the team. The network security officer has training and experience with various network security devices and applications, computer forensics procedures (chain of custody, usage of forensics tools), and has had experience dealing with network security incidents in a manner that can stand up in a court of law. They can help the other team members in the search and seizure process. The supervisor for the system administrators and network security officer will be familiar with each of them and has a need to know what kinds of issues are happening on the network. They also have the managerial skills needed (we hope) to effectively lead and coordiante the activites of the sys admins and network security officer. The dean or principal also needs to be kept in the loop as they will be able to help guide the other individuals from the school, and they will know who to communicate with outside of the school should a university system or school district administrative facillity need to be involved. If an incident occurred, the hierarchical order of notification and involvement would be:

a) system administrators and or school network security officer
b) system administrators'/network security officer's supervisor
c) the school's dean or principal
d) the school district office
e) if needed, law enforcement would be involved

To facillitate communication, all of the team members should commmunicate via cellular telephone, or if no cellular service is availible, a landline must be used. Network communication of any sort should be discouraged since it is sniffable, and if an attacker sees anything suspicious they may do something drastic (cd /, rm –rf anyone?) or stop the attack and

Preparation (continued)

get away. A contact information tree for each of the individuals on the team should be made and kept current. This information should be given to each team member. The information should be kept in a secure location, but it should also be easily accessible to team members.

The incident response team will also need to have access to a secure room that can be used for meeting to discuss various issues or problems regarding an incident. This room should also have a very secure area where chain of custody logs and any pertinent physical or electronic evidence can be kept under lock and key. According to http://www.first.org/events/progconf/2002/sup-02-brown-paper.pdf, page 5,

> 'Create a Chain-of-Custody form and manage it vigilantly. The idea behind Chain-of-Custody is simple. Keep track of anyone who has accessed the evidence from this point forward.
> Your Chain-of-Custody should include:
> 1. Who Has physical possession
> 2. Why They have physical possession
> 3. Where They have physical possession
> 4. Any Comments
> 5. Signature
>
> If the evidence is changing possession then the releasing person and the gaining person should sign the document.'

The chain of custody documentation should also have the time and date of each event logged within it. If the chain of custody proceedure is not followed to a tee, the entire incident and all of the evidence gathered could be deemed as inadmissable by legal authorities. Any time backup images, hardware, log files intended to be used as evidence, or any other matierial supporting the incident is touched. It must be put in the chain of custody log. Be careful!!!

Identification

If this were an actual incident, it would be difficult to detect in many environments. Early detection would be crucial. The tools used on the attacked platforms are not particularly invasive, nor is the impact to systems they are deployed on significant enough to attract a great deal of attention. The early warning signs for an attack like this would be subtle. Indicators for this scenario could possibly include:

- A individual notifies the network support or help desk team and reports that they had a 'last login time' that was atypical for their particular usage pattern.
- Staff or students may report seeing someone who appeared to be using one computer one moment, and then using another a moment later.
- Staff or students may report seeing an individual logging in as a network administrator. The desktop themes can help identify that.

Identification (contnued)

- Error messages may appear on machines that had keylogging software installed when the log file fills the hard drive, or if something else with the keylogging program is broken.

If any of these types of issues are reported to systems administrators or help desks, they would be wise to follow up immediately. They may be experiening the beginnings of a system compromise. Depending upon how the exploit is installed or downloaded on a system, portions of the process could be logged by the NIDS in our facillity, or the compromised system may have some evidence in its log files assuming they are examined in enough time. Here again, time is critical.

In the later stages of the attack, or after the compromise has occurred, we might see some different signs. If registry keys, processes, or files matching the description given in figures 2 and 4; or any files found match up with those in the tables on page 7 you probably have a keylogger on the system and you should be worried. If you see any of the files or processes depicted in figures 6 and 7 you may be dealing with the ptrace local root exploit. Note that the filenames for this exploit could be anything, but the checksums and code used to compile it should not vary.

Concerning the NIDS; if it has been set up properly, and appropriate rulesets have been implemented to catch sites missed by the filtering application, you should be able to see log files of the exploit code being downloaded if it was done over a cleartext protocol such as ftp, http, or tftp. If scp, https, or another encrypted protocol is used the activity will not be logged.

If some of the clues are discovered, the decision to perform a backup or take an image for forensic analysis may be made. A few copies of he original image should be made, and the original along with one backup copy should be locked up in the team's secure room inside the chain of custody locker. The logs for any work done with the evidence should be filled out as well.

Our network security officer can now has an image they can analyze using forensics utilities. This can reveal the hidden details of the exploit such as deleted, moved, or modified files; a filesystem activity timeline, compiled exploit code; rogue account creation; rootkits, etc. For more information regarding such utilities, see http://www.sleuthkit.org/.

Containment

Since our fake network security incident handling team has ESP, they knew there was going to be a problem and the network security officer whipped out his team's jump kit and got to work on the problem. Jump kit? What is a jump kit? A jump kit is a pre-assembled kit (as in well before an incident occurs) that has the tools and utilities needed to perform forensic analysis,

Containment (continued)

backups, or anything else that the team finds will be useful in the event of a network security incident. A list of the items in our team's jump kit can be found below:

a) The Peguin Sleuth Kit for performing forensics analysis
b) Knoppix STD Version 1 for performing analysis
c) A FAST computer system with enough hard disk space to store copies of images on for analysis purposes. This machine is kept in top working order. It should also has a DVDRW/CDRW drive.
d) The NSRL from NIST (http://www.nsrl.nist.gov/subscription.htm)
e) An anti-static floor mat and wrist straps
f) More chain of custody logging forms
g) Writing utensils
h) The team contact information tree
i) Norton Ghost
j) Legal pads for taking notes

As permission to use a live machine to run this demo on was refused, log excerpts and screen shots from Autopsy could not be obtained. Furthermore, the author was not able to obtain an image of the Knoppix bootable Linux CD's operating system as 'dd' could not take an image from '/ramdisk/' where the files modified during this demo are located. This may be because it is a temporary filling system, and exists in memory. For an example of a filesystem timeline activity log generated by Autopsy see http://www.giac.org/practical/GCIH/Heather_Larrieu_GCIH.pdf, pages 42-53. For examples of using 'dd' to take images for forensic use, see http://www.crazytrain.com/dd.html.

Again, since our team was somewhat prepared, they were able to run Autopsy on an image they took of the system in question. The file system time analysis revealed changes to the system files mentioned in the section of this document entitled 'Exploiting the System Part Two: The 'Ptrace-kmod.c' Exploit.' They were able to see most of the things done by the attacker. While it seemed as if the teacher had carried out the attack, the team was able to determine otherwise. The teacher used their Linux account rarely, and normally only to post items or the classes they teach on their section of the school website. The attack had to come from elsewhere.

It was also observed that the root (the attacker) had modified /var/log/lastlog, /var/log/wtmp, and /var/run/utmp. Also, the exploit files were moved out of the teacher's home directory, and into a hidden directory in a newly created account called tracer. The account did not follow the naming scheme used in the school, and did not have any of the normal files that average accounts on their system normally have. In addition, tracer forgot to try and get rid of their .bash_history file. In there our team ound the commands used to make the account, modify the aforementioned files in /var,

Containment (continued)

and the command which was used to delete the ptrace exploit binary was discovered. Oooops… ☺

After a bit of discussion and debate, it was decided that the system would not be taken out of service. The tracer account would be monitored or the rest of the day, and at the end of the day it would be deleted from the system. Also, all user accounts on all servers were locked that evening. The team suspected that somehow another users' authentication information had been obtained by the attacker, and that they could possibly have multiple userids and passwords in their possesion. Using some automated workstation file scanning utilites (Novell Zenworks for Desktops), any of the machines the teacher may have used were checked over for keyloggers or sniffers. The team worked throughout the night to accomplish this. They also set up automated scripts to search the fileservers of files that were considered contraban in an attempt to find any other evidence. Nothing else was found.

Eradication

The exploit code found was analyzed and it was discovered that it was a local root exploit which many Linux systems were vulnerable to. A patched kernel had been released that would prevent the ptrace exploit from being able to do it's nasty job. This kernel was downloaded and installed. In addition, any other patches the linux system needed were applied. Investigation was begun on ways to prevent kernel root exploits as well. The main software being invesitgated for this purpose was the grsecurity kernel patch at http://www.grsecurity.net/features.php.

Since the team suspected a keylogger, some investigation was done about how to discover and prevent such things from remaining hidden. While some are detectable by antivirus software, others are not. Among the options investigated were Privacy Keyboard for Workstations (http://www.anti-keyloggers.com/ ), and some freeware solutions such as Spybot (http://www.safer-networking.org/) were also implemented until a more thourough evaluation of the products could be completed.

Recovery

Since there was no data loss, or severe damage to the servers or services running on them, a complete system restore was not needed. Instead, they team was able to patch the system that was compromised, and over the next few weeks every user was required to change their password or their account would remain turned off.

Lessons Learned

What did the team learn from this experience that could help them be better prepared? See below:

- Keep your virus definitions, operating systems, and software applications up to date!
- Pay attention to those little unexplainable things that happen on your network. They could point to a bigger issue.
- Implement filtering of web content, and a NIDS.
- Practice network security incident response with your team. Practice makes perfect (or efficient if not perfect).
- Keep up to date with the different security solutions availible to you.
- Definitely implement warning banners, and make sure that the networks acceptable use policies are distributed and enforced.
- Last but not least, do develop a security incident handling team and practice, practice, practice.

Works Cited

Knoppix. KNOPPIX – Live Linux Filesystem On CD.
<http://www.knopper.net/knoppix/index-en.html>.

Tatham, Simon. PuTTY: A Free Telnet/SSH Cleint. 12 February 2004.
<http://www.chiark.greenend.org.uk/~sgtatham/putty/>.

Academic Computing Services. Lynx Information.
<http://lynx.browser.org/>.

KEYKatcher, Inc. KEYKatcher: The Easiest way to monitor your PC!.
< http://www.keykatcher.com/>.

AMECISCO, Inc. KeyLogger. 15 February 2004.
<http://www.amecisco.com>.

Mikko Technology. KeyKey 2000 Professional. 26 March 2003.
< http://www.mikkotech.com/keykey.html>.

StAllIOns, Red0xd. Skl0g Keylogger: stAllIOnized KeyLogger.
<http://skl0g.cjb.net/>.

iSEC, Inc. ISEC Security Research :: News. 18 February 2004.
<http://www.isec.pl/news.html>.

U.S. Department of Homeland Security. Common Vulnerabilities and Exposures: CAN-2003-0127. 13 March 2003.
<http://cve.mitre.org/cgi bin/cvename.cgi?name=CAN-2003-0127>.

SecurityFocus. Linux Kernel Privileged Process Hijacking Vulnerability. 15 January 2004.
<http://www.securityfocus.com/bid/7112>.

Works Cited

Packet Storm. OpenFuck.c: Remote exploit for Apache + OpenSSL v0.9.6d. 14 March 2003.

<http://packetstormsecurity.nl/0303-exploits/OpenFuck.c>.

die.net. ptrace(2) – Linux man page.

<http://www.die.net/doc/linux/man/man2/ptrace.2.html>.

FreeBSD.org. "3.7: Race Conditions." FreeBSD Developer's Handbook.

<http://www.freebsd.org/doc/en_US.ISO8859-1/books/developershandbook/secure-race-conditions.html>.

Beyond Security. Ptrace Exploit Code Released. 23 March, 2003.

<http://www.securiteam.com/exploits/5CP0Q0U9FY.html>.

Spade, Samuel. SamSpade.org.

<http://www.samspade.org/>.

Internet Free Expression Alliance, Childrens Internet Protection Act (Pub. L. 106-554)

<http://www.ifea.net/cipa.html>

Gotroot. Understanding Linux Permissions. April 25 @ 21:12:50 EST

<http://www.linuxbeginner.org/modules.php?name=News&file=article&sid=43>

University of Edinburgh. Using the vi editor.

<http://unixhelp.ed.ac.uk/vi/>

Caswell, Brian and Roesch, Marty Copyright 2002, 2003, 2004

<http://www.snort.org>

Works Cited

Silicon Defense, Copyright 2003
<http://www.silicondefense.com/software/snortsnarf>

Brown, Christopher: Procedural Aspects of Obtaining Computer Evidence with Highlights from the DoJ Search & Seizure Manual. page 5
<http://www.first.org/events/progconf/2002/sup-02-brown-paper.pdf>

Carrier, Brian. The Sleuth Kit (software package)
<http://www.sleuthkit.org/>

National Institute of Standards and Technology; 8/20/2003
<http://www.nsrl.nist.gov/subscription.htm>

Larrieu, Heather; GCIH Version 2.1a Practical Assignment October 7, 2003. pages 42-53
<http://www.giac.org/practical/GCIH/Heather_Larrieu_GCIH.pdf>

Knoppix. KNOPPIX - Live Linux Filesystem On CD.
<http://www.knopper.net/knoppix/index-en.html>.

Tatham, Simon. PuTTY: A Free Telnet/SSH Cleint. 12 February 2004.
<http://www.chiark.greenend.org.uk/~sgtatham/putty/>.

Academic Computing Services. Lynx Information.
<http://lynx.browser.org/>.

Rude, Thomas; DD and Computer Forensics; August 2000
<http://www.crazytrain.com/dd.html>

Works Cited

dev@grsecurity.net
<http://www.grsecurity.net/features.php>

Raytown Corporation LLC, Copyright © 1998-2003
<http://www.anti-keyloggers.com/>

Kolla, Patrick M. Copyright 2000-2004
<http://www.safer-networking.org/>