Global Information Assurance Certification Paper

GIAC Certified Intrusion Analyst Practical Assignment

Donald Parker
GIAC GCIA Practical Version 3.3
Submission Date:

Table of Contents

Hping and Packet Crafting
Friend or Foe?

Abstract

This paper will demonstrate the value of packet crafting as both an instructional tool, and an invaluable resource to help secure one's network assets. Through the use of crafted packets and the knowledge required to effectively craft them one will by default learn the tcp/ip stack as well as it's various responses to certain stimulus. This is how it serves as an instructional tool. Also by crafting custom packets yourself one will be able to definitively confirm certain network conditions. The crafting of packets complements the use of other security programs such as Nessus[1] by allowing you a greater flexibility in what you want to create at the packet level, and whatever payload you wish to have the packet carry. All of these concepts will be clearly demonstrated in the following pages with examples of command syntax as well as tcpdump[2] snippets showing the crafted packet as it appears on the wire.

Introduction

There is an obvious requirement in today's networked world to have solid defenses in place. Most of today's tools being used by the network security professional are of the automated variety ie: Nessus, Saint, amongst others. These tools are excellent tools in and of themselves. They do not however allow you the flexibility required in today's mixed threat environment to fully test your network. The use crafted packets will also allow you to test your Intrusion Detection System ruleset, and confirm the systems functionality. There are some examples shown of how the Snort IDS reacts to certain stimulus later in this paper. Also this is the day of buffer overflow's[3], format string exploits, and a plethora of CGI exploits. While packet crafting will not fix many of these exploits it will allow you to confirm if you are vulnerable to them. This will only work on systems that are vulnerable to ISN prediction though. Systems such as Windows 98 for example. Otherwise the replication of such attacks using crafted packets will fail due the necessity of the 3 way handshake. That is one will have to effectively guess the tcp sequence number increments and apply them to the following push ack packets with the embedded malicious payload.

Hping will allow you to manually inject them yourself thereby helping you to understand exactly how it is possible to pull of some of these attacks. It is imperative that if one is to properly defend then one must learn how to attack. This can be done through the judicious use of crafted packets carrying an embedded malicious payload which you fire at your network, and or specific machines. Another side benefit of doing so will be learning exactly how the exploit works. Is ISN prediction invloved? Is it simply a matter of firing push packets with a payload or is the 3 way handshake critical.

These questions require answers, which will in turn allow you to effectively craft the packets, and by extension as mentioned give you insight into how the exploit works. Another aspect that is highly under-rated in packet crafting are the educational uses of packet crafting. By having a junior security professional craft packets in a lab environment they are able to see the tcp/ip stack in action as it sends out various flags.

They will also be able to recreate common messages that they see as evidenced by the common "icmp host unreachable" error. The analyst will realize that this error is received when you send a packet to a host, which does not exist. As I just explained this can be invaluable in learning the inner working of tcp/ip or just reinforcing lessons learned. The packet crafting tool used for this paper is HPing.rc2[4] which was coded by Antirez. This tool is not available for Windows as it stands right now. It is however compilable in all modern Linux distributions, as well as Solaris 2.X onwards, and all BSD variants. You will also require libpcap to run the tool.

The Syn packet "Knock Knock"

The syn packet is the first step in the 3 way handshake of tcp/ip. It is akin to one putting your hand out to shake someone else's hand.
The syn packet is located in the 13th byte of the tcp header, and has the binary value of 2 ie: it is the second bit from the right in the 13th byte, and one counts in the following fashion as seen below in figure 1.

Fig 1

X	X	U	A	P	R	S	F
128	64	32	16	8	4	2	1

As seen by the above noted one counts from right to left, and in the above noted number scheme. The two X values at position 128 and 64 are assigned to ECN. For a good link on ECN (explicit congestion notification) please see http://www.icir.org/floyd/ecn.html The assigning of numerical values to the bits within a byte is important. One reason for doing so would be bit masking which is also a valuable skill for the network analyst. For an excellent tutorial on bit masking please see the following url at the below noted; http://security-forums.com/forum/viewtopic.php?t=4489

We will now show the command usage used when creating a syn packet using HPing and the ensuing output using a tcpdump snippet. Tcpdump as well can be found at the following url; http://www.tcpdump.org/ This program also requires the use of libpcap which can be found on the same page. Now with the appropriate programs in place and their associated dependencies we are ready to head to the forge to craft some packets!

The two packets you see below are just one ip addy sending a Syn packet to another ip addy. For brevities sake I am using ip addy to signify IP address. To do this using Hping is a very simple task. Just type in "exactly" the below noted command syntax show in Figure 2 and voila a syn packet is sent! The xxx.xxx.xxx.xxx noted in the below example is the destination ip addy that you are sending a syn packet to.

Fig 2

```
hping -S xxx.xxx.xxx.xxx
```

19:54:34.236334 src.xxx.xxx.xxx.1321 > dst.xxx.xxx.xxx.0: S
897957123:897957123(0) win 512
0x0000 4500 0028 6510 0000 4006 24bd xxxx xxxx E..(e...@.$..r.|
0x0010 xxxx xxxx 0529 0000 3585 bd03 67f0 e290 .r...)..5...g...
0x0020 5002 0200 7aac 0000 P...z...

19:54:36.227147 src.xxx.xxx.xxx.1323 > dst.xxx.xxx.xxx.0: S
576193543:576193543(0) win 512
0x0000 4500 0028 8711 0000 4006 02bc xxxx xxxx E..(....@....r.|
0x0010 xxxx xxxx 052b 0000 2258 0407 0738 40dd .r...+.."X...8@.
0x0020 5002 0200 4940 0000 P...I@..

Please note that the src and dst are used in the first octets to denote the source and destination addresses you see above. (would be where you insert the ip address)
The 0 noted above right before the S denotes the port that the syn packet is being sent to. The default port on Hping is 0, unless you specify otherwise it will always default to it for your outbound packets.

The syn packet is the first step in the TCP/IP hand shake. To open communications between two computers the very first step is to send a syn packet to the computer you wish to communicate with. This would be followed by the syn/ack, in turn followed by the ack. At this point you are ready for the setup of communications and then the exchange of data.

This is one of the most common scans out there today. The Syn scan is also noisy and easily detectable by properly configured Intrusion Detection Systems such as Snort, Netranger, amongst many others. It is a good scan to run however due to the conventions of TCP/IP ie: send a syn you expect a syn/ack back on an open port. As mentioned above when an open port with a service running upon it receives a syn packet it will respond with a syn/ack confirming it is open and ready for business. Think of port 25/SMTP, port 110/POP3 amongst others that would respond to this type of scan. Assuming that they are not hidden behind firewalls that is. Even if they are though the lack of a response is invaluable information in and of itself. More on that when we look at the rst packet.

Be aware that when you do not specify a destination port on the targeted computer it will default to 0. Also if you do not specify a source port it will use a random ephemeral port[6] and go up numerically from there. More on how to specify both src/dst ports later. On with the basics for now

The Reset packet, is it useful for scanning?

The below noted packet is a reset packet. The reset packet is used to reset a connection. As you can see the command syntax is very similar. The only change is in the actual switch itself. Instead of -S it is -R.

The rst packet is often used to perform what is known as inverse mapping. What this means is that rst packets are sent out and the response received is what will tell you if the host exists or not. If you send out a rst scan you would get one of two things. You will either get no response which will indicate to you that the host is probably alive or an ICMP host unreachable msg. This would indicate that the host does not exist. This is what is known as inverse mapping. Some IDS systems will not log rst packets/scans due to the sheer multitude of them. This is why the inverse scan is popular.

hping -R xxx.xxx.xxx.xxx

19:54:57.669980 src.xxx.xxx.xxx.1239 > dst.xxx.xxx.xxx.0: R
1975237774:1975237774(0) win 512
0x0000 4500 0028 890e 0000 4006 00bf xxxx xxxx E..(....@....r.|
0x0010 xxxx xxxx 04d7 0000 75bb bc8e 631c a4e4 .r......u...c...
0x0020 5004 0200 7dbb 0000 P...}..

19:54:58.666747 src.xxx.xxx.xxx.1240 > dst.xxx.xx.xxx.0: R
225427189:225427189(0) win 512
0x0000 4500 0028 1b76 0000 4006 6e57 xxxx xxxx E..(.v..@.nW.r.|
0x0010 xxxx xxxx 04d8 0000 0d6f bef5 458a 2a1c .r.......o..E.*.
0x0020 5004 0200 7bfa 0000 P...{...

The Fin packet, a scanning type perhaps?

This packet is a fin packet ie: to close a conx already established. Once again the command syntax is very similar to the above two examples.

The fin packet is also used to perform the fin scan. This type of scan will elicit different responses depending on the type of platform your running, be it Windows or Unix/Linux. When a Windows machine receives an unsolicited fin packet it will send back a reset packet whether or not that port is running a service. On a Unix/Linux machine a port running a service will just ignore unsolicited fin packets and send back no response.

hping -F xxx.xxx.xxx.xxx

19:55:24.992034 src.xxx.xxx.xxx.1502 > dst.xxx.xxx.xxx.0: F
1235235694:1235235694(0) win 512
0x0000 4500 0028 991c 0000 4006 f0b0 xxxx xxxx E..(....@....r.|
0x0010 xxxx xxxx 05de 0000 49a0 336e 3917 cbd5 .r......I.3n9...
0x0020 5001 0200 3507 0000 P...5...

19:55:25.986754 src.xxx.xxx.xxx.1503 > dst.xxx.xxx.xxx.0: F
1775365876:1775365876(0) win 512
0x0000 4500 0028 ccb0 0000 4006 bd1c xxxx xxxx E..(....@....r.|
0x0010 xxxx xxxx 05df 0000 69d1 eef4 00c1 98f3 . r......i.....
0x0020 5001 0200 c486 0000 P.....

Crafting the ICMP protocol

The below noted packet is an icmp echo request ie: ping. This packet is useful to determine whether or not a specific host is up or not. The command syntax is a little different for this packet. We will specify after hping that we want a icmp packet by putting in the numerical value 1 followed by the ip addy of the host we are pinging. There are a great many uses for icmp as well as various types of icmp. Please see the following url for further information on icmp http://www.faqs.org/rfcs/rfc792.html This protocol is often used for the ubiquitous ping scan as mentioned above. One reason to use this protocol is to see if specific hosts are alive.

```
hping -1 xxx.xxx.xxx.xxx

19:55:46.914365 src.xxx.xxx.xxx > dst.xxx.xxx.xxx: icmp: echo request
0x0000   4500 001c 20cf 0000 4001 690f xxxx xxxx       E.....@.i..r.|
0x0010   xxxx xxxx 0800 13e6 e419 0000                 .r.........

19:55:47.906748 src.xxx.xxx.xxx > dst.xxx.xxx.xxx: icmp: echo request
0x0000   4500 001c ee83 0000 4001 9b5a xxxx xxxx       E.....@..Z.r.|
0x0010   xxxx xxxx 0800 12e6 e419 0100                 .r.........
```

Crafting the UDP protocol

The packet below is simply a udp packet. To send one is rather easy as well. We will have to tell hping that we want a udp packet by putting in the numerical value 2. The default protocol for hping is tcp. This is why of course we need to tell hping what protocol we wish to send by putting in the numerical value of 2. This is of use when probing services which are udp based vice tcp. Such as netbios, nfs, dns, nis amongst others.

```
hping -2 xxx.xxx.xxx.xxx

19:56:39.753975 src.xxx.xxx.xxx.2462 > dst.xxx.xxx.xxx.0:  udp 0
0x0000   4500 001c c394 0000 4011 c639 xxxx xxxx       E.....@..9.r.|
0x0010   xxxx xxxx 099e 0000 0008 053d                 .r.........=

19:56:40.746747 src.xxx.xxx.xxx.2463 > dst.xxx.xxx.xxx.0:  udp 0
0x0000   4500 001c bdc2 0000 4011 cc0b xxxx xxxx       E.....@....r.|
0x0010   xxxx xxxx 099f 0000 0008 053c                 .r.........<
```

Creating more complex packets

The below noted packet is a syn packet directed at port 21 aka ftp. To send a syn packet at a specific port requires a few more switches. This is where the usage of hping begins to shine. As noted below we are sending a syn (-S) packet to xxx.xxx.xxx.xxx specifically on their ftp port by putting in the (-p) switch. To specify the destination port you put in

the -p. To specify the source port on your machine you want the packet to go out on you would use the -s switch followed by a port number just as the destination port example below.

hping -S xxx.xxx.xxx.xxx -p 21

```
19:57:01.789384 src.xxx.xxx.xxx.1548 > dst.xxx.xxx.xxx.21: S
1371884204:1371884204(0) win 512
0x0000   4500 0028 0661 0000 4006 836c xxxx xxxx      E..(.a..@..l.r.|
0x0010   xxxx xxxx 060c 0015 51c5 4aac 669b 5b07      .r.....Q.J.f.[.
0x0020   5002 0200 58aa 0000                          P...X...

19:57:02.786747 src.xxx.xxx.xxx.1549 > dst.xxx.xxx.xxx.21: S
1979208427:1979208427(0) win 512
0x0000   4500 0028 d63a 0000 4006 b392 xxxx xxxx      E..(.:..@....r.|
0x0010   xxxx xxxx 060d 0015 75f8 52eb 364e 01c6      .r......u.R.6N..
0x0020   5002 0200 b5c5 0000                          P.......
```

More complex packet crafting with a payload inserted

The below noted is a push packet directed at a specific port. In this case http port 80. The "payload" in the push packet should be done up ahead of time in a file that you will specify in the command string. You will as well have to make sure that the packet length is long enough to handle your payload. Hence another switch. I will go over and explain each switch one by one for this type of packet.

-P Tells hping to send a push packet
24.114.xxx.xxx This is the destination ip
-d Allows you +/- the size of the packet itself in this case we have set it to 80 bytes
-p Specifies the destination port in this case port 80
-E Tells hping where to look for a file which it is to insert as a payload ie: /home/don/test.sig Quite useful obviously for pre-compiled exploits ie: buffer overuns

hping -P xxx.xxx.xxx.xxx -d 80 -p 80 -E /home/don/test.sig

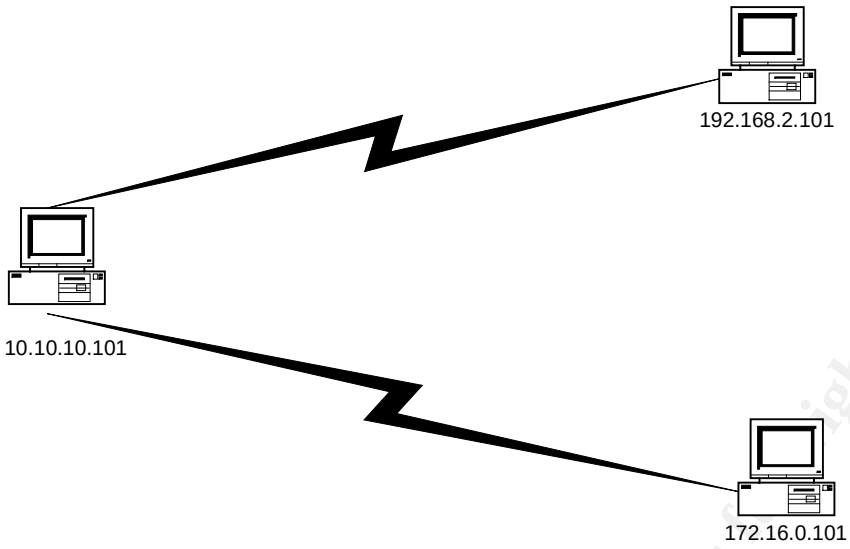
```
19:58:25.721579 src.xxx.xxx.xxx.2426 > dst.xxx.xxx.xxx.80: P
729845249:729845329(80) win 512
0x0000   4500 0078 4a7b 0000 4006 3f02 xxxx xxxx      E..xJ{..@.?..r.|
0x0010   xxxx xxxx 097a 0050 2b80 8e01 5ce6 ac80      .r...z.P+...\...
0x0020   5008 0200 4178 0000 4745 5420 632b 6469      P...Ax..GET.c+di
0x0030   720a 4745 5420 432b 4449 520a 2f2f 6874      r.GET.C+DIR.//ht
0x0040   7470 2031 2e30 0a00 0000 0000 0000 0000      tp.1.0.........
0x0050   0000 0000 0000 0000 0000 0000 0000 0000      ................
0x0060   0000 0000 0000 0000 0000 0000 0000 0000      ................
0x0070   0000 0000 0000 0000                          ........
```

```
19:58:26.716801 src.xxx.xxx.xxx.2427 > dst.xxx.xxx.xxx.80: P
823113587:823113667(80) win 512
0x0000   4500 0078 732c 0000 4006 1651 xxxx xxxx      E..xs,..@..Q.r.|
0x0010   xxxx xxxx 097b 0050 310f b773 7ce2 c9a0      .r...{.P1..s|...
0x0020   5008 0200 d559 0000 4745 5420 632b 6469      P....Y..GET.c+di
0x0030   720a 4745 5420 432b 4449 520a 2f2f 6874      r.GET.C+DIR.//ht
0x0040   7470 2031 2e30 0a00 0000 0000 0000 0000      tp.1.0.........
0x0050   0000 0000 0000 0000 0000 0000 0000 0000      ................
0x0060   0000 0000 0000 0000 0000 0000 0000 0000      ................
0x0070   0000 0000 0000 0000                          ........
```

Further packet crafting usage

I will now show you how to do what is called Idle Host Scanning. What this means exactly is that we are using one machines ip addy to scan the target computer for open services. To simplify I will use a random ip addy to scan another addy. This I will do by using HPing of course. The one other caveat is that someone's machine needs to be idle. By that I mean not being used by him. This is needed because while I am spoofing his address and sending syn packets to my target I will be sending syn packets as well to his machine to monitor his IP Id numbers. It is through the monitoring of said numbers that we will know if the target machine has open services or not.

When a machine is idle and you send syn packets to it the IP Id numbers will normally go up in a predictable sequence. If the sequence varies it is because the host is now active. By this I mean that the target machine will send to his computer a syn/ack. His machine will respond with an ack packet. This communication between the two will cause the IP Id numbers to change from it's predictable sequence. Thus indicating to us that our spoofed machine has found an open port. All this is done without exposing ourselves to the target machine. The stacks that are generating predictable IP ID numbers are all Windows based systems as well as some older Linux variants. For more information on this scanning technique please see the following url for several examples of it. http://www.hping.org/papers.html However for clarities sake a picture is worth a thousand words I was once told so please see the diagram below for a visual representation.



The above noted diagram will help explain and visualize the Idle Host Scan technique first discovered by the author of Hping (Salvatore Sanfilippo).

The biggest advantage of the idle host scan is that it allows someone to scan a remote machine with total anonymity. Anonymity is of great concern to someone is who scanning your machine for open ports. It is one of the first steps taken before one will attempt to exploit a machine.

Now for ease of understanding I have labeled all 3 machines involved with non-routable IP's.

10.10.10.101 This machine will be the person doing the idle host scan
172.16.0.101 This will be the machine whose address is being spoofed by 10.10.*
192.168.2.101 This is the machine that 10.10.* wants to scan using 172.16.* addy

Now what happens is that the person doing the scan will have two sessions of hping open. The first hping session will be used to send syn packets to the computer whose address he will be spoofing. This is done so that the person doing the scan can monitor the spoofed hosts IP ID numbers. As a side note this will only work on operating systems who generate predictable IP ID numbers. All Microsoft operating systems do generate predictable IP ID numbers as of this time. Older versions of linux were prone to this as well as older releases of UNIX.

Now what happens is that the scanner 10.10.* will send syn packets to a range of ports using the spoofed IP address 17.16.* to 192.168.* While the scanner is sending these packets he is also sending the afore-mentioned syn packets to 172.16.* who he is spoofing. He does this to monitor for any changes in the IP ID increments. If the IP ID increments were to change from 1 up to a large shift then the scanner would then know that the port he had just scanned using the spoofed address is open.
The scanner using the spoofed address now knows what ports are open on his potential

target. This was all done with total anonymity for the scanner. If the target is using a firewall which has logging or an IDS system the only IP address showing up will be that of the spoofed address 172.16.* This in a nutshell is what is called the Idle Host Scan.

To accomplish said attack we will need to have two sessions of HPing going as well as tcpdump running. 1st session of HPing will contain the below command syntax

hping -S xxx.xxx.xxx.xxx -a xxx.xxx.xxx.xxx -p ++21
-S This again is a syn packet
The first xxx.xxx.xxx.xxx is our target machine
-a The switch used to spoof an ip addy
The xxx.xxx.xxx.xxx after the –a is our spoofed address
-p The switch used to specify destination port
++21 Tells hping to syn packet port 21 on up sequentially

The 2nd session of HPing will contain the below noted command syntax

hping -1 xxx.xxx.xxx.xxx

-1 Tells Hping to send ICMP packet
xxx.xxx.xxx.xxx This is the host your checking for IP ID number increment

By sending icmp packets to his machine I will get back the info I need to execute this. I will get back ttl's and more importantly of course the IP Id numbers. I will keep pinging his box all the while I am sending spoofed syn packets to the target machine in the hope they respond. This will result in his machine changing it's IP Id numbers from it predictable sequence. Thus indicating that it has found an open port. Be aware though that this will only work with a middle man with whom you can monitor it's IP Id numbers. If you have a machine which is running no services, and is firewalled this will not work. Seen as any packets icmp/syn or otherwise will simply be dropped. Your best bet would be to ask someone you know who has a broadband account if they would be willing to let you experiment with their machine. Either way here is a url that does an excellent job of explaining the IP ID attack. There are many more out there just google for them. http://www.bursztein.net/secu/temoinus.html This is a perfect example of pulling of the exploit or scan to truly understand, and thusly be able to defend against. Also to be able to recognize it when and if it hits your networks.
I have included on this page some examples of HPing strings and the feedback as well as the tcpdump logs. Feel free to experiment with the below noted. Not just that mess around with fragmented packets, setting your X and Y flags and the like. You will only learn by playing around.

As mentioned earlier in the paper you can use Hping to test out your IDS ruleset by sending crafted packets to it, and confirming it is responding to certain stimulus such as Null packets for example. The below noted examples clearly show some examples of

how to do this. This is exactly how Hping is able to help secure one's network. Tools such as Nessus are excellent for vulnerability scanning, and Hping's IDS testing capabilities are complimentary as well. These two tools used together can greatly enhance one's security posture. On to the examples for now!

The purpose of the following tcpdump traces, snort output, and Hping command line syntax is to demonstrate the value of Hping. It's crafted packets will allow you to test and confirm your IDS ruleset. Only the tcp protocol was used in testing for the following examples. Though one can get as creative as one wishes with the other supported protocols, and tcp fields under Hping. For the below noted snort output, Snort 2.0 build 72 was used along with the default ruleset.

For ease of viewing and understanding the below noted packets I will give a brief explanation of the fields found within the packet header itself.

Testbox sending crafted packets via Hping is 192.168.2.112

192.168.2.112 sending out a null packet

Command line syntax used for Hping and ensuing output fm Hping

monkeylabs:/home/don # **hping 192.168.2.113 -p 22 -c 2**

HPING 192.168.2.113 (eth0 192.168.2.113): NO FLAGS are set, 40 headers + 0 data bytes

Tcpdump trace of outgoing packets on 192.168.2.112

```
09:15:57.034761 192.168.2.112.2796 > 192.168.2.113.22: . [tcp sum ok] win 512 (ttl 64, id 32865, len 40)
0x0000   4500 0028 8061 0000 4006 743d c0a8 0270        E..(.a..@.t=...p
0x0010   c0a8 0271 0aec 0016 1082 878f 0598 76fa        ...q.........v.
0x0020   5000 0200 080d 0000                            P.......

09:15:58.027608 192.168.2.112.2797 > 192.168.2.113.22: . [tcp sum ok] win 512 (ttl 64, id 21805, len 40)
0x0000   4500 0028 552d 0000 4006 9f71 c0a8 0270        E..(U-..@..q...p
0x0010   c0a8 0271 0aed 0016 2803 9aac 133d 434b        ...q....(....=CK
0x0020   5000 0200 0378 0000                            P....x..
```

Explanation of packet header metrics found in the packet sent above

09:15:57.034761 This is the time that the packet was sent right down to the micro second

192.168.2.112.2796 This is the IP address of the transmitting computer, and source port

> This tells you it is being sent from the IP address on the left to the one on the right

192.168.2.113.22 This is the IP address of the destination computer and dst port

[tcp sum ok] This means that the tcp sequence number is valid

win 512 This tells the dst computer that the src computer can receive up to 512KB

ttl 64 This value represents the time in milliseconds that the packet has to reach it's destination before being discarded.

id 21805 This is the number assigned to the IP header so it can be reassembled in case of fragmentation.

len 40 This is the overall length of the packet itself in bytes.

Will now show the packets as they are received on the destination computer and the ensuing snort alert output.

Testbox receiving crafted packets is 192.168.2.113

Tcpdump trace of incoming packets on 192.168.2.113

```
07:51:00.346081 192.168.2.112.1312 > 192.168.2.113.22: . [tcp sum ok] win 512 (ttl 64,
id 57916, len 40)
0x0000   4500 0028 e23c 0000 4006 1262 c0a8 0270        E..(.<..@..b...p
0x0010   c0a8 0271 0520 0016 39d4 590a 735c 85cc        ...q....9.Y.s\..
0x0020   5000 0200 9675 0000 0c8e 1600 e8a7             P....u.....

07:51:01.342902 192.168.2.112.1313 > 192.168.2.113.22: . [tcp sum ok] win 512 (ttl 64,
id 64711, len 40)
0x0000   4500 0028 fcc7 0000 4006 f7d6 c0a8 0270        E..(....@.....p
0x0010   c0a8 0271 0521 0016 02ed 0c77 4d2e d25c        ...q.!.....wM..\
0x0020   5000 0200 f38c 0000 5549 444c 0d0a             P.......UIDL..
```

Snort output due to crafted packets received on 192.168.2.113

```
[**] [111:9:1] spp_stream4: STEALTH ACTIVITY (NULL scan) detection [**]
06/18-07:51:00.346081 192.168.2.112:1312 -> 192.168.2.113:22
TCP TTL:64 TOS:0x0 ID:57916 IpLen:20 DgmLen:40
******** Seq: 0x39D4590A  Ack: 0x735C85CC  Win: 0x200  TcpLen: 20
```

[**] [100:1:1] spp_portscan: PORTSCAN DETECTED to port 22 from 192.168.2.112 (STEALTH) [**]
06/18-07:51:00.832829

[**] [111:9:1] spp_stream4: STEALTH ACTIVITY (NULL scan) detection [**]
06/18-07:51:01.342902 192.168.2.112:1313 -> 192.168.2.113:22
TCP TTL:64 TOS:0x0 ID:64711 IpLen:20 DgmLen:40
******** Seq: 0x2ED0C77 Ack: 0x4D2ED25C Win: 0x200 TcpLen: 20

Portscan.log entry fm 192.168.2.113

Jun 18 07:51:04 192.168.2.112:1316 -> 192.168.2.113:22 NULL ********

****

I will now do show what happens when a XMAS packet is sent. Once again the above noted format will be used. If you become confused by the meaning of some of the packet metrics used please see the earlier explanation of the header metrics.

192.168.2.112 sending out a XMAS packet

Command line syntax used for Hping and ensuing output fm Hping

monkeylabs:/home/don # **hping -S -R -P -A -F -U 192.168.2.113 -p 22 -c 5 -t 118 –y**

HPING 192.168.2.113 (eth0 192.168.2.113): RSAFPU set, 40 headers + 0 data bytes

Tcpdump trace of outgoing packets on 192.168.2.112

08:05:08.858815 192.168.2.112.1211 > 192.168.2.113.22: SFRP [tcp sum ok] 318976457:318976457(0) ack 1818840445 win 512 urg 0 (DF) [tos 0x10] (ttl 118, id 50387, len 40)
0x0000 4510 0028 c4d3 4000 7606 b9ba c0a8 0270 E..(..@.v.....p
0x0010 c0a8 0271 04bb 0016 1303 31c9 6c69 4d7d ...q.....1.liM}
0x0020 503f 0200 23f0 0000 0c8e 1600 e8a7 P?..#.....

08:05:09.857708 192.168.2.112.1212 > 192.168.2.113.22: SFRP [tcp sum ok] 16649609:16649609(0) ack 172211276 win 512 urg 0 (DF) [tos 0x10] (ttl 118, id 30817, len 40)
0x0000 4510 0028 7861 4000 7606 062d c0a8 0270 E..(xa@.v..-...p
0x0010 c0a8 0271 04bc 0016 00fe 0d89 0a43 bc4c ...q.........C.L
0x0020 503f 0200 4d8b 0000 5041 5353 2067 P?..M...PASS.g

Testbox receiving crafted packets is 192.168.2.113

Tcpdump trace of incoming packets on 192.168.2.113

08:05:08.858815 192.168.2.112.1211 > 192.168.2.113.22: SFRP [tcp sum ok] 318976457:318976457(0) ack 1818840445 win 512 urg 0 (DF) [tos 0x10] (ttl 118, id 50387, len 40)
0x0000 4510 0028 c4d3 4000 7606 b9ba c0a8 0270 E..(..@.v.....p
0x0010 c0a8 0271 04bb 0016 1303 31c9 6c69 4d7d ...q.....1.liM}
0x0020 503f 0200 23f0 0000 0c8e 1600 e8a7 P?..#.....

08:05:09.857708 192.168.2.112.1212 > 192.168.2.113.22: SFRP [tcp sum ok] 16649609:16649609(0) ack 172211276 win 512 urg 0 (DF) [tos 0x10] (ttl 118, id 30817, len 40)
0x0000 4510 0028 7861 4000 7606 062d c0a8 0270 E..(xa@.v..-...p
0x0010 c0a8 0271 04bc 0016 00fe 0d89 0a43 bc4c ...q.....C.L
0x0020 503f 0200 4d8b 0000 5041 5353 2067 P?..M...PASS.g

Snort output due to crafted packets received on 192.168.2.113

[**] [111:6:1] spp_stream4: STEALTH ACTIVITY (Full XMAS scan) detection [**]
06/18-08:05:08.858815 192.168.2.112:1211 -> 192.168.2.113:22
TCP TTL:118 TOS:0x10 ID:50387 IpLen:20 DgmLen:40 DF
**UAPRSF Seq: 0x130331C9 Ack: 0x6C694D7D Win: 0x200 TcpLen: 20 UrgPtr: 0x0

[**] [100:1:1] spp_portscan: PORTSCAN DETECTED to port 22 from 192.168.2.112 (STEALTH) [**]
06/18-08:05:08.861384

[**] [111:6:1] spp_stream4: STEALTH ACTIVITY (Full XMAS scan) detection [**]
06/18-08:05:09.857708 192.168.2.112:1212 -> 192.168.2.113:22
TCP TTL:118 TOS:0x10 ID:30817 IpLen:20 DgmLen:40 DF
**UAPRSF Seq: 0xFE0D89 Ack: 0xA43BC4C Win: 0x200 TcpLen: 20 UrgPtr: 0x0

Portscan.log entry fm 192.168.2.113

Jun 18 08:05:12 192.168.2.112:1215 -> 192.168.2.113:22 FULLXMAS **UAPRSF

As seen in the above noted examples Hping is very much capable of testing out an IDS ruleset through the use of crafted packets. This is of value for the simple fact that it does confirm unequivocally that your IDS rulesets are triggering to expected stimulus such as the one's shown above.

Conclusion

Both the benefits and dangers posed by packet crafting are undeniable. Specifically so with a tool like HPing. Like many other tools available today this one can also be used by the "script kiddie". The aforementioned however poses little danger when using such

a tool. In the hands of a capable hacker this tool becomes a very powerful weapon indeed. It is not only helpful in performing scans as shown above it can also be used to send files through firewall rulesets as you can specify the port you will use. There are a great many uses for both the grey/black hat hacker out there.
On the other hand the benefits of packet crafting as an instructional tool are also undeniable. To properly craft packets one has no choice but to learn the conventions and rules of tcp/ip. It is no different if one wants to learn how to effectively use the tcp sequence number feature in HPing. One will have to learn about ISN prediction for use against such stacks as the one used by Windows 98. There are also the loose source and strict source options as well which will show how this Internet Protocol option works. Also as mentioned above there is the aspect of verifying your firewalls ruleset, or your routers access control list manually. This enables you to not blindingly rely upon the output of such auditing tools as Nessus or Saint.
The studying of your crafted packets as well via the tcpdump tool will also allow you to become more familiar with the tcp/ip packet metrics such as the time to live (ttl), the windows size, and others. This is turn will allow you to fingerprint systems by just looking at the metrics vice having to input the data into such programs as p0f.
Overall the benefits of packet crafting far outweigh the dangers posed by such packets. After all by crafting them yourself you will be able to tell when one has made a mistake and improperly crafted some. When it comes to network security every little edge you can give yourself is important.

REFERENCES:

1) Nessus home page URL http://www.nessus.org/
2) Tcpdump home page URL http://www.tcpdump.org/
3) Buffer overflow explanation
http://www.cse.msu.edu/~westrant/symlink/pages/exploits/overflows.htm
4) HPing homepage URL http://www.hping.org/
5)TCP/IP intro http://www.yale.edu/pclt/COMM/TCPIP.HTM
6) Ephemeral ports intro http://wombat.doc.ic.ac.uk/foldoc/foldoc.cgi?ephemeral+port

Part 2: Network Detects

1. Source of Trace

This trace was collected on my company's network. For purposes of sanitation the attackers IP has been changed to 192.xxx.xxx.xxx and the IP being attacked has been changed to 10.xxx.xxx.xxx. The IP that is being attacked here is facing the internet directly, and not behind a router or located within a DMZ. The trace that I was handed to work on is noted below. By the time I got to work on it I only had a flat ascii file, and not a binary one which would have been preferable. It was however too late by then to do a full pull on the attacking IP address as the database had been overwritten with new data.

13:59:43.833031 212.xxx.xxx.xxx.3467 > 13x.xxx.xxx.xxx.21: P 1233702904:1233702920(16) ack 1469443794 win 65502
0x0000 4500 0038 92ea 0000 2c06 fdc7 xxxx xxxx E..8....,.....?q
0x0010 xxxx xxxx 0d8b 0015 4988 cff8 5795 eed2 ..fs....I...W...
0x0020 5018 ffde bc69 0000 5553 4552 2061 6e6f P....i..USER.ano
0x0030 6e79 6d6f 7573 0d0a nymous..

13:59:44.628180 212.xxx.xxx.xxx.3467 > 13x.xxx.xxx.xxx.21: P 16:23(7) ack 31 win 65472
0x0000 4500 002f 92f5 0000 2c06 fdc5 xxxx xxxx E../....,.....?q
0x0010 xxxx xxxx 0d8b 0015 4988 d008 5795 eef0 ..fs....I...W...
0x0020 xxxx xxxx 83ba 0000 4357 4420 2f0d 0a P.......CWD./..

13:59:45.437923 212.xxx.xxx.xxx.3467 > 13x.xxx.xxx.xxx.21: P 23:42(19) ack 75 win 65428
0x0000 4500 003b 935f 0000 2c06 fd4f xxxx xxxx E..;._..,..O..?q
0x0010 xxxx xxxx 0d8b 0015 4988 d00f 5795 ef1c ..fs....I...W...
0x0020 5018 ff94 456f 0000 4445 4c45 202f 316b P...Eo..DELE./1k
0x0030 6274 6573 742e 7074 660d 0a btest.ptf..

13:59:46.521105 212.xxx.xxx.xxx.3467 > 13x.xxx.xxx.xxx.21: P 42:50(8) ack 108 win 65395
0x0000 4500 0030 939a 0000 2c06 fd1f xxxx xxxx E..0....,.....?q
0x0010 xxxx xxxx 0d8b 0015 4988 d022 5795 ef3d ..fs....I.."W..=
0x0020 5018 ff73 723a 0000 5459 5045 2041 0d0a P..sr:..TYPE.A..

13:59:48.034855 212.xxx.xxx.xxx.3467 > 13x.xxx.xxx.xxx.21: P 78:97(19) ack 146 win 65357
0x0000 4500 003b 941b 0000 2c06 fc93 xxxx xxxx E..;....,.....?q
0x0010 xxxx xxxx 0d8b 0015 4988 d046 xxxx xxxx ..fs....I..FW..c
0x0020 xxxx xxxx 331c 0000 5354 4f52 202f 316b P..M3...STOR./1k
0x0030 6274 6573 742e 7074 660d 0a btest.ptf..

13:59:48.985094 212.xxx.xxx.xxx.3486 > 13x.xxx.xxx.xxx.1026: S 1236633085:1236633085(0) ack 1215492453 win 65535 <mss 1460>
0x0000 4500 002c 944c 0000 2c06 fc71 xxxx xxxx E..,.L..,..q..?q
0x0010 xxxx xxxx 0d9e 0402 49b5 85fd 4872 f165 ..fs....I...Hr.e
0x0010 8389 6673 0d9e 0402 49b5 85fd 4872 f165 ..fs....I...Hr.e
0x0020 6012 ffff 7edd 0000 0204 05b4 0000 `...~.....

13:59:49.777698 212.xxx.xxx.xxx.3486 > 13x.xxx.xxx.xxx.1026: P 1:1025(1024) ack 1 win 65535
0x0000 4500 0428 948a 0000 2c06 f837 xxxx xxxx E..(....,..7..?q
0x0010 xxxx xxxx 0d9e 0402 49b5 85fe 4872 f165 ..fs....I...Hr.e
0x0020 5018 ffff 5436 0000 7465 7374 7465 7374 P...T6..testtest
0x0030 7465 7374 7465 7374 7465 7374 7465 7374 testtesttesttest

```
0x0040   7465 7374 7465 7374 7465 7374 7465 7374      testtesttesttest
0x0050   7465 7374 7465 7374 7465 7374 7465 7374      testtesttesttest
0x0060   7465 7374 7465 7374 7465                     testtestte

13:59:53.067647 212.xxx.xxx.xxx.3467 > 13x.xxx.xxx.xxx.21: P 125:133(8) ack 249
win 65254
0x0000   4500 0030 95a0 0000 2c06 fb19 xxxx xxxx      E..0....,.....?q
0x0010   xxxx xxxx 0d8b 0015 4988 d075 5795 efca      ..fs....I..uW...
0x0020   5018 fee6 71e7 0000 5459 5045 2041 0d0a      P...q...TYPE.A..

13:59:53.761518 212.xxx.xxx.xxx.3467 > 13x.xxx.xxx.xxx.21: P 133:152(19) ack 268
win 65235
0x0000   4500 003b 95e3 0000 2c06 xxxx xxxx xxxx      E..;....,.....?q
0x0010   xxxx xxxx 0d8b 0015 4988 d07d 5795 efdd      ..fs....I..}W...
0x0020   5018 fed3 2ef4 0000 5245 5452 202f 316b      P.......RETR./1k
0x0030   6274 6573 742e 7074 660d 0a                  btest.ptf..

13:59:54.002394 212.xxx.xxx.xxx.3487 > 13x.xxx.xxx.xxx.1026: S
1238101234:1238101234(0) ack 3791771758 win 65535 <mss 1460>
0x0000   4500 002c 95e7 0000 2c06 fad6 xxxx xxxx      E..,....,.....?q
0x0010   xxxx xxxx 0d9f 0402 49cb ecf2 e201 d86e      ..fs....I......n
0x0020   6012 ffff 9738 0000 0204 05b4 0000           `....8........

13:59:54.708872 212.xxx.xxx.xxx.3467 > 13x.xxx.xxx.xxx.21: P 152:160(8) ack 352
win 65151
0x0000   4500 0030 95f1 0000 2c06 fac8 xxxx xxxx      E..0....,.....?q
0x0010   xxxx xxxx 0d8b 0015 4988 d090 5795 f031      ..fs....I...W..1
0x0010   8389 6673 0d8b 0015 4988 d090 5795 f031      ..fs....I...W..1
0x0020   5018 fe7f 71cc 0000 5459 5045 2041 0d0a      P...q...TYPE.A..

13:59:56.046245 212.xxx.xxx.xxx.3467 > 13x.xxx.xxx.xxx.21: P 188:198(10) ack 390
win 65113
0x0000   4500 0032 95fa 0000 2c06 fabd xxxx xxxx      E..2....,.....?q
0x0010   xxxx xxxx 0d8b 0015 4988 d0b4 5795 f057      ..fs....I...W..W
0x0020   5018 fe59 0a5a 0000 4c49 5354 202d 6c61      P..Y.Z..LIST.-la
0x0030   0d0a

13:59:56.949650 212.xxx.xxx.xxx.3467 > 13x.xxx.xxx.xxx.21: P 198:217(19) ack 436
win 65067
0x0000   4500 003b 95ff 0000 2c06 faaf xxxx xxxx      E..;....,.....?q
0x0010   xxxx xxxx 0d8b 0015 4988 d0be 5795 f085      ..fs....I...W...
0x0020   5018 fe2b 44c0 0000 4445 4c45 202f 316b      P..+D...DELE./1k
0x0030   6274 6573 742e 7074 660d 0a                  btest.ptf..
```

14:00:12.051944 212.xxx.xxx.xxx.3467 > 13x.xxx.xxx.xxx.21: P
1233703174:1233703191(17) ack 1469444395 win 64901
0x0000 4500 0039 9647 0000 2c06 fa69 xxxx xxxx E..9.G..,..i..?q
0x0010 xxxx xxxx 0d8b 0015 4988 d106 5795 f12b ..fs....I...W..+
0x0020 5018 fd85 7cfa 0000 4445 4c45 202f 7370 P...|...DELE./sp
0x0030 6163 652e 6173 700d 0a ace.asp..

2. Detect was generated by

This detect was generated by an unknown alarm on our Shadow/Netranger IDS system. The alert is unknown due to the fact that this trace was handed to me for analysis after the fact, and the alert itself being deleted.

3. Probability that source IP was spoofed

The likelihood that the source IP was spoofed in this trace is extremely unlikely. This is due to the fact that the attacker is trying to upload some files to an anonymous FTP server. To do this successfully he will require that the TCP/IP 3 way handshake[1] be fully completed.

4. Description of the attack

The attacker likely used a program called Grim's Ping[2] to scan for open ftp shares. Once open ftp shares were found (aka: anonymous ftp) the attacker then verified that uploads were allowed by uploading a test file, followed by other ftp commands which will be shown below in detail. By uploading and probably downloading the file the attacker is able to verify the upload and download speed of the site. (note again please that the packet trace is incomplete and appears to have some dropped packets) Also the space.asp command is used to verify how much space there is to host files.It is assumed that the tool used is Grim's Ping due to the similarity in traffic noted by other people which will be shown in the correlation section.
Though the tool was not used by myself to confirm my findings due to hardware limitations it can be safely assumed to be Grim's Ping due to the activity noted and the correlations with other people traffic noted once again in the correlation section below.

The attacking computer is IP 192.xxx.xxx.xxx and the computer being attacked is 10.xxx.xxx.xxx

13:59:42.961152 192.xxx.xxx.xxx.3467 > 10.xxx.xxx.xxx.21: S
1233702903:1233702903(0) win 40148 <mss 1460,nop,wscale 4,nop,nop,sackOK>
0x0000 4500 0034 92e5 0000 2c06 fdd0 xxx xxxx E..4....,.....?q
0x0010 xxxx xxxx 0d8b 0015 4988 cff7 0000 0000 ..fs....I.....
0x0020 8002 9cd4 ad11 0000 0204 05b4 0103 0304 .....
0x0030 0101 0402

Right here the attacker has sent a syn packet to verify if an ftp service is running on the computer. Our system did not log the syn/ack sent back our company computer. However there is no doubt that it did as evidenced by the follow on packets noted below. The reason that the attacker sent a syn packet is because if there is an ftp service running on the machine the ftp service will respond with a syn/ack indicating that there a service listening on that port.

```
13:59:43.833031 192.xxx.xxx.xxx.3467 > 10.xxx.xxx.xxx.21: P
1233702904:1233702920(16) ack 1469443794 win 65502
0x0000   4500 0038 92ea 0000 2c06 fdc7 xxxx xxxx      E..8....,.....?q
0x0010   xxxx xxxx  0d8b 0015 4988 cff8 5795 eed2      ..fs....I...W...
0x0020   5018 ffde bc69 0000 5553 4552 2061 6e6f      P....i..USER.ano
0x0030   6e79 6d6f 7573 0d0a                          nymous..
```

After the 3 way handshake is completed the machine running the ftp service pushes a login prompt to the attacking machine. At this time the attacker attempts to log in as anonymous as evidenced the ascii breakout in the above packet.

```
13:59:44.628180 192.xxx.xxx.xxx.3467 > 10.xxx.xxx.xxx.21: P 16:23(7) ack 31 win
65472
0x0000   4500 002f 92f5 0000 2c06 fdc5 xxxx xxxx      E../....,.....?q
0x0010   xxxx xxxx  0d8b 0015 4988 d008 5795 eef0      ..fs....I...W...
0x0020   5018 ffc0 83ba 0000 4357 4420 2f0d 0a        P.......CWD./..
```

Now that the attacker has successfully logged into the ftp server as anonymous he issues the above noted command of CWD[3]. This command when issued will change the working directory that one is presently in. In essence this will allow you to navigate the directory structure of the ftp server.

```
13:59:45.437923 192.xxx.xxx.xxx.3467 > 10.xxx.xxx.xxx.21: P 23:42(19) ack 75 win
65428
0x0000   4500 003b 935f 0000 2c06 fd4f xxxx xxxx      E..;._..,..O..?q
0x0010   xxxx xxxx  0d8b 0015 4988 d00f 5795 ef1c      ..fs....I...W...
0x0020   5018 ff94 456f 0000 4445 4c45 202f 316b      P...Eo..DELE./1k
0x0030   6274 6573 742e 7074 660d 0a                  btest.ptf..
```

This is where our system has apparently dropped a packet. This is inferred due to the fact that the attacker is now deleting the 1k btest.ptf file. As seen in the above packet by the DELE[4] command.It is unlikely that the ftp server has such a file, however it is possible. It does make more sense though that the attacker has just uploaded this file to test the ftp server. By that I mean is the ftp server configured for only downloads or uploads as well. It should be mentioned as well that the window size of the attacker shown here is 65428, and variants thereof throughout the trace. This would indicate a Cisco device. The fact remains that Grim's Ping is a Win32 based tool. It is possible that the attacker has changed the Windows size in his tcp/ip stack as a method of obfuscating the operating system that he is using.

13:59:46.521105 192.xxx.xxx.xxx.3467 > 10.xxx.xxx.xxx.21: P 42:50(8) ack 108 win 65395
0x0000 4500 0030 939a 0000 2c06 fd1f xxxx xxxx E..0....,.....?q
0x0010 xxxx xxxx 0d8b 0015 4988 d022 5795 ef3d ..fs....I.."W..=
0x0020 5018 ff73 723a 0000 5459 5045 2041 0d0a P..sr:..TYPE.A..

Here the attacker is specifying Ascii mode by the TYPE A command. There are two modes of use in FTP. They are ascii and binary.

13:59:47.360182 192.xxx.xxx.xxx.3467 > 10.xxx.xxx.xxx.21: P 50:78(28) ack 127 win 65376
0x0000 4500 0044 93e7 0000 2c06 fcbe xxxx xxxx E..D....,.....?q
0x0010 xxxx xxxx 0d8b 0015 4988 d02a 5795 ef50 ..fs....I..*W..P
0x0020 5018 ff60 8a38 0000 504f 5254 2032 3132 P..`.8..PORT.xxx
0x0030 2c31 3630 2c36 332c 3131 332c 3133 2c31 xxx.xxx
0x0040 3538 0d0a xxx.

The attacker is now using the PORT command to specify a non-default user side data port. The ip address of the attacker has been sanitized as seen by the xxx's. following the PORT command.

13:59:48.034855 192.xxx.xxx.xxx.3467 > 10.xxx.xxx.xxx.21: P 78:97(19) ack 146 win 65357
0x0000 4500 003b 941b 0000 2c06 fc93 xxxx xxxx E..;....,.....?q
0x0010 xxxx xxxx 0d8b 0015 4988 d046 5795 ef63 ..fs....I..FW..c
0x0020 5018 ff4d 331c 0000 5354 4f52 202f 316b P..M3...STOR./1k
0x0030 6274 6573 742e 7074 660d 0a btest.ptf..

The attacker is now issuing the STOR command which is the command issued to copy the 1kb test.ptf file noted above over to the ftp server. He is doing this once again to confirm the ftp server is allowing uploads.

13:59:49.777698 192.xxx.xxx.xxx.3486 > 10.xxx.xxx.xxx.1026: P 1:1025(1024) ack 1 win 65535
0x0000 4500 0428 948a 0000 2c06 f837 xxxx xxxx E..(....,..7..?q
0x0010 xxxx xxxx 0d9e 0402 49b5 85fe 4872 f165 ..fs....I...Hr.e
0x0020 5018 ffff 5436 0000 7465 7374 7465 7374 P...T6..testtest
0x0030 7465 7374 7465 7374 7465 7374 7465 7374 testtesttesttest
0x0040 7465 7374 7465 7374 7465 7374 7465 7374 testtesttesttest
0x0050 7465 7374 7465 7374 7465 7374 7465 7374 testtesttesttest
0x0060 7465 7374 7465 7374 7465 testtestte

Here we see the 1kb test file being transferred over via ascii mode which was specified a couple of packets ago as seen 3 packets above ie: TYPE A

14:00:16.040112 192.xxx.xxx.xxx.3467 > 10.xxx.xxx.xxx.21: P 23:47(24) ack 94 win 64808
0x0000 4500 0040 9740 0000 2c06 9822 xxxx xxxx E..@.@..,..i..?q
0x0010 xxxx xxxx 0d8b 0015 4988 d11d 5795 f188 ..fs....I...W...
0x0020 5018 fd28 ab36 0000 5354 4f52 202f 7870 P..(.6..STOR./xp
0x0030 2d41 6e74 6953 7079 3345 2e65 7865 0d0a -AntiSpy3E.exe..

Here we see the attacker actually transfer what is probably warez ie: the program Anti-Spy3.exe

14:00:24.003492 192.xxx.xxx.xxx.3467 > 10.xxx.xxx.xxx.21: P 47:71(24) ack 178 win 64724
0x0000 4500 0040 9789 0000 2c06 f920 xxxx xxxx E..@....,.....?q
0x0010 xxxx xxxx 0d8b 0015 4988 d135 5795 f1dc ..fs....I..5W...
0x0020 5018 fcd4 bd3a 0000 4445 4c45 202f 7870 P....:..DELE./xp
0x0030 2d41 6e74 6953 7079 3345 2e65 7865 0d0a -AntiSpy3E.exe..

Now the attacker deletes the warez program here transferred over. This further displays the level of control he now has. He can now upload and delete files at will. The attacker then goes on to repeat the above noted several times. Thereby confirming his ability to both upload, delete, and navigate the ftp server directory structure. Once again it should be noted that the file I received was incomplete and looks as if there are dropped packets as well.

The above noted traffic also appears elsewhere as seen below. It is because of this type of traffic also being noted by others that I tentatively conclude this to be Grim's Ping.
The below noted was taken from http://eyeonsecurity.org/papers/pubscanning.pdf
I will highlight the appropriate sections below which lend weight to my assertion that Grim's Ping seems to be the tool in question here that was used.

Once again this is Grim's Ping Automated tool, with Companion software, as you will see further down.

[2] Wed 13Jun01 14:23:49 - (000019) CWD /
[6] Wed 13Jun01 14:23:49 - (000019) 250 Directory changed to /
[2] Wed 13Jun01 14:23:49 - (000019) TYPE I
[6] Wed 13Jun01 14:23:49 - (000019) 200 Type set to I.
[2] Wed 13Jun01 14:23:50 - (000019) PORT 213,51,52,27,17,98
[6] Wed 13Jun01 14:23:50 - (000019) 200 PORT Command successful.
[2] Wed 13Jun01 14:23:50 - (000019) STOR /1mbtest.ptf

The scanner uploads a 1mb test file to the root directory.

```
[6] Wed 13Jun01 14:23:50 - (000019) 150 Opening BINARY mode data connection for
1mbtest.ptf.
[4] Wed 13Jun01 14:23:50 - (000019) Receiving file d:\anonftp\1mbtest.ptf
[4] Wed 13Jun01 14:25:16 - (000019) Received file d:\anonftp\1mbtest.ptf successfully
(11.9 Kb/sec - 1048578 bytes)
[6] Wed 13Jun01 14:25:16 - (000019) 226-Maximum disk quota limited to 300000 Kbytes
[6] Wed 13Jun01 14:25:16 - (000019) Used disk quota 1024 Kbytes, available 298975 Kbytes
[6] Wed 13Jun01 14:25:16 - (000019) 226 Transfer complete.
[2] Wed 13Jun01 14:25:17 - (000019) PORT 213,51,52,27,6,55
[6] Wed 13Jun01 14:25:17 - (000019) 200 PORT Command successful.
[2] Wed 13Jun01 14:25:17 - (000019) TYPE I
[6] Wed 13Jun01 14:25:17 - (000019) 200 Type set to I.
[2] Wed 13Jun01 14:25:17 - (000019) RETR /1mbtest.ptf
```

Then it downloads the file back.

```
[6] Wed 13Jun01 14:25:17 - (000019) 150 Opening BINARY mode data connection for
1mbtest.ptf (1048578 bytes).
[3] Wed 13Jun01 14:25:17 - (000019) Sending file d:\anonftp\1mbtest.ptf
[3] Wed 13Jun01 14:26:29 - (000019) Sent file d:\anonftp\1mbtest.ptf successfully (14.3
Kb/sec - 1048578 bytes)
[6] Wed 13Jun01 14:26:29 - (000019) 226-Maximum disk quota limited to 300000 Kbytes
[6] Wed 13Jun01 14:26:29 - (000019) Used disk quota 1024 Kbytes, available 298975 Kbytes
[6] Wed 13Jun01 14:26:29 - (000019) 226 Transfer complete.
[2] Wed 13Jun01 14:26:29 - (000019) TYPE A
[6] Wed 13Jun01 14:26:29 - (000019) 200 Type set to A.
[2] Wed 13Jun01 14:26:30 - (000019) PORT 213,51,52,27,9,50
[6] Wed 13Jun01 14:26:30 - (000019) 200 PORT Command successful.
```



```
[2] Wed 13Jun01 14:26:30 - (000019) LIST -la
[6] Wed 13Jun01 14:26:30 - (000019) 150 Opening ASCII mode data connection for /bin/ls.
[6] Wed 13Jun01 14:26:30 - (000019) 226-Maximum disk quota limited to 300000 Kbytes
[6] Wed 13Jun01 14:26:30 - (000019) Used disk quota 1024 Kbytes, available 298975 Kbytes
[6] Wed 13Jun01 14:26:30 - (000019) 226 Transfer complete.
[2] Wed 13Jun01 14:26:30 - (000019) DELE /1mbtest.ptf
[6] Wed 13Jun01 14:26:30 - (000019) 250 DELE command successful.
```

And finally delete the test file. Till now the following statistics are gathered from my site:

Upload/Download is enabled, my speed, deletable files (I had changed the configuration to allow deletion of files by the anonymous user).

```
[2] Wed 13Jun01 14:26:30 - (000019) TYPE A
[6] Wed 13Jun01 14:26:30 - (000019) 200 Type set to A.
[2] Wed 13Jun01 14:26:30 - (000019) PORT 213,51,52,27,9,51
[6] Wed 13Jun01 14:26:30 - (000019) 200 PORT Command successful.
[2] Wed 13Jun01 14:26:31 - (000019) STOR /space.asp
[6] Wed 13Jun01 14:26:31 - (000019) 150 Opening ASCII mode data connection for space.asp.
[4] Wed 13Jun01 14:26:31 - (000019) Receiving file d:\anonftp\space.asp
[4] Wed 13Jun01 14:26:31 - (000019) Received file d:\anonftp\space.asp successfully (4.91
Kb/sec - 2648 bytes)
[6] Wed 13Jun01 14:26:31 - (000019) 226-Maximum disk quota limited to 300000 Kbytes
[6] Wed 13Jun01 14:26:31 - (000019) Used disk quota 2 Kbytes, available 299997 Kbytes
[6] Wed 13Jun01 14:26:31 - (000019) 226 Transfer complete.
```

This file is included with Grim's Ping companion and will give out information about the ftp server, as described in the tools section.

At the same moment the following log is found from my HTTP server (IIS/5.0):

2001-06-13 12:26:38 213.51.52.27 - x.x.x.x 80 GET /space.asp |-|0|404_Object_Not_Found 404 -

Of course, if I had used the same directory for both http and ftp, the asp script would have executed and given out further information about my machine to the scanner. Also note the timing.

[2] Wed 13Jun01 14:26:38 - (000019) **DELE /space.asp**
[6] Wed 13Jun01 14:26:38 - (000019) 250 DELE command successful.
[5] Wed 13Jun01 14:26:38 - (000019) Closing connection for user ANONYMOUS (00:02:49 connected)

Once the ASP files is not found on the HTTP server, the scanner just deletes the file, and leaves little or no trace of his scan and moves on to the next target.

Problems caused by FTP Pub scanning

Till now this is what I got. Maybe if I wait longer I'd find myself full of Warez and my IP address on some Warez site, IRC channel or bulletin board, with most of my bandwidth being abused, not that nice. Apart from this Corporate sites could be targeted by the software makers and accused as distributing illegal software (Warez) and similar legal issues.

Of course, when the Administrator does not set a quota for anonymous FTP servers, it is very possible that pub scanners will take up all free space. This is probably the most popular type of denial of service.

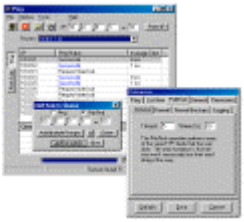
---

5. Attack mechanism

The attack mechanism noted here is the tool known as Grim's Ping. This tool is an automated scanner used to search for open ftp shares. By that I mean anonymous logins on ftp servers. Once these anonymous logins are found a test file is then loaded to verify that one can upload to the ftp server. Once done the file is deleted. Grim's Ping as such will support various plug-ins[5] as well. As noted below it has a GUI interface for ease of use, and various uses as seen in the preferences dialog box. The below noted screen captures are taken from http://www.webattack.com/get/grimping.shtml

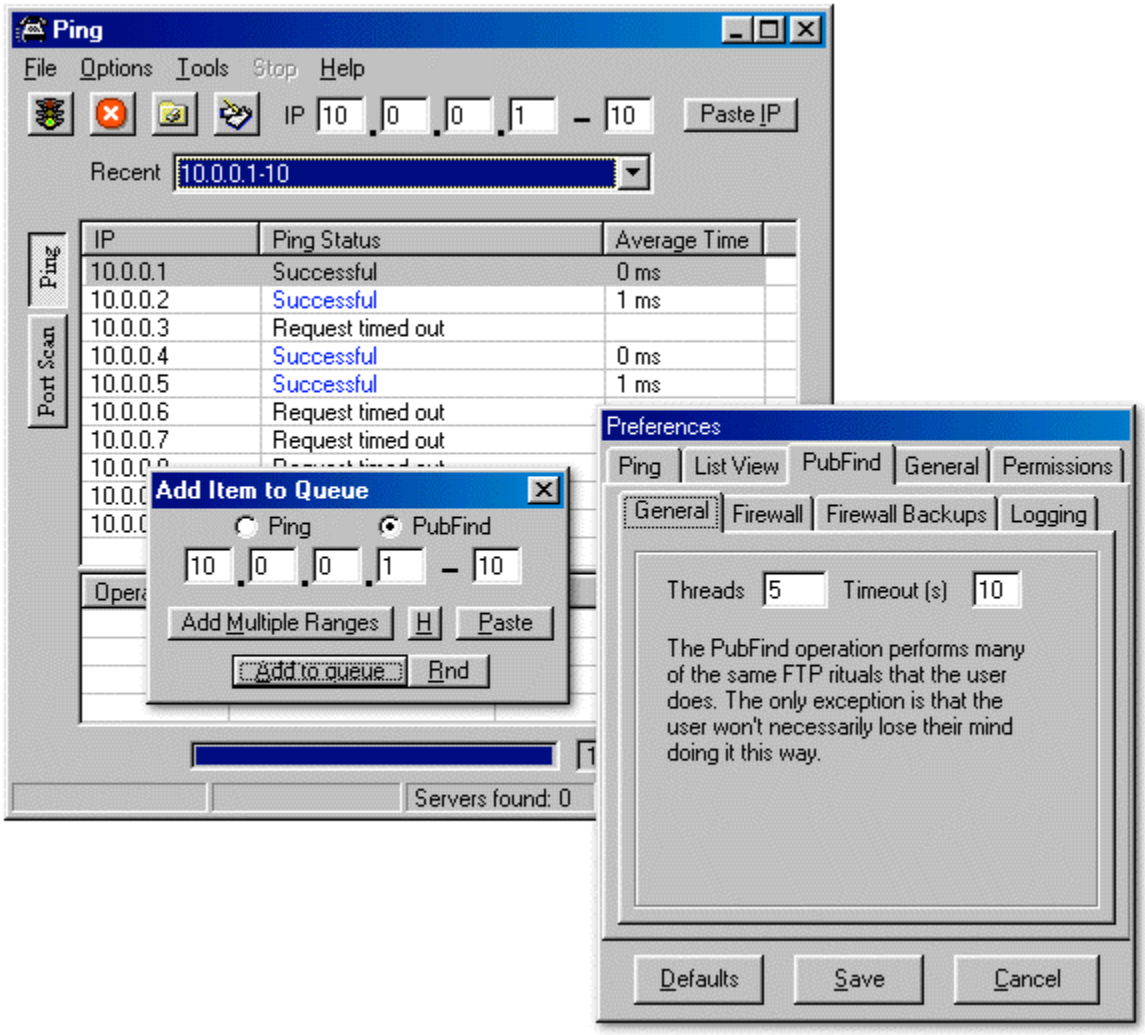
Home :· Freeware :· TCP/IP Networking :· Network Information

Grims Ping 1.7.1
advanced ping tool



click for full size

Grim's Ping is a network scanner and ping tool with many advanced features and options not commonly found in every tool of this kind. It includes advanced firewall support, public directory scanning, support for directory permissions, a customizable port list and much more. If upgrading, uninstall any previous version first.



6. Correlations

This tool has been noted by a great number of system administrators who run ftp servers with anonymous login enabled. There is a good thread on Neohapsis that describes this; http://archives.neohapsis.com/archives/snort/2002-04/0223.html
There is also a good discussion detailing Grim's Ping at the following sites;
http://lists.jammed.com/incidents/2001/12/0192.html
http://lists.jammed.com/incidents/2001/05/0146.html
As well Michael Hotaling a GCIA wrote his part I on Grim's Ping and provides excellent insights into the tool itself and it's usage. The traffic generated and captured by him lends credence to my assertion that this is the work of Grim's Ping.

7. Evidence of active targeting

The tool Grim's Ping can actively among other things search for open ftp shares. The user simply needs to input a range he wishes to search. This would be evidence of active targeting, however not in the conventional sense. The tool is looking for the open shares but, it is just combing through ip ranges to find them, vice actively targeting unique ip addresses.
So in essence it does not meet the classical definition of active targeting. That being said it is this analyst's opinion that a tool searching for specific vulnerabilities over even a large IP range could or should be considered active targeting.

8. Severity

The severity is calculated using the following formula;
Severity=(criticality + lethality) – (system countermeasures + network countermeasures)

Criticality = 4 This is issued a four for the reason that with new legislation introduced in North America regarding the use of pirated software ie: RIAA/DMCA it could be damaging to a corporation to be found hosting pirated software on it's computer assets. This is not to mention the possible storing of child pornography which would be very damaging to a corporation, and possibly it's publicly traded stock.

Lethality = 4 The aim of Grim's Ping is simply to store warez. This in and of itself would not result in damage to the system, or a denial of service. There could be a slowdown in download speed due to heavy warez downloading, but that would be the extent of it. That being said there are several vulnerabilities associated with running Ftp servers;

a) http://www.cert.org/advisories/CA-2001-33.html
b) http://www.cert.org/advisories/CA-2001-07.html
c) http://www.cert.org/advisories/CA-2000-13.html

These same vulnerabilities were also noted by the aforementioned Michael Hotaling in his paper on Grim's Ping. In light of these vulnerabilities the lethality factor is a high one.

System countermeasure = 1 There is very little in the way of countermeasures in place on this machine. By that I mean there is no authentication for logins to the ftp server itself. The obvious way to secure this would be to disable uploads. However this may be unfeasible on this machine dependant upon what services the people want offered on this

machine. The other way would simply be to enable authentication via password. As it stands anyone can upload files to the server which is a very poor security practice.

Network Countermeasures = 4 The IDS system in place fired off an alarm which caused an analyst to verify the packet and it's contents. This in turn led to the discovery of the Grim's Ping tool doing it's work. In essence the networks countermeasure did their work quite well.

Severity = (4 + 4) – (1 + 4) = 3

The tool Grim's Ping did successfully manage to do it's work. This was mitigated by the fact however that the IDS system protecting the network did it's job and detected the scan.

9. Defensive recommendation

All files uploaded to the ftp server become owned by the ftp server. This would prevent further manipulation of the files by the uploader. Make sure that the ftp root directory and it's sub-directories are not owned by the ftp account. However the defensive recommendations also vary depending on the operating system that the ftp server is run on. The simplest defence for this type of attack is simply to disable anonymous ftp login.

10. Multiple choice question

Q: What is the main purpose of the tool known as Grim's Ping

a) To check for file upload/download ratios
b) To check for open ftp shares
c) To verify the ftp server software type
d) To verify download speeds

A: (b) To check for open ftp shares

References

1) http://www.wikipedia.org/wiki/TCP
2) http://grimsping.cjb.net/
3) http://www.freesoft.org/CIE/RFC/959/23.htm
4) http://www.freesoft.org/CIE/RFC/959/21.htm
5) http://grimsping.cjb.net/downloads.htm

Part 2 Network Detects cont'd (2nd detect)

1. Source of trace

This trace was taken from the 2002.9.27 binary log file at the following url;

http://www.incidents.org/logs/Raw/ There is no information available about the network itself or it's topology.There is no information either as to the placement of the IDS system. Since the checksums are all invalid, it's assumed that all of the IP's in the trace have been changed, and as such will be used as is without further obfuscation.

2. Detect was generated by

This detect was generated by Snort Version 1.8.4 (Build 99)
By Martin Roesch (roesch@sourcefire.com, www.snort.org)
The standard rule set that comes with this build of Snort was used with no additions to it. The Snort rules that triggered the alerts that I will analyze are;

alert tcp $EXTERNAL_NET any -> $HOME_NET 3128
(msg:"SCAN Squid Proxy attempt"; flags:S;
classtype:attempted-recon; sid:618; rev:2;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 8080
(msg:"SCAN Proxy \(8080\) attempt"; flags:S;
classtype:attempted-recon; sid:620; rev:2;)

3. Probability that source IP was spoofed

The probability that the source IP was spoofed in this case is low. This is due to the fact that the person running the scan would want the syn/ack's to go back to them vice a spoofed address. That being said it is also possible that it was spoofed. This would mean the attacker was using the Idle Host Scan technique[1] Taking a look at the metrics provided in the packets noted below though seem to indicate this is a linux machine. Note the mss of 1460 and the ttl of 43. As well there is the win size of 5840. This last metric of win 5840 does go against the OS being linux though. Though this can easily be changed by using a packet shaping utility such as HPing[2] .

13:10:05.156507 66.28.100.206.34806 > 32.245.87.117.8080: S [bad tcp cksum 1913!]
1456426247:1456426247(0) win 5840 <mss 1460,sackOK,timestamp 4821579
0,nop,wscale 0> (DF) (ttl 43, id 57164, len 60, bad cksum 3e02!)
0x0000 4500 003c df4c 4000 2b06 3e02 421c 64ce E..<.L@.+.>.B.d.
0x0010 20f5 5775 87f6 1f90 56cf 4d07 0000 0000 ..Wu....V.M.....
0x0020 a002 16d0 20d8 0000 0204 05b4 0402 080a ....
0x0030 0049 924b 0000 0000 0103 0300 .I.K.....

13:10:08.156507 66.28.100.206.34806 > 32.245.87.117.8080: S [bad tcp cksum 1913!]
1456426247:1456426247(0) win 5840 <mss 1460,sackOK,timestamp 4821879
0,nop,wscale 0> (DF) (ttl 43, id 57165, len 60, bad cksum 3e01!)
0x0000 4500 003c df4d 4000 2b06 3e01 421c 64ce E..<.M@.+.>.B.d.
0x0010 20f5 5775 87f6 1f90 56cf 4d07 0000 0000 ..Wu....V.M.....
0x0020 a002 16d0 1fac 0000 0204 05b4 0402 080a ....
0x0030 0049 9377 0000 0000 0103 0300 .I.w.....

4. Description of the attack

The source IP address is scanning for port 1080/3128/8080 which are Socks server and Squid server respectively. The reason that someone would scan for these is in the hope of finding Socks or Squid servers which will accept their connection attempts to it. In essence a misconfigured server. This is done in the hope that the scanner will be able to do some anonymous surfing. This means that the person will be able to surf to sites and the IP address showing will not be theirs but that of the Socks server or of the Squid. The attacker in this case IP address 66.28.100.206 scanned the Class A address of 32.245.87.* The entire last octet was not scanned, however good portions of it were. This is a relatively simple scan to do as there are a variety of tools that have been written for just this purpose. As seen below the IP belongs to the following. There were no CVE's that dealt specifically with a proxy or squid scan so none were quoted.

OrgName: Cogent Communications
OrgID: COGC
Address: 1015 31st Street, NW
City: Washington
StateProv: DC
PostalCode: 20007
Country: US

NetRange: 66.28.0.0 - 66.28.255.255
CIDR: 66.28.0.0/16
NetName: COGENT-NB-0000
NetHandle: NET-66-28-0-0-1
Parent: NET-66-0-0-0-0
NetType: Direct Allocation
NameServer: AUTH1.DNS.COGENTCO.COM
NameServer: AUTH2.DNS.COGENTCO.COM
Comment: ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
Comment: Reassignment information for this block can be found at
Comment: rwhois.cogentco.com 4321
RegDate: 2000-10-12
Updated: 2001-12-05

5. Attack mechanism

There are a number of tools that can be used to do this type of scan. Notably among them is NMap[3]. There are other tools for Windows that have been coded for this type of scan. There are no "tells" or signatures attached to this scan that would give away what type of tools was used.

6. Correlations

These types of scans have been going on for some time and are still ongoing. Some earlier correlations to this type of activity are;

http://mix.twistedpair.ca/pipermail/inet-access/2001-April/000112.html
http://www.dshield.org/pipermail/list/2003-January/006824.php

These types as noted are very common on the web today. This is due to the fact in part that there are so many misconfigured servers out there today. Some sites dedicate sections to proxy/squid servers that are found to be open and accepting all connections.

7. Evidence of active targeting

There is definite active targeting here in both specific port numbers as well as a specific subnet of a class A ip address. The targeting is evident in the trace, a part is included below;

```
13:10:05.156507 66.28.100.206.34806 > 32.245.87.117.8080: S [bad tcp cksum 1913!]
1456426247:1456426247(0) win 5840 <mss 1460,sackOK,timestamp 4821579
0,nop,wscale 0> (DF) (ttl 43, id 57164, len 60, bad cksum 3e02!)
0x0000          4500 003c df4c 4000 2b06 3e02 421c 64ce      E..<.L@.+.>.B.d.
0x0010          20f5 5775 87f6 1f90 56cf 4d07 0000 0000   .  .Wu....V.M.....
0x0020          a002 16d0 20d8 0000 0204 05b4 0402 080a      ................
0x0030          0049 924b 0000 0000 0103 0300                .I.K........

13:10:08.156507 66.28.100.206.34806 > 32.245.87.117.8080: S [bad tcp cksum 1913!]
1456426247:1456426247(0) win 5840 <mss 1460,sackOK,timestamp 4821879
0,nop,wscale 0> (DF) (ttl 43, id 57165, len 60, bad cksum 3e01!)
0x0000          4500 003c df4d 4000 2b06 3e01 421c 64ce      E..<.M@.+.>.B.d.
0x0010          20f5 5775 87f6 1f90 56cf 4d07 0000 0000   ..  Wu....V.M.....
0x0020          a002 16d0 1fac 0000 0204 05b4 0402 080a       ................
0x0030          0049 9377 0000 0000 0103 0300                .I.w........

13:10:08.156507 66.28.100.206.36786 > 32.245.87.117.3128: S [bad tcp cksum 1913!]
1466005935:1466005935(0) win 5840 <mss 1460,sackOK,timestamp 4821879
0,nop,wscale 0> (DF) (ttl 43, id 60120, len 60, bad cksum 3276!)
0x0000          4500 003c ead8 4000 2b06 3276 421c 64ce      E..<..@.+.2vB.d.
0x0010          20f5 5775 8fb2 0c38 5761 79af 0000 0000       ..Wu...8Way.....
0x0020          a002 16d0 fe0d 0000 0204 05b4 0402 080a      ................
0x0030          0049 9377 0000 0000 0103 0300                .I.w........

13:10:11.156507 66.28.100.206.36786 > 32.245.87.117.3128: S [bad tcp cksum 1913!]
1466005935:1466005935(0) win 5840 <mss 1460,sackOK,timestamp 4822179
0,nop,wscale 0> (DF) (ttl 43, id 60121, len 60, bad cksum 3275!)
```

```
0x0000          4500 003c ead9 4000 2b06 3275 421c 64ce     E..<..@.+.2uB.d.
0x0010          20f5 5775 8fb2 0c38 5761 79af 0000 0000     ..Wu...8Way.....
0x0020          a002 16d0 fce1 0000 0204 05b4 0402 080a     ................
0x0030          0049 94a3 0000 0000 0103 0300               .I..........

13:10:14.156507 66.28.100.206.40797 > 32.245.87.118.8080: S [bad tcp cksum 1913!]
1462772562:1462772562(0) win 5840 <mss 1460,sackOK,timestamp 4822479
0,nop,wscale 0> (DF) (ttl 43, id 46749, len 60, bad cksum 66b0!)
0x0000          4500 003c b69d 4000 2b06 66b0 421c 64ce     E..<..@.+.f.B.d.
0x0010          20f5 5776 9f5d 1f90 5730 2352 0000 0000     ..Wv.]..W0#R....
0x0020          a002 16d0 2f40 0000 0204 05b4 0402 080a     ..../@..........
0x0030          0049 95cf 0000 0000 0103 0300               .I..........

13:10:17.156507 66.28.100.206.40797 > 32.245.87.118.8080: S [bad tcp cksum 1913!]
1462772562:1462772562(0) win 5840 <mss 1460,sackOK,timestamp 4822779
0,nop,wscale 0> (DF) (ttl 43, id 46750, len 60, bad cksum 66af!)
0x0000          4500 003c b69e 4000 2b06 66af 421c 64ce     E..<..@.+.f.B.d.
0x0010          20f5 5776 9f5d 1f90 5730 2352 0000 0000     ..Wv.]..W0#R....
0x0020          a002 16d0 2e14 0000 0204 05b4 0402 080a     ................
0x0030          0049 96fb 0000 0000 0103 0300               .I..........

13:10:17.156507 66.28.100.206.42789 > 32.245.87.118.3128: S [bad tcp cksum 1913!]
1466989426:1466989426(0) win 5840 <mss 1460,sackOK,timestamp 4822779
0,nop,wscale 0> (DF) (ttl 43, id 60809, len 60, bad cksum 2fc4!)
0x0000          4500 003c ed89 4000 2b06 2fc4 421c 64ce     E..<..@.+./.B.d.
0x0010          20f5 5776 a725 0c38 5770 7b72 0000 0000     ..Wv.%.8Wp{r....
0x0020          a002 16d0 e143 0000 0204 05b4 0402 080a     .....C..........
0x0030          0049 96fb 0000 0000 0103 0300               .I..........

13:10:20.156507 66.28.100.206.42789 > 32.245.87.118.3128: S [bad tcp cksum 1913!]
1466989426:1466989426(0) win 5840 <mss 1460,sackOK,timestamp 4823079
0,nop,wscale 0> (DF) (ttl 43, id 60810, len 60, bad cksum 2fc3!)
0x0000          4500 003c ed8a 4000 2b06 2fc3 421c 64ce     E..<..@.+./.B.d.
0x0010          20f5 5776 a725 0c38 5770 7b72 0000 0000     ..Wv.%.8Wp{r....
0x0020          a002 16d0 e017 0000 0204 05b4 0402 080a     ................
0x0030          0049 9827 0000 0000 0103 0300               .I.'........
```

8. Severity = (criticality + lethality) – (system countermeasures + network countermeasures)

criticality = 1 There does not appear to be any responses back from the scanned machines however we cannot be certain due to imcomplete logs and unknow network topology.

lethality = 1 If there were machines which allowed foreign connections this would result a minor degradation bandwith alone, and nothing further.

system countermeasure = 3 There does not appear to be any responses which means that

the proxy/squid servers have been properly configured. Assuming there were any listening there as well. However as stated we do not have complete logs, or a network topology to work with.

network countermeasures = 3 This is again a three for the reason that there were no responses. This is as it should be. Once again though incomplete logs and unknown topology.

Severity = +4

9. Defensive recommendation

Defensive recommendatios would be that all assetts such as squid/proxy server be behind the router, and if not then in a DMZ. Secondly that the servers are properly configure to allow only internal connections.

10. Multiple choice question

Q: What ports are the squid and proxy servers normally associated with?

a) 8080 and 3128
b) 8080 and 1080
c) 3128 and 80
d) 80 and 8080

A: (a) The answer is (a) the squid and proxy ports are normally associated with these ports.

References

1) http://www.sans.org/rr/audit/hping2.php
2) http://www.hping.org/
3) http://www.insecure.org/nmap/

Questions from www.incidents.org posting on 31 May 2003

The below noted questions were asked by "Smith, Donald" <Donald.Smith@qwest.com>

1)
>Can you dump the packets WITH ethernet address information?
>That might give you a clue about the network.

Yes the packets can be dumped with ethernet address information, and by ethernet I assume you mean layer 2 information ie: mac addresses. This would help ascertain what are routers and what are individual workstations yes.

2)
>What is the pattern? Two very simular packets against 8080 then
>two very simular packets towards 3128. Take a closer look at the ip id
>and the seq/ack.

The ip id numbers are increasing as per the normal Win16/32 based tcp/ip stacks ie: 1 up. Older releases of linux had this problem as well. The sequence number is normal as well. Yes it has repeated but that is normal behavior when a external computer tries to access a service without recieving the syn/ack back. It will try once again with the same tcp sequence number. This is expected behavior and is not anomolous in and of itself.

3)
>Why would a scanning tool send two syn's against the same port on the
>same system?

That is unknown and impossible to verify quite simply put. Who knows why the creator of this scanning tool wrote it this way. Could simply be poor coding on their part.

4)
.>I dont see the 1080 packets. Did you mean 8080?

There were two instances of the dst port of 1080 showing up but I meant port 8080 yes, my mistake

Part 2: Network Detects cont'd (3rd detect)

1. Source of trace

This trace was taken from the 2002.9.9 binary log file at the following url; http://www.incidents.org/logs/Raw/ There is no information available about the network itself or it's topology.There is no information either as to the placement of the IDS system. Since the checksums are all invalid, it's assumed that all of the IP's in the trace have been changed, and as such will be used as is without further obfuscation.

2. Detect was generated by

This detect was generated by Snort Version 1.8.4 (Build 99)
By Martin Roesch (roesch@sourcefire.com, www.snort.org)
The standard rule set that comes with this build of Snort was used with no additions to it.
The Snort rule that triggered the alert that I will analyze is;

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-IIS ISAPI .ida attempt"; flow:to_server,established; uricontent:".ida?"; nocase; reference:arachnids,552; classtype:web-application-attack; reference:bugtraq,1065; reference:cve,CAN-2000-0071; sid:1243; rev:8;)

3. Probability that source IP was spoofed

The probability the the source IP was spoofed is extremely low due to the nature of the attack being attempted. The attacker here is attempting a buffer overflow and would require the end result of his attempt, ie: GET default.ida file, to come back to him if successful. By that I mean he would expect to receive that file. So spoofing his address would pretty much negate the attack itself.

4. Description of the attack

This attack is a result of Code Red II worm[1] doing it's work. How the Code Red worm works is that a GET[2] command is used to send a buffer overflow in which the desired end state is to exploit the Index Service dll exploit made public on 18 Jun 2001. The targeted dll which is the aim of the exploit is the idq.dll. This file will allow ISAPI extensions to access administrative scripts ie: .ida files and internet data queries ie: .idq files. This specific attack targets the default.ida file which if vulnerable will allow the attacker to gain control of the system. For further details see the below noted CVE's, as well as the APNIC search performed below.
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0500
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0506

inetnum: 203.248.0.0 - 203.255.255.255
netname: KRNIC-KR
descr: KRNIC
descr: Korea Network Information Center
country: KR
admin-c: HM127-AP
tech-c: HM127-AP
remarks: **
remarks: KRNIC is the National Internet Registry
remarks: in Korea under APNIC. If you would like to
remarks: find assignment information in detail
remarks: please refer to the KRNIC Whois DB
remarks: http://whois.nic.or.kr/english/index.html
remarks: **
mnt-by: APNIC-HM
mnt-lower: MNT-KRNIC-AP
changed: hostmast@rs.krnic.net 19981015
changed: hostmaster@apnic.net 20010606
status: ALLOCATED PORTABLE

source: APNIC
person: Host Master
address: 11F, KTF B/D, 1321-11, Seocho2-Dong, Seocho-Gu,
address: Seoul, Korea, 137-857
country: KR
phone: +82-2-2186-4500
fax-no: +82-2-2186-4496
e-mail: hostmaster@nic.or.kr
nic-hdl: HM127-AP
mnt-by: MNT-KRNIC-AP
changed: hostmaster@nic.or.kr 20020507
source: APNIC
inetnum: 203.252.128.0 - 203.252.191.255
netname: KONKUKNET-KR
descr: Konkuk University
descr: 1 Hwayang-dong Kwangjin-gu
descr: SEOUL
descr: 143-130
country: KR
admin-c: IL47-KR
tech-c: DH91-KR
remarks: This IP address space has been allocated to KRNIC.
remarks: For more information, using KRNIC Whois Database
remarks: whois -h whois.nic.or.kr
mnt-by: MNT-KRNIC-AP
remarks: This information has been partially mirrored by APNIC from
remarks: KRNIC. To obtain more specific information, please use the
remarks: KRNIC whois server at whois.krnic.net.
changed: hostmaster@nic.or.kr 20030414
source: KRNIC

5. Attack mechanism

What the attacker does is use a HTTP GET command to send his buffer overflow to the targeted system. The command to be executed on the remote IIS server is the GET default.ida file. The buffer fill sent across using the GET command is used to overwrite the buffer size thereby allowing the malicious code to be executed ie: GET default.ida

18:17:26.256507 203.252.131.140.1569 > 32.245.166.119.80: P 2053849798:2053851262(1464) ack 468932805 win 32120 [tos 0x10] (ttl 240, id 0, len 1504, bad cksum 0!)

```
0x0000   4510 05e0 0000 0000 f006 0000 cbfc 838c  E...............
0x0010   20f5 a677 0621 0050 7a6b 42c6 1bf3 58c5  ...w.!.PzkB...X.
0x0020   5018 7d78 0000 0000 4745 5420 2f64 6566  P.}x....GET./def
0x0030   6175 6c74 2e69 6461 3f4e 4e4e 4e4e 4e4e  ault.ida?NNNNNNN
0x0040   4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e  NNNNNNNNNNNNNNNN
0x0050   4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e  NNNNNNNNNNNNNNNN
0x0060   4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e  NNNNNNNNNNNNNNNN
0x0070   4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e  NNNNNNNNNNNNNNNN
0x0080   4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e  NNNNNNNNNNNNNNNN
0x0090   4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e  NNNNNNNNNNNNNNNN
0x00a0   4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e  NNNNNNNNNNNNNNNN
0x00b0   4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e  NNNNNNNNNNNNNNNN
0x00c0   4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e  NNNNNNNNNNNNNNNN
0x00d0   4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e  NNNNNNNNNNNNNNNN
0x00e0   4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e  NNNNNNNNNNNNNNNN
0x00f0   4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e  NNNNNNNNNNNNNNNN
0x0100   4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e 4e4e  NNNNNNNNNNNNNNNN
0x0110   4e4e 4e4e 4e4e 4e4e 4e00 0000 0000 0000  NNNNNNNNN.......
0x0120   0000 0000 0000 0000 c303 0000 0078 00fa  .............x..
0x0130   2025 7539 3039 3025 7536 3835 3825 7563  .%u9090%u6858%uc
0x0140   6264 3325 7537 3830 3125 7539 3039 3025  bd3%u7801%u9090%
0x0150   7536 3835 3825 7563 6264 3325 7537 3830  u6858%ucbd3%u780
0x0160   3125 7539 3039 3025 7539 3039 3025 7538  1%u9090%u9090%u8
0x0170   3139 3025 7530 3063 3325 7530 3030 3325  190%u00c3%u0003%
0x0180   7538 6230 3025 7535 3331 6225 7535 3366  u8b00%u531b%u53f
0x0190   6625 7530 3037 3825 7530 3030 3025 7530  f%u0078%u0000%u0
0x01a0   303d 6120 2048 5454 502f 312e 300d 0a43  0=a..HTTP/1.0..C
0x01b0   6f6e 7465 6e74 2d74 7970 653a 2074 6578  ontent-type:.tex
0x01c0   742f 786d 6c0a 484f 5354 3a77 7777 2e77  t/xml.HOST:www.w
0x01d0   6f72 6d2e 636f 6d0a 2041 6363 6570 743a  orm.com..Accept:
0x01e0   202a 2f2a 0a43 6f6e 7465 6e74 2d6c 656e   */*.Content-len
0x01f0    6774 683a 2033 3536 3920 0d0a 0d0a 558b  gth:.3569.....U.
0x0200   ec81 ec18 0200 0053 5657 8dbd e8fd ffff  .......SVW......
```

6. Correlations

This worm was first directed at the www.whitehouse.gov website. It has since targeted all machines, however it is only destructive to those running Microsoft's IIS webserver. These attacks have been noted and talked about in many forums and sites, notably;
http://www.sanctuminc.com/news/alerts/2001/20010801_codered.html
http://www.unixwiz.net/techtips/CodeRedII.html
http://www.cert.org/incident_notes/IN-2001-09.html

7. Evidence of active targeting

The Code Red II worm propagates itself by generating it's own random IP addresses. This is part of the worm itself[3] , ie: it's source code, the ability to generate random IP addresses in which it will try in turn to infect. It is a rather insidious method of propagation. The Code Red II variant improved itself over the initial Code Red in it's ability[4] to "intelligently" generate random IP's that may be hosting IIS servers.
That being said though these are still randomly generated IP's which as randomly infers it is still firing out packets willy nilly in the hopes of finding another IIS server with a vulnerable default or unpatched installation. To sum up it is in a sense active targeting in that it is trying to infect only IIS servers.

8. Severity = (criticality + lethality) – (system countermeasures + network countermeasures)

criticality = 3 The IIS server is generally used to advertise a web presence for a company and or site. A Code Red infection could result in a loss of bandwidth and therefore adversely affect the site or company.
lethality = 5 Were the IIS server successfully compromised then the attack has gained access to system resources, and fully compromised the machine.
system countermeasures = 2 It does not appear to have answered back, however we can not be certain as full logs would be required.
network countermeasures = 2 We are again uncertain as to whether or not an answer was received due to the incomplete logs, and unknown network topology, and or patch level for the IIS server.

9. Defensive recommendation

The best defence is to make sure that if you are running IIS servers is that they are fully patched. Secondly if you are running these servers ensure that they are placed in a DMZ[5] if at all possible.

10. Multiple choice question

Q: What server type does the Code Red worm attack?

a) Apache
b) IIS
c) Samba
d) VAX/VMS

A: (b) The answer is b IIS. Code Red was written specifically for Microsoft IIS web servers.

References

1) http://www.thesitewizard.com/news/coderediiworm.shtml
2) http://www.sanctuminc.com/news/alerts/2001/20010801_codered.html

3) http://www.eeye.com/html/Research/Advisories/AL20010804.html
4) http://www.eeye.com/html/Research/Advisories/AL20010804.html
5) http://www.simonzone.com/software/guarddog/manual2/tutorial-zones.html

3.0 Analyze This

3.1 Executive Summary

The following analysis is based on 5 days worth of traffic downloaded from the GIAC website. These 5 days of traffic were broken up into three distinct files. Those being Alert, Scans, and Out Of Spec files. Due to the volume of traffic generated only the top 10 talkers from the Alert files were analyzed, and where possible correlated with the Scan, as well as the Out of Spec files. The top 10 types of Scans as well were looked at to look for any potential trends, and lastly the Out of Spec files were perused as well for any type of correlation possible with the other two afore-mentioned file types.
It was noted that there were some questionable practices on the MY.NET network. Namely the use of file sharing software like WinMX, and the presence of online game servers such as Half Life. These practices are undesirable for several reasons which will be expanded upon in this document.

3.2 File Selection

The files analyzed were from 20-24 Apr 2003. These files were downloaded from the following url; http://www.incidents.org/logs/ as per the instructions for this assignment.

Analysis was performed on the below mentioned file;

alert.030416.gz	Apr20	scans.030416.gz	Apr20	OOS Report 2003 04 20 1651
alert.030417.gz	Apr21	scans.030417.gz	Apr21	OOS Report 2003 04 21 3207
alert.030418.gz	Apr22	scans.030418.gz	Apr22	OOS Report 2003 04 22 9834
alert.030419.gz	Apr23	scans.030419.gz	Apr23	OOS Report 2003 04 23 3063
alert.030420.gz	Apr24	scans.030420.gz	Apr24	OOS Report 2003 04 24 1402

3.3 Methodology

The analysis was done using both Win32 and Linux applications. The vast majority of the analysis however was done using Linux applications. The files were downloaded and then amalgamated into one large file for each type of file using the cat command. These files (1 file now for each type of ie: 1 large file each for Alert/Scan/OOS) were then put through snortsnarf. It was noticed at this time that snortsnarf had problems with the obfuscation used in some of the files ie: MY.NET.101.62 The files at this point had the MY.NET removed and replaced with 10.10 using the sed command. There were problems with the OOS files in snortsnarf and the perl script written by Ricky Smith was able to rectify these problems, and allow the processing of said files by snortsnarf. Many

thanks to Rick Smith for his script! It should be noted as well that snortsnarf is a Perl based program and therefore will require that Perl be installed on your computer. Per the administrivia for the Part III it was decided by myself that after review of the logs that the top ten in terms of alarms generated would be looked at seen as no specific alarm or IP stood out from the crowd. Doing it this way would fulfill the detects requirement as well as the Top Ten requirement.

cat usage as follows

cat alert.030416 alert030417 alert.030418 alert.030419 alert.030420 > alert_all

sed usage as follows

sed –e "s/MY.NET/10.10/" alert_all > alert_final
snortsnarf usage as follows

./snortsnarf.pl –rs file_name

3.4 Top 10 Talkers Alert files

Signature (click for sig info)	# Alerts	# Sources	# Dests
SMB Name Wildcard	129879	23787	38169
Incomplete Packet Fragments Discarded	45619	82	83
XDCC client detected attempting to IRC	30299	12	21
High port 65535 tcp - possible Red Worm - traffic	18518	133	132
spp_http_decode: IIS Unicode attack detected	18098	874	1055
CS WEBSERVER - external web traffic	13007	5173	1
EXPLOIT identd overflow	12151	1	1
spp_http_decode: CGI Null Byte attack detected	5726	178	122
Tiny Fragments - Possible Hostile Activity	8424	15	319
TFTP - External TCP connection to internal tftp server	8091	3	3

The above noted are the Top Ten talkers for the Alert files that were downloaded. This was after the combined alert file was processed through snortsnarf. The Top Ten alerts will be analyzed in the sequence that they appear on the above noted chart. It should be noted that these are the top ten in the sense of the number of alarms generated.

* Please note that none of the above statistics will be quoted in the below noted analysis*
* Please simply refer back to the above noted chart for a breakdown on alarm statistics*

3.4.1 SMB Name Wildcard

The SMB Name Wildcard alarm is tripped when an external host is probing port 137 (NetBIOS) in an attempt to ascertain what file shares and IP addresses may be available on that machine. This can be automated through the use of scripts and or programs written expressly to look for open shares on Windows machines. Programs such as Nbtdump do an excellent job of collecting such information. If successful an intruder can gain valuable information such as shares, usernames, and possibly the password policy to name a few possible sources of information which are of critical value.

```
04/20-11:34:14.522214 [**] SMB Name Wildcard [**] 194.148.17.27:3299 ->
10.10.1.0:137
04/20-11:34:19.010271 [**] SMB Name Wildcard [**] 194.148.17.27:3338 ->
10.10.1.39:137
04/20-11:34:19.470124 [**] SMB Name Wildcard [**] 194.148.17.27:3342 ->
10.10.1.43:137
04/20-11:34:22.439842 [**] SMB Name Wildcard [**] 194.148.17.27:3369 ->
10.10.1.70:137
04/20-11:34:23.319734 [**] SMB Name Wildcard [**] 194.148.17.27:3377 ->
10.10.1.78:137
```

Recommendation

The easiest way to defeat this type of enumeration is to simply block off the port at the gateway router, and also use a firewall. This would also result in far less white noise for the IDS to handle, and thereby help the network security analyst focus on more worrisome traffic.

3.4.2 Incomplete Packet Fragments Discarded

The Incomplete Packets Fragments Discarded alarm is tripped when the IDS processes a packet that has been fragmented. Packet fragmentation is normally associated with the attempts of hackers to penetrate poorly configured routers and to try and circumvent firewall rulesets. These days though all modern firewalls will drop fragmented packets outright. Dependent on your router configuration this may or may not be the case as well. There are some benign causes for fragmentation as well such as poorly configured computers for one ie: someone has tweaked their registry settings such as the maximum segment size. If someone has increased the size of the maximum segment size to beyond what the maximum transmission unit is for their transmission medium then it will fragment all the traffic going out. For example the MTU for ethernet is 1540, and in turn the maximum segment size is normally 1500. You have to save 40 bytes for the packet header and transmission information. So if the maximum segment size is too large the packets will always fragment, and performance can easily become and issue.

The single highest offender in this alarm category was MY.NET.252.166 This IP address was sending fragments for over a half hour to 213.236.6.254 which resolved to an IP range located in Spain.

```
04/24-13:32:45.361551 [**] Incomplete Packet Fragments Discarded [**]
10.10.252.166:0 -> 213.236.6.254:0
04/24-13:32:46.122914 [**] Incomplete Packet Fragments Discarded [**]
10.10.252.166:0 -> 213.236.6.254:0
04/24-13:32:46.723540 [**] Incomplete Packet Fragments Discarded [**]
10.10.252.166:0 -> 213.236.6.254:0
04/24-13:32:47.204315 [**] Incomplete Packet Fragments Discarded [**]
10.10.252.166:0 -> 213.236.6.254:0
04/24-13:32:48.944615 [**] Incomplete Packet Fragments Discarded [**]
10.10.252.166:0 -> 213.236.6.254:0
```

```
inetnum:        213.236.6.0 - 213.236.7.255
netname:        INTERBOOK-NET
descr:          VIA NET.WORKS Spain
country:        ES
admin-c:        EM49-RIPE
tech-c:         PC2637-RIPE
tech-c:         AG265-RIPE
tech-c:         PP214-RIPE
status:         ASSIGNED PA
mnt-by:         IBOOK-RIPE-MNT
mnt-lower:      IBOOK-RIPE-MNT
changed:        pcuevas@vianetworks.es 20011127
source:         RIPE
```

It is unknown why the MY.NET user was sending fragments to the above noted IP for over a half hour and this bears further scrutiny by the local security analysts due to the limited files that were supplied.
The remaining IP addresses that were noted covered a wide range which were all noted sending inbound packet fragments to the MY.NET network and can likely be attributed to scanning attempts. There were no external IP addresses of note that were logged to further expand upon.

Recommendation

The simplest way to get rid of all scanning attempts associated with this type of activity is to once again block fragmented packets at the border gateway router. Unless the MY.NET network has a specific need to accept fragmented packets then they should simply be discarded in the most efficient manner possible as indicated above. This will also save in bandwidth and the white noise generated on the IDS. Again this helps simplify the security analysts job by being able to focus on the more troubling alarms.

3.4.3 XDCC client detected attempting to IRC

This alarm is triggered when someone is attempting to access an IRC server. IRC is generally associated with ports such 6667, 7000 and the such. All alarms that were tripped originated from the MY.NET network and as such were outbound. This may seem to benign however it is a poor security practice to allow such activity. One reason being that many trojans communicate back to their server by means of IRC and ICQ. By allowing this type of activity through then a potentially infected machine now has a open unmonitored port to report "home to". Not a very good idea for obvious reasons. There is also the rising problem of Botnet's and the potential embarrassment caused to a corporation should they be implicated in such activity as botnet DoS attacks. As mentioned all the offenders for this alarm were internal users.

```
04/21-09:26:29.919080 [**] [UMBC NIDS IRC Alert] XDCC client detected
attempting to IRC [**] 10.10.198.221:1026 -> 205.188.149.12:6667
04/21-09:28:54.945825 [**] [UMBC NIDS IRC Alert] XDCC client detected
attempting to IRC [**] 10.10.198.221:2585 -> 205.188.149.12:6667
04/21-09:28:58.639445 [**] [UMBC NIDS IRC Alert] XDCC client detected
attempting to IRC [**] 10.10.198.221:2726 -> 205.188.149.12:6667
04/21-09:28:58.935477 [**] [UMBC NIDS IRC Alert] XDCC client detected
attempting to IRC [**] 10.10.198.221:2751 -> 205.188.149.12:6667
04/21-09:29:01.842323 [**] [UMBC NIDS IRC Alert] XDCC client detected
attempting to IRC [**] 10.10.198.221:3000 -> 205.188.149.12:6667
```

Recommendation

The simplest solution once again is to block these ports at the border gateway router, as well as to limit the installation of software on internal network computers to the system administrator. No attempt was made to look for trojans calling home via IRC means.

3.4.4 High port 65535 tcp - possible Red Worm – traffic

This alarm is generated by communications emanating on port 65535 either as a source port or destination port. Code Red is a worm which targets IIS servers and attempts to exploit a buffer overflow condition present in a default install. There have been some variants upon the original Code Red worm since it's initial discovery. There is an equal amount of alarms generated from both external and internal sources. Due to the limitations of the logs analyzed I highly suggest that all MY.NET (10.10.xxx.xxx) addresses running IIS software be checked for possible infection.

```
04/20-00:19:23.284809 [**] High port 65535 udp - possible Red Worm -
traffic [**] 10.10.207.230:6257 -> 219.241.24.159:65535
04/20-00:25:38.332823 [**] High port 65535 udp - possible Red Worm -
traffic [**] 10.10.207.230:6257 -> 24.168.194.55:65535
04/20-00:42:38.151386 [**] High port 65535 udp - possible Red Worm -
traffic [**] 10.10.207.230:6257 -> 24.193.35.243:65535
04/20-01:48:48.335322 [**] High port 65535 udp - possible Red Worm -
traffic [**] 10.10.207.230:6257 -> 61.22.12.45:65535
```

04/20-03:36:23.115843 [**] High port 65535 udp - possible Red Worm - traffic [**] 10.10.207.230:6257 -> 24.194.50.162:65535

Recommendation

As well to verify that all vendor patches be applied religiously, and lastly download the latest DAT files from your anti-virus vendor. As a final note it is unlikely that any of these machines are infected due to them transmitting on a port other then 65535. That being said it is best to be safe then sorry.

3.4.5 spp_http decode: IIS Unicode attack detected

This alarm is triggered when Unicode characters / and \ and .. are detected. The problem with this alarm is that it is very often seen in normal http traffic. There is a unicode exploit though that can be used to exploit Microsoft's IIS ver 4 and ver 5. What the exploit does in essence is allow an attacker who sends a specially crafted url string to gain access to files and folders he normally would not have access to.

04/21-08:51:49.610654 [**] spp http decode: IIS Unicode attack detected [**] 10.10.91.101:1304 -> 211.32.117.201:80
04/21-08:51:49.610654 [**] spp http decode: IIS Unicode attack detected [**] 10.10.91.101:1304 -> 211.32.117.201:80
04/21-08:51:49.610654 [**] spp http decode: IIS Unicode attack detected [**] 10.10.91.101:1304 -> 211.32.117.201:80
04/21-08:51:49.610654 [**] spp http decode: IIS Unicode attack detected [**] 10.10.91.101:1304 -> 211.32.117.201:80
04/21-08:51:49.610654 [**] spp http decode: IIS Unicode attack detected [**] 10.10.91.101:1304 -> 211.32.117.201:80

Recommendation

The only realistic defence against this type of attack is to ensure that all patches are applied, and that your anti-virus software updates are regularly downloaded. If at all possible these web serves should also be located in a dmz.

3.4.6 CS WEBSERVER - external web traffic

This alarm was triggered when an external IP address connected to a MY.NET network web server. It is highly probably that the vast majority of these alarms are false positives, however without complete logs as mentioned it is impossible to say with certainty.

04/20-00:16:39.339971 [**] CS WEBSERVER - external web traffic [**] 66.77.73.236:1806 -> 10.10.100.165:80
04/20-00:22:08.838752 [**] CS WEBSERVER - external web traffic [**] 66.77.73.236:2559 -> 10.10.100.165:80

```
04/20-00:33:20.474071 [**] CS WEBSERVER - external web traffic [**]
66.77.73.236:3210 -> 10.10.100.165:80
04/20-00:39:25.972564 [**] CS WEBSERVER - external web traffic [**]
66.77.73.236:1423 -> 10.10.100.165:80
04/20-00:39:46.370788 [**] CS WEBSERVER - external web traffic [**]
66.77.73.236:1765 -> 10.10.100.165:80
```

Recommendation

The running of web servers is very much a common practice. The simplest solution to securing these web servers is to ensure that all patches are applied, and that the web server itself be placed in a dmz in case it is indeed compromised.

3.4.7 EXPLOIT identd overflow

The identd overflow vulnerability is a cross platform vulnerability which affects both the Win32 platform and linux architecture. It is unclear as to whether or not any machines are even running this service on the MY.NET network. Should there be any running this service then all the machines noted in this alert file should be verified for possible signs of compromise. There was only one External IP sending to port 113 of again only one internal IP. This machine should be investigated. Interestingly enough the external IP belongs to a university which are generally known as hot beds of hacker activity.

```
04/22-18:02:49.157290 [**] EXPLOIT identd overflow [**]
128.252.249.68:3878 -> 10.10.205.118:113
04/22-18:02:49.157414 [**] EXPLOIT identd overflow [**]
128.252.249.68:3878 -> 10.10.205.118:113
04/22-18:02:51.502270 [**] EXPLOIT identd overflow [**]
128.252.249.68:3878 -> 10.10.205.118:113
04/22-18:02:51.885992 [**] EXPLOIT identd overflow [**]
128.252.249.68:3878 -> 10.10.205.118:113
04/22-18:02:52.142833 [**] EXPLOIT identd overflow [**]
128.252.249.68:3878 -> 10.10.205.118:113
```

```
OrgName:    Washington University
OrgID:      WASHIN
Address:    Network Technology Services
Address:    1 Brookings Drive, Campus Box 1048
City:       St. Louis
StateProv:  MO
PostalCode: 63130
Country:    US

NetRange:   128.252.0.0 - 128.252.255.255
CIDR:       128.252.0.0/16
NetName:    WASHINGTON-U
NetHandle:  NET-128-252-0-0-1
Parent:     NET-128-0-0-0-0
NetType:    Direct Assignment
```

```
NameServer: WUGATE.WUSTL.EDU
NameServer: WUARCHIVE.WUSTL.EDU
NameServer: ADMIN.STARNET.NET
NameServer: NEWS.STARNET.NET
Comment:
RegDate:    1987-06-03
Updated:    2000-04-25
```

Recommendation

Should there be any machines running the identd service then I would recommend that it be changed to ridentd so as to minimize the risk as much as possible.

3.4.8 spp_http_decode: CGI Null Byte attack detected

This alarm is triggered when the http decoding routine finds a `%00` in an http request. This alarm is prone to false positives due to cookies and ssl usage. Due to the limited logs it is not possible at this time to determine whether or not these are indeed false positives. The only to tell would be to look at the packet contents.

04/22-19:05:07.999922 [**] spp http decode: CGI Null Byte attack detected [**] 10.10.218.218:3757 -> 216.241.219.22:80
04/22-19:05:07.999922 [**] spp http decode: CGI Null Byte attack detected [**] 10.10.218.218:3757 -> 216.241.219.22:80
04/22-19:05:07.999922 [**] spp http decode: CGI Null Byte attack detected [**] 10.10.218.218:3757 -> 216.241.219.22:80
04/22-19:05:07.999922 [**] spp http decode: CGI Null Byte attack detected [**] 10.10.218.218:3757 -> 216.241.219.22:80
04/22-19:05:07.999922 [**] spp http decode: CGI Null Byte attack detected [**] 10.10.218.218:3757 -> 216.241.219.22:80

Recommendation

There is no defense to this attack that this analyst is aware of at this time beyond the usual patch maintenance on your web servers. If you are receiving a substantial amount of these alarms then the packet contents should then be inspected and the offending IP addresses then excluded the border gateway router. That being said all of the alarms that were triggered came from internal addresses going to external webservers. This brings down the severity of the alarm. Though this could indicate some internal users attempting to hack into external webservers which could prove embarassing to the network owners.

3.4.9 Tiny Fragments - Possible Hostile Activity

This alarm is fired when packets are received with the MF option set in the IP header. This may indicated that someone is trying to bypass a poorly configured firewall, and or loosely configured router. Though this attack is by and large ineffective nowadays as most modern firewalls drop these packets. Depending on your situation your router may or may not be configured to drop these as well.

```
04/21-05:07:35.171675 [**] Tiny Fragments - Possible Hostile Activity
[**] 142.161.28.130 -> 10.10.236.18
04/21-05:22:37.875573 [**] Tiny Fragments - Possible Hostile Activity
[**] 142.161.28.130 -> 10.10.236.18
04/21-06:07:44.074276 [**] Tiny Fragments - Possible Hostile Activity
[**] 142.161.28.130 -> 10.10.236.18
04/21-06:12:44.641156 [**] Tiny Fragments - Possible Hostile Activity
[**] 142.161.28.130 -> 10.10.236.18
04/21-06:12:45.687990 [**] Tiny Fragments - Possible Hostile Activity
[**] 142.161.28.130 -> 10.10.236.18
```

Recommendation

It is best to drop these packets at the earliest point which would normally be the border gateway router once again. Just insure that your router configuration is properly tuned. As a second line of defense have your internal firewalls set to drop fragmented packets as well. Lastly ensure that your IDS has a rule triggered to fire should any fragmented packets somehow make it through.

3.4.10 TFTP - External TCP connection to internal tftp server

This alarm triggers when an external TCP connection attempt to the TFTP port of 69 is detected. There should never be any external connections allowed to an internal TFTP server. Though TFTP works using UDP it is still very undesirable to allow any type to external connection to TFTP. Only two external sources were noted attempting to connect were noted. It is unlikely that they succeeded due to the attempt being TCP however the entire packet trace should be looked at to confirm this. The entire packet logs were not provided so I am unable to do so at this time.

```
04/23-02:45:09.498729 [**] TFTP - External TCP connection to internal
tftp server [**] 193.253.220.98:14132 -> 10.10.212.90:69
04/23-03:49:00.002310 [**] TFTP - External TCP connection to internal
tftp server [**] 193.253.220.98:14658 -> 10.10.212.90:69
04/23-03:49:07.579390 [**] TFTP - External TCP connection to internal
tftp server [**] 193.253.220.98:14658 -> 10.10.212.90:69
04/23-03:49:12.950622 [**] TFTP - External TCP connection to internal
tftp server [**] 193.253.220.98:14658 -> 10.10.212.90:69
04/23-03:49:17.591550 [**] TFTP - External TCP connection to internal
tftp server [**] 193.253.220.98:14658 -> 10.10.212.90:69
```

Recommendation

Access to port 69 should be blocked at the border gateway router. There is no need to have access enabled for this port. Seen as TFTP is normally used solely for router configuration backups, and saving IOS images access should be restricted to the machine hosting the TFTP service. It should also be firewalled off from the rest of the network if at all possible.

3.5 Top Ten Scanners by source IP

Rank	Total # Alerts	**Source IP**	# Signatures triggered	Destinations involved
rank #1	90414 alerts	**130.85.87.50**	2 signatures	(773 destination IPs)
rank #2	18548 alerts	**130.85.87.44**	1 signatures	(168 destination IPs)
rank #3	13393 alerts	**130.85.207.230**	2 signatures	(7347 destination IPs)
rank #4	12962 alerts	**146.164.34.42**	1 signatures	(10991 destination IPs)
rank #5	11323 alerts	**193.11.250.21**	2 signatures	(10258 destination IPs)
rank #6	9202 alerts	**217.70.4.246**	1 signatures	(8073 destination IPs)
rank #7	8837 alerts	**216.137.3.107**	1 signatures	(3781 destination IPs)
rank #8	8538 alerts	**130.85.1.3**	1 signatures	(3141 destination IPs)
rank #9	8480 alerts	**130.85.97.94**	1 signatures	(7965 destination IPs)
rank #10	7212 alerts	**81.56.209.187**	1 signatures	(6264 destination IPs)

The above noted chart is the output generated by snortsnarf after it had processed all 5 days worth of the scan logs which were downloaded. It should be noted once again that the above graph represents the Top Ten Scanners by source IP.

I will break down the ports scanned by each of the above noted, and see if this gives us any insight into the alert files we have already now looked at.

130.85.87.50
This IP address is being used as a Half-Life server. This conclusion was reached by the source port usage of 27022 which is a port associated with Half-Life game servers. As well the massive amount of udp packets seems to confirm this as well.

```
Apr 20 20:16:49 130.85.87.50:27022 -> 65.96.183.164:27005 UDP
Apr 20 20:16:49 130.85.87.50:27022 -> 24.58.195.45:27005 UDP
```

```
Apr 20 20:16:49 130.85.87.50:27022 -> 68.39.48.160:27005 UDP
Apr 20 20:16:50 130.85.87.50:27022 -> 68.61.159.89:27005 UDP
Apr 20 20:16:50 130.85.87.50:27022 -> 12.237.242.42:27005 UDP
```

130.85.87.44

This IP address is also being used as a Half-Life server. This one is using port 27021 vice 27022. Once again the sheer mass of udp packets would seem to confirm this.

```
Apr 21 20:32:08 130.85.87.44:27021 -> 65.73.232.252:27010 UDP
Apr 21 20:32:08 130.85.87.44:27021 -> 24.156.106.124:13571 UDP
Apr 21 20:32:08 130.85.87.44:27021 -> 12.208.227.9:7130 UDP
Apr 21 20:32:08 130.85.87.44:27021 -> 68.81.168.163:29411 UDP
Apr 21 20:32:08 130.85.87.44:27021 -> 68.59.5.162:28834 UDP
```

130.85.207.230

This IP address has WinMX installed on it and is being used for file sharing. This was deduced by the port usage of port 6257 a port well known for WinMX. Also the amount of chaotic IP address connections seems to confirm this as well.

```
Apr 20 05:10:31 130.85.207.230:6257 -> 43.235.144.205:8654 UDP
Apr 20 05:10:32 130.85.207.230:6257 -> 82.47.136.30:6257 UDP
Apr 20 05:10:32 130.85.207.230:6257 -> 80.180.5.108:6257 UDP
Apr 20 05:10:33 130.85.207.230:6257 -> 219.107.140.216:6257 UDP
Apr 20 05:10:33 130.85.207.230:6257 -> 219.1.86.107:6257 UDP
```

146.164.34.42

This external IP is scanning the MY.NET network for port 443 (HTTPS). This attacker is just doing reconnaissance to ascertain if any machines are offering this service. This is also evident in the way he scanned. He scanned the entire subnet.

```
Apr 20 08:23:25 146.164.34.42:40214 -> 130.85.7.36:443 SYN ******S*
Apr 20 08:23:25 146.164.34.42:40215 -> 130.85.7.37:443 SYN ******S*
Apr 20 08:23:25 146.164.34.42:40216 -> 130.85.7.38:443 SYN ******S*
Apr 20 08:23:25 146.164.34.42:39895 -> 130.85.5.254:443 SYN ******S*
Apr 20 08:23:25 146.164.34.42:40217 -> 130.85.7.39:443 SYN ******S*
```

193.11.250.21

This external IP address is also scanning the MY.NET network for port 443(HTTPS) as was the above noted IP. Same method, they scanned the entire subnet.

```
Apr 20 07:46:20 193.11.250.21:1370 -> 130.85.3.28:443 SYN ******S*
```

```
Apr 20 07:46:20 193.11.250.21:1376 -> 130.85.3.34:443 SYN ******S*
Apr 20 07:46:20 193.11.250.21:1629 -> 130.85.4.31:443 SYN ******S*
Apr 20 07:46:20 193.11.250.21:1630 -> 130.85.4.32:443 SYN ******S*
Apr 20 07:46:20 193.11.250.21:1381 -> 130.85.3.39:443 SYN ******S*
```

217.70.4.246
This external IP address was scanning the MY.NET network for open shares on port 139.

```
Apr 20 06:05:47 217.70.4.246:4858 -> 130.85.20.137:139 SYN ******S*
Apr 20 06:05:47 217.70.4.246:1217 -> 130.85.21.206:139 SYN ******S*
Apr 20 06:05:47 217.70.4.246:1220 -> 130.85.21.209:139 SYN ******S*
Apr 20 06:05:48 217.70.4.246:1231 -> 130.85.21.219:139 SYN ******S*
Apr 20 06:05:48 217.70.4.246:1237 -> 130.85.21.225:139 SYN ******S*
```

216.137.3.107
This external IP address was scanning the MY.NET network for open proxies. Specifically on ports 1080/3128/4588/6588/8080.

```
Apr 20 14:59:00 216.137.3.107:38598 -> 130.85.4.74:3128 SYN ******S*
Apr 20 14:59:00 216.137.3.107:46997 -> 130.85.4.76:8080 SYN ******S*
Apr 20 14:59:00 216.137.3.107:49322 -> 130.85.4.77:1080 SYN ******S*
Apr 20 14:59:00 216.137.3.107:1179 -> 130.85.4.77:4588 SYN ******S*
Apr 20 14:59:00 216.137.3.107:20343 -> 130.85.4.78:1080 SYN ******S*
```

130.85.1.3
This IP address appears to be a DNS server for the MY.NET network as evidenced by the number of queries it received and responded back to on port 53.

```
Apr 20 03:59:33 130.85.1.3:57312 -> 61.152.82.18:53 UDP
Apr 20 03:59:33 130.85.1.3:57312 -> 128.63.2.53:53 UDP
Apr 20 03:59:33 130.85.1.3:57312 -> 193.188.34.241:53 UDP
Apr 20 03:59:33 130.85.1.3:57312 -> 12.148.62.7:53 UDP
Apr 20 03:59:34 130.85.1.3:57312 -> 192.149.252.22:53 UDP
```

130.85.97.94
This IP address appears to be a web server due to the large amount of connections outbound to port 80. The large portion of port destinations checked were all legitimate sites which adds to this theory.

```
Apr 23 00:50:29 130.85.97.94:2456 -> 130.122.18.139:80 SYN ******S*
Apr 23 00:50:29 130.85.97.94:2470 -> 130.161.153.117:80 SYN ******S*
```

```
Apr 23 00:50:29 130.85.97.94:2492 -> 111.22.176.62:80 SYN ******S*
Apr 23 00:50:30 130.85.97.94:2502 -> 87.140.197.110:80 SYN ******S*
Apr 23 00:50:30 130.85.97.94:2507 -> 130.209.228.244:80 SYN ******S*
```

81.56.209.187
This external IP address was scanning the MY.NET network for open shares on port 139 and port 1433.

```
Apr 22 07:08:56 81.56.209.187:1768 -> 130.85.1.191:139 SYN ******S*
Apr 22 07:08:57 81.56.209.187:1805 -> 130.85.1.228:139 SYN ******S*
Apr 22 07:08:57 81.56.209.187:1809 -> 130.85.1.232:139 SYN ******S*
Apr 22 07:08:57 81.56.209.187:1810 -> 130.85.1.233:139 SYN ******S*
Apr 22 07:08:57 81.56.209.187:1814 -> 130.85.1.237:139 SYN ******S*
```

OOS Files

Of the below noted OOS source addresses the vast majority are due to Queso fingerprinting attempts and file sharing applicatioins. This type of activity was also noted by Johnny Calhoun analyst number 0600.

`****P***` Is normally associated with file sharing applications such as Kazaa and Morpheus.

`12****S*` This type of packet is associated with Queso fingerprinting attempts due to it's unusual flag combination.

Rank	Total # Alerts	**Source IP**	# Signatures triggered	Destinations involved
rank #1	247 alerts	**148.63.89.23**	1 signatures	10.10.219.82
rank #2	239 alerts	**210.253.206.147**	1 signatures	10.10.211.26
rank #3	217 alerts	**69.3.168.167**	1 signatures	10.10.211.26
rank #4	209 alerts	**68.54.93.181**	1 signatures	10.10.6.7
rank #5	157 alerts	**24.35.40.13**	1 signatures	(9 destination IPs)
rank #6	145 alerts	**216.95.201.25**	1 signatures	(6 destination IPs)
rank #7	144 alerts	**216.95.201.28**	1 signatures	(6 destination IPs)
rank #8	137 alerts	**66.140.25.157**	1 signatures	(11 destination IPs)
		216.95.201.31	1 signatures	(6 destination IPs)
rank #10	136 alerts	**216.95.201.30**	1 signatures	(8 destination IPs)

5 external IP addresses to monitor in future

1) 66.140.25.157 fm the OOS files
This IP address should be watched closely or just blocked at the gateway router due to the amount scanning for particular ports that they are doing. This is an obvious attempt at casing the ports on the below noted machine.

```
04/19-03:32:10.420610 [**] OOS [**] 66.140.25.157:43645 ->
10.10.60.38:23 TCP TTL:44 TOS:0x0 ID:14861 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x5FDF61C3 Ack: 0x0 Win: 0x16D0 TcpLen: 40 TCP Options (5)
=> MSS: 1460 SackOK TS: 1215651067 0 NOP WS: 0
04/19-03:32:10.427236 [**] OOS [**] 66.140.25.157:43649 ->
10.10.60.38:81 TCP TTL:44 TOS:0x0 ID:12894 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x5FC16992 Ack: 0x0 Win: 0x16D0 TcpLen: 40 TCP Options (5)
=> MSS: 1460 SackOK TS: 1215651067 0 NOP WS: 0
04/19-03:32:10.440292 [**] OOS [**] 66.140.25.157:43662 ->
10.10.60.38:4438 TCP TTL:44 TOS:0x0 ID:10520 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x5FBF72D8 Ack: 0x0 Win: 0x16D0 TcpLen: 40 TCP Options (5)
=> MSS: 1460 SackOK TS: 1215651067 0 NOP WS: 0
04/19-03:32:10.460498 [**] OOS [**] 66.140.25.157:43669 ->
10.10.60.38:7464 TCP TTL:44 TOS:0x0 ID:45936 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x5F97E8A0 Ack: 0x0 Win: 0x16D0 TcpLen: 40 TCP Options (5)
=> MSS: 1460 SackOK TS: 1215651067 0 NOP WS: 0
04/19-03:32:10.467132 [**] OOS [**] 66.140.25.157:43670 ->
10.10.60.38:7810 TCP TTL:44 TOS:0x0 ID:60498 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x5F74B79E Ack: 0x0 Win: 0x16D0 TcpLen: 40 TCP Options (5)
=> MSS: 1460 SackOK TS: 1215651067 0 NOP WS: 0
04/19-03:32:10.473579 [**] OOS [**] 66.140.25.157:43671 ->
10.10.60.38:8130 TCP TTL:44 TOS:0x0 ID:7316 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x5FFB475C Ack: 0x0 Win: 0x16D0 TcpLen: 40 TCP Options (5)
=> MSS: 1460 SackOK TS: 1215651067 0 NOP WS: 0
04/19-03:32:16.403093 [**] OOS [**] 66.140.25.157:43655 ->
10.10.60.38:8081 TCP TTL:44 TOS:0x0 ID:47595 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x5F88E226 Ack: 0x0 Win: 0x16D0 TcpLen: 40 TCP Options (5)
=> MSS: 1460 SackOK TS: 1215651667 0 NOP WS: 0
04/19-03:32:16.409875 [**] OOS [**] 66.140.25.157:43656 ->
10.10.60.38:81 TCP TTL:44 TOS:0x0 ID:24219 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x5F440556 Ack: 0x0 Win: 0x16D0 TcpLen: 40 TCP Options (5)
=> MSS: 1460 SackOK TS: 1215651667 0 NOP WS: 0
```

domain: FREENODE.NET
owner-address: Peer-Directed Projects Center
owner-address: 9212 Burdine St. #1005
owner-address: 77096-3221
owner-address: Houston
owner-address: Texas
owner-address: United States of America
admin-c: RL168-GANDI
tech-c: RL168-GANDI
bill-c: RL168-GANDI
nserver: ns1748.freenode.net 66.140.25.156

nserver: ns.bofh.it 213.92.8.2
reg_created: 2002-07-13 09:15:07
expires: 2003-07-13 09:15:07
created: 2002-07-13 15:15:08
changed: 2003-03-14 00:27:52

person: Robert Levin
nic-hdl: RL168-GANDI
address: 9212 Burdine St. Apt. 1005
address: 77096-3221
address: Houston
address: Texas
address: United States of America
phone: +1 7137212351
e-mail: hko0s37u@somegeek.org

2) 81.56.209.187 from the Scan files
This IP address should be watched closely due to the sheer size of the scan performed on the MY.NET network. It is possible that after performing this scan that this person may be back again to scan for another port or ports.

```
Apr 22 07:08:56 81.56.209.187:1768 -> 130.85.1.191:139 SYN ******S*
Apr 22 07:08:57 81.56.209.187:1805 -> 130.85.1.228:139 SYN ******S*
Apr 22 07:08:57 81.56.209.187:1809 -> 130.85.1.232:139 SYN ******S*
Apr 22 07:08:57 81.56.209.187:1810 -> 130.85.1.233:139 SYN ******S*
Apr 22 07:08:57 81.56.209.187:1814 -> 130.85.1.237:139 SYN ******S*
```

inetnum: 81.56.115.0 - 81.56.255.255
netname: FR-PROXAD-ADSL
descr: Proxad / Free Telecom
descr: Dynamic pool (IP/ADSL FT)
descr: NCC#2003034473
country: FR
admin-c: ACP23-RIPE
tech-c: TCP8-RIPE
status: ASSIGNED PA
mnt-by: PROXAD-MNT
changed: nhyvernat@corp.free.fr 20030402
source: RIPE
route: 81.56.0.0/15
descr: ProXad network / Free SA
descr: Paris, France
origin: AS12322
notify: ripe-notify@proxad.net
mnt-by: PROXAD-MNT

changed: nhyvernat@corp.free.fr 20021115
source: RIPE
role: Administrative Contact for ProXad
address: Free SA / ProXad
address: 24, rue Emile Menier
address: 75116 Paris
phone: +33 1 56 26 20 00
fax-no: +33 1 49 04 48 71
e-mail: hostmaster@proxad.net
trouble: Information: http://www.proxad.net/
trouble: Spam: mailto:abuse@proxad.net
admin-c: RA999-RIPE
tech-c: NH1184-RIPE
tech-c: NS496-RIPE
nic-hdl: ACP23-RIPE
notify: ripe-notify@proxad.net
mnt-by: PROXAD-MNT
changed: nhyvernat@corp.free.fr 20010809
changed: tom@proxad.net 20021125
changed: nhyvernat@corp.free.fr 20030331
source: RIPE

3) 158.36.40.5 from the Scan files
This IP address should be monitored due to the large scale port scan that was performed by them on the MY.NET network. Due to the sheer size of it this would indicate to me that this IP address should be watched for further activity.

```
Apr 22 05:16:56 158.36.40.5:30121 -> 130.85.1.30:80 SYN ******S*
Apr 22 05:16:56 158.36.40.5:30123 -> 130.85.1.32:80 SYN ******S*
Apr 22 05:16:56 158.36.40.5:30188 -> 130.85.1.97:80 SYN ******S*
Apr 22 05:16:56 158.36.40.5:30192 -> 130.85.1.101:80 SYN ******S*
Apr 22 05:16:57 158.36.40.5:30205 -> 130.85.1.114:80 SYN ******S*
```

inetnum: 158.36.0.0 - 158.36.255.255
netname: UNINETT1
descr: Academic and research institutions
descr: Uninett South & East Norway
country: NO
admin-c: PK21
tech-c: HE15
tech-c: JG155-RIPE
tech-c: UN49-RIPE
rev-srv: nac.no
rev-srv: biff.uninett.no
rev-srv: sunic.sunet.se
mnt-by: UNINETT-MNT
mnt-lower: UNINETT-MNT

mnt-irt: IRT-UNINETT-CERT
status: ASSIGNED PI
changed: Havard.Eidnes@runit.sintef.no 19930302
changed: Havard.Eidnes@runit.sintef.no 19950321
changed: Jarle.Greipsland@runit.sintef.no 19970105
changed: he@uninett.no 20020923
changed: jarle@uninett.no 20030210
source: RIPE
role: UNINETT NOC
address: UNINETT
address: N-7465 Trondheim
address: Norway
phone: +47 73 55 79 60
phone: +47 73 55 79 61
fax-no: +47 73 55 79 01
e-mail: drift@uninett.no
admin-c: OS372-RIPE
tech-c: OS372-RIPE
tech-c: HE15
nic-hdl: UN49-RIPE
remarks: Abuse: abuse@uninett.no
remarks: Security: cert@uninett.no
mnt-by: UNINETT-MNT
changed: he@uninett.no 20020522
source: RIPE

4) 80.14.15.28 from the scan files.
This IP address should be watched due to the sheer size of the ftp scan performed against the MY.NET network.

```
Apr 21 08:05:58 80.14.15.28:4754 -> 130.85.1.50:21 SYN ******S*
Apr 21 08:05:58 80.14.15.28:4789 -> 130.85.1.79:21 SYN ******S*
Apr 21 08:05:58 80.14.15.28:4799 -> 130.85.1.89:21 SYN ******S*
Apr 21 08:05:58 80.14.15.28:4812 -> 130.85.1.98:21 SYN ******S*
Apr 21 08:05:58 80.14.15.28:4816 -> 130.85.1.102:21 SYN ******S*
```

domain: wanadoo.fr
descr: Wanadoo Interactive
descr: 48, rue Camille Desmoulins
descr: 92442 Issy Les moulineaux cedex
admin-c: BD179-FRNIC
tech-c: FTI-FRNIC
zone-c: NFC1-FRNIC
nserver: ns.wanadoo.fr 193.252.19.10
nserver: ns.wanadoo.com

nserver: ns2.wanadoo.fr 193.252.19.11
nserver: ns2.wanadoo.com
mnt-by: FR-NIC-MNT
mnt-lower: FR-NIC-MNT
changed: frnic-dbm-updates@nic.fr 20011017
source: FRNIC

role: Contacts of FTI
address: France Telecom Interactive
address: 41, rue Camille Desmoulins
address: 92442 Issy Les Moulineaux cedex
phone: +33 1 41 33 39 00
fax-no: +33 1 41 33 39 01
e-mail: postmaster@wanadoo.fr
e-mail: abuse@wanadoo.fr
trouble: mail postmaster for ANY problem.
admin-c: SC1509-FRNIC
tech-c: TEFS1-FRNIC
tech-c: SC1509-FRNIC
tech-c: NS1058-FRNIC
tech-c: CC1215-FRNIC
tech-c: IH678-FRNIC
nic-hdl: FTI-FRNIC
notify: ripe.mnt@fti.net
mnt-by: FT-INTERACTIVE
changed: Patrice.Robert@fti.net 19990413
changed: Patrice.Robert@fti.net 19990415
changed: Patrice.Robert@fti.net 19990506
changed: addr-reg@rain.fr 19990921
changed: migration-dbm@nic.fr 20001015
source: FRNIC

5) 216.150.46.215 from the Alert files.
This IP address should be watched due to the large scale Sub7 scan performed against the MY.NET network.

```
04/21-12:32:59.612459 [**] Possible trojan server activity [**]
216.150.46.215:4432 -> 10.10.1.4:27374
04/21-12:33:18.284656 [**] Possible trojan server activity [**]
216.150.46.215:2632 -> 10.10.2.120:27374
04/21-12:33:30.455833 [**] Possible trojan server activity [**]
216.150.46.215:4303 -> 10.10.3.67:27374
04/21-12:33:30.457942 [**] Possible trojan server activity [**]
216.150.46.215:4312 -> 10.10.3.76:27374
04/21-12:33:30.501190 [**] Possible trojan server activity [**]
216.150.46.215:4338 -> 10.10.3.101:27374
```

Registrant:
NetsCorp
Teresa Walters
PO Box 440812
Laredo, TX 78044
US
956-791-6387 (FAX) 956-791-0448
716@whois.gkg.net

Administrative Contact:
NetsCorp
Teresa Walters
PO Box 440812
Laredo, TX 78044
US
956-791-6387 (FAX) 956-791-0448
716@whois.gkg.net

Technical Contact:
NetsCorp
Teresa Walters
PO Box 440812
Laredo, TX 78044
US
956-791-6387 (FAX) 956-791-0448
716@whois.gkg.net

Internal IP address to be audited

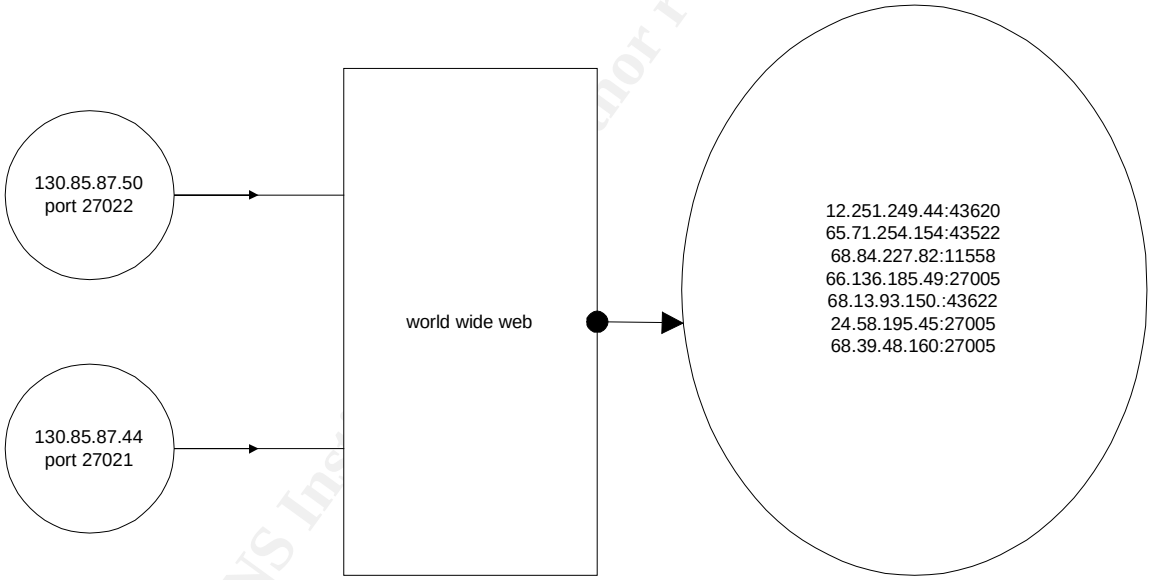
130.85.238.198 from the Scan files.

This IP address from the MY.NET network is performing a very chaotic scan of external IP addresses with a source port anchored at 1142. A thorough google search was performed for this source port of 1142 with nil results. The strange variety of destination ports being scanned is baffling as well. The fact that the protocol used is UDP is odd as well when taken into account the strange activity that this computer is displaying. I would highly recommend that this computer be disconnected from the network and looked at for any evidence of malware. An example of the chaotic scanning activity is noted below. A more detailed analysis could of been completed had the complete logs been available.

```
Apr 23 02:40:51 130.85.238.198:1142 -> 12.238.4.4:3909 UDP
Apr 23 02:40:51 130.85.238.198:1142 -> 12.207.235.152:2769 UDP
Apr 23 02:40:51 130.85.238.198:1142 -> 165.82.96.166:3635 UDP
Apr 23 02:40:51 130.85.238.198:1142 -> 12.227.52.246:1201 UDP
Apr 23 02:40:51 130.85.238.198:1142 -> 12.225.56.89:1052 UDP
```

```
Apr 23 02:40:51 130.85.238.198:1142 -> 12.231.127.237:1090 UDP
Apr 23 02:40:52 130.85.238.198:1142 -> 12.209.248.91:1135 UDP
Apr 23 02:40:52 130.85.238.198:1142 -> 198.82.80.137:2082 UDP
Apr 23 02:40:52 130.85.238.198:1142 -> 206.74.70.43:1474 UDP
Apr 23 02:40:52 130.85.238.198:1142 -> 65.26.133.100:2388 UDP
Apr 23 02:40:52 130.85.238.198:1142 -> 213.209.64.85:2465 UDP
Apr 23 02:40:52 130.85.238.198:1142 -> 12.236.10.33:3573 UDP
Apr 23 02:40:52 130.85.238.198:1142 -> 12.208.145.219:1326 UDP
Apr 23 02:40:53 130.85.238.198:1142 -> 152.1.84.182:1113 UDP
```

Link Graph Analysis



As shown in the above noted link graph there are two IP addresses belonging to the MY.NET network which are Half Life game servers. The bubble to the far right exemplifies but a small sample of the over 800 IP addresses it was communicating with. This very clearly shows the large amount of MY.NET bandwidth that is being used for purposes of a dubious nature. This is not to mention as well the problems associated with game serves such as Half Life and Unreal Tournament which can be used in a DDoS attack. This could cause potential embarrassment to the organization running this network. As well depending on the country you live in possible court action against your company. This practice needs to be halted immediately and ideally the ports blocked at the router so this problem does not reoccur.

Defensive Recommendations

It is highly recommended that the use of file sharing applications such as WinMX and Kazaa amongst others be immediately halted. With new legislation being introduced in certain countries institutions may be help responsible for the illegal file sharing of copyrighted material.
It is also recommended that the running of game serves such as Half Life and Unreal Tournament be halted immediately as well. These servers can be used in the furtherance of such malicious activities as DDoS attacks. This could prove embarrassing and possibly costly to an organization should they be sued for loss of commerce of bandwidth by an affected DDoS victim.
It is still recommended however that certain ports be blocked outright in an effort to tighten up the security of the network. If ports that are not offering services such as 139 or 1433 and 1434 then they should be blocked at the gateway. This will in turn save on whitenoise.
A fuller detailed analysis should still be completed at the location itself where the analyst could have access to the full and complete logs as well as the network schematics. Lastly it would of been very helpful had the Analyze This! data been supplied in binary format.

References

1) Ricky Smith http://www.giac.org/practical/GCIA/Ricky_Smith_GCIA.pdf
2) Johnny Calhoun http://www.giac.org/practical/GCIA/Johnny_Calhoun.GCIA.pdf
3) NBtdump http://www.atstake.com/research/tools/info_gathering/
4) Botnet http://zine.dal.net/previousissues/issue19/botnet.php
5) Code Red http://www.sans.org/rr/paper.php?id=32
6) Worm http://best-managers.com/viruses-explained.htm
7) Unicode Exploit http://www.cgisecurity.com/lib/URLEmbeddedAttacks.html
8) DMZ http://www.homenethelp.com/web/explain/port-forwarding-dmz.asp
9) identd http://www.securityfocus.com/bid/2840/discussion/
10) ridentd http://www.xs4all.nl/~rmeijer/rident.html
11) Decoding http://archives.neohapsis.com/archives/snort/2000-11/0244.html
12) Cookies http://computer.howstuffworks.com/cookie1.htm
13) MF http://www.incidents.org/archives/intrusions/msg10000.html
14) TFTP http://thedp.netfirms.com/cgi-bin/tut_load.pl?file=tftp
15) Queso http://www.iss.net/security_center/advice/Intrusions/2000321/default.htm

Last Updated: May 22nd, 2017

Upcoming Training

Mentor Session - SEC503	Oceanside, CA	May 29, 2017 - Jun 29, 2017	Mentor
Security Operations Center Summit & Training	Washington, DC	Jun 05, 2017 - Jun 12, 2017	Live Event
SANS Houston 2017	Houston, TX	Jun 05, 2017 - Jun 10, 2017	Live Event
SANS Columbia, MD 2017	Columbia, MD	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS London July 2017	London, United Kingdom	Jul 03, 2017 - Jul 08, 2017	Live Event
SANSFIRE 2017	Washington, DC	Jul 22, 2017 - Jul 29, 2017	Live Event
SANSFIRE 2017 - SEC503: Intrusion Detection In-Depth	Washington, DC	Jul 24, 2017 - Jul 29, 2017	vLive
SANS San Antonio 2017	San Antonio, TX	Aug 06, 2017 - Aug 11, 2017	Live Event
SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Adelaide 2017	Adelaide, Australia	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Virginia Beach 2017	Virginia Beach, VA	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS Network Security 2017	Las Vegas, NV	Sep 10, 2017 - Sep 17, 2017	Live Event
SANS vLive - SEC503: Intrusion Detection In-Depth	SEC503 - 201709,	Sep 11, 2017 - Oct 18, 2017	vLive
SANS London September 2017	London, United Kingdom	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Baltimore Fall 2017	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	Live Event
Baltimore September 2017 - SEC503: Intrusion Detection In-Depth	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	vLive
Community SANS Scottsdale SEC503	Scottsdale, AZ	Oct 02, 2017 - Oct 07, 2017	Community SANS
SANS October Singapore 2017	Singapore, Singapore	Oct 09, 2017 - Oct 28, 2017	Live Event
Community SANS Ottawa SEC503	Ottawa, ON	Oct 16, 2017 - Oct 21, 2017	Community SANS
SANS San Diego 2017	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	Live Event
San Diego Fall 2017 - SEC503: Intrusion Detection In-Depth	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	vLive
SANS Seattle 2017	Seattle, WA	Oct 30, 2017 - Nov 04, 2017	Live Event
SIEM & Tactical Analytics Summit & Training	Scottsdale, AZ	Nov 28, 2017 - Dec 05, 2017	Live Event
SANS OnDemand	Online	Anytime	Self Paced
SANS SelfStudy	Books & MP3s Only	Anytime	Self Paced