Global Information Assurance Certification Paper

Intrusion Detection and Analysis

Joona Airamo
GIAC GCIA Practical v3.3
Submitted 25 April 2003

Table of Contents

Assignment 1 – White Paper "Comparison of Firewalls, IDS and IPS"

Abstract

This paper studies the roles of three security devices: the gateway firewall, the network intrusion detection system and the intrusion prevention system. The strengths and weaknesses are listed for all of them. The needs of detection are reasoned by showing that all attacks and anomalies cannot be prevented with full accuracy. In the conclusion there is a summary of the differentiating factors of these devices.

Security Layers

A good security consists of layers of prevention, detection and reactions. These layers provide *defense in depth* – an important attribute for the defender.

The preventing methods include all devices, configurations, processes and policies that together decrease the probability of certain threats to occur, as well as minimize the potential damage these threats may cause. The prevention is an essential and foremost protection method, since it stops the bad things before they happen. Almost every component in a computer network may have a role in preventing incidents to occur; in this paper we focus on gateway firewall devices.

The detection means noticing events that violate the organization's security policy. Here we focus on network based intrusion detection systems that have the ability to distinguish the normal, legitimate traffic and the traffic that is suspicious.

Finally, the reactions contain pre-planned processes, automatic or manual, that have the purpose to get rid of intruders or other problems and to return the systems back to the normal state as fast and reliably as possible.

Gateway Firewall

A gateway firewall is able to monitor only the traffic that goes through the firewall. Therefore the firewall must be located in the network in such a way that any two network devices whose communication must be regulated can send packets to each other only through the firewall. In practice, the firewalls are located between the network segments to monitor all traffic that travel from one segment to another one. A limitation of this approach is that the firewall is unable to restrict the communication within a network segment.

Because the firewall operates in an *inline* mode, it has full control over the traffic that attempts to travel through the firewall. The firewall stops every packet for inspection and lets the packet to continue only after the inspection has been done with an approving result. If the firewall breaks down or is overloaded, it fails to the safe side: no unauthorized traffic can pass the firewall.

The firewall bases its matching criteria on values that are located in the network and transport layer headers of the packet and typically omits the application payload validation. The firewall is good at preventing connections to non-allowed services, but fails to detect attacks within a connection that it allows to pass.

NIDS

The NIDS listens a network segment and monitors all traffic that travels there. Therefore it sees more traffic than a firewall does – not only the packets that arrive or leave the segment but also all the traffic between the hosts in the single segment. Depending on the needs to monitor the traffic between hosts within a single segment – and also on the size of the segments – it may be more cost-effective to utilize a NIDS than to try to isolate the hosts so that they would be able to communicate with each other only through a firewall.

Since the NIDS is not a part of the network path between the communicating devices, it is unable to prevent the offending packets to pass; it can only notice that this is happening. Marcus Ranum [Ranum] wrote an excellent article how frustrating this can be. He also pointed out that since the network administrator always knows his network better than an attacker, he can use this advantage to monitor for unexpected network events to catch the attacker's moves.

There are several reactive functions that have been implemented in many NIDS. These include sending terminating packets, such as TCP Reset, to kill the violating connections or instructing a firewall to prevent further packets from the violating source. Larsen and Haile [Larsen] show several examples where this approach works only partially, or not at all. The NIDS is well suited for detecting incidents and raising alerts. The preventive abilities are limited, though, since at the time when the NIDS at the earliest can perform a reactive action, the damage may have already occurred.

The NIDS usually performs a more thorough investigation for network traffic than a typical firewall. The firewall's decision is most often based on the information in the network packet's header fields; the NIDS typically looks at the payload part of the packet. A very common rule for a NIDS is to search for a pre-defined string of bytes in the payload that would indicate a certain type of incident of taking place. Another approach is to validate the traffic against some well-defined specification in order to find anomalies. Shankar and Paxson [Shankar] developed an interesting method for an IDS to learn the

network characteristics to better distinguish between the malicious traffic and non-harming garbage.

Intrusion Prevention System (IPS)

The idea of the intrusion preventing system is an interesting one. By combining the best parts of the firewall and the NIDS, the IPS is not only able to detect the attacks, but also to prevent them to pass to their destinations.

The IPS works in inline mode, just like the firewall. Therefore the traffic cannot bypass the IPS and every packet is subject to the inspection. The inline topology also gives the same restrictions to the IPS than the firewall has: only the traffic that goes through the IPS can be regulated. This means that if the IPS is monitoring the traffic between two network segments, the communication between hosts within a single segment is not restricted by the IPS.

While the firewall typically looks only the header information in the packets, the IPS inspects also the payload. The IPS and NIDS have identical techniques to detect incidents at the time when they are attempted. The difference comes from the fact that the IPS is better equipped to prevent the violating traffic to pass than the NIDS is.

There is one more advantage of implementing IPS instead of NIDS. Horizon [horizon] describes many tricks that an attacker can use to evade an intrusion detection system. All of them work because they make the NIDS to interpret the traffic differently than the target of the attack. Many of these techniques, such as fragmentation and TCP out-of-order-segments can be better handled by the IPS, since it can force the packets to wait until all relevant information has been arrived for deciding whether the traffic may be passed or denied.

False Positives

One of the biggest challenges in the art of intrusion detection is to minimize the occurrence of false alarms. To do that, the detection mechanisms of single incidents should be as explicit as possible. However, this leads to a situation where a detection signature for a single attack fails to detect a small variation of the same attack. On the other hand, if the signature is loose enough to catch multiple attack variations, the probability to raise false alarms increase.

It is not even possible to make an explicit detection signature for every interesting network activity. If the desire is to detect the leakage of company secrets by catching keywords from SMTP or IRC traffic, for example, the false positive rate can be pretty high.

The challenge of reducing the false positives applies to both IPS and NIDS, since they both try to detect incidents from the traffic. There is a fundamental

difference, however, of the effect of false positives on these systems. If the NIDS reports a false positive, it burdens the administrator of the NIDS, but the operational traffic that caused the alarm is not affected. If the IPS judges wrongly, the operational traffic stops until the administrator inspects the situation manually and fixes the IPS rule base. For this reason the IPS should only run rules that have proved to generate false alarms rarely or not at all.

Correlation

The role of the intrusion detection system is not only to catch immediate, single violating packets, but also to look at the big picture. A single network packet may look innocent when alone, but together with other pieces of information may turn out to be an essential part of an attack. The pieces that the IDS correlates may come from multiple sources and at different times.

The correlation is never done in real-time. It should rather be understood as a background process that monitors the logs that other components have produced. The events that generate a correlation are gone long before any preventive mechanism can learn that these events should have been stopped.

Conclusions

The three analyzed network security devices – the gateway firewall, the NIDS and the IPS – have separate roles, functions and strengths. The differentiating factors between the devices are:

- NIDS operates in stealth mode and is able to monitor all traffic within a single segment; IPS and firewall operate in inline mode between two or more network segments
- The inline operating mode gives the firewall and IPS an advantage over the IPS in preventing the unwanted traffic to pass the security device
- Similarly, because of the operating mode, the devices behave differently when overloaded or if broken. The firewall and IPS disturb the legitimate traffic but do not let the unwanted traffic to pass; the NIDS passes (some) traffic without inspecting it.
- Firewall inspects only the IP and transport headers; NIDS and IPS look at the packet payloads
- Because the NIDS and IPS have a lot more complex task interpreting the traffic than the firewall, they have much greater risk to have a bug in the interpreting code causing a vulnerability in the product. There have been many examples of vulnerabilities in the traffic interpreting programs, such as the bug in Snort's stream4 preprocessor [snort-stream4].

These factors are also summarized in the table below.

	Gateway Firewall	NIDS	IPS
IP header check	*Yes*	*Yes*	*Yes*
Transport header check	*Yes*	*Yes*	*Yes*
Payload check	*No*	*Yes*	*Yes*
Inline	Yes	No	Yes
Preventive capabilities	*Yes*	*Limited*	*Yes*
When overloaded or broken	*Drops traffic*	*Does not inspect (all) data*	*Drops traffic*
Effect of false positives	*N/A*	*May stop legitimate traffic*	*Stops legitimate traffic*
Stealth mode	*No*	*Yes*	*No*
Prone to have vulnerabilities	*Less likely than NIDS or IPS*	*Yes*	*Yes*

The study here has focused on their primary roles of the firewall, NIDS and IPS. If the functions of these devices are combined to a single box, the combined solution also combines the strengths and weaknesses of the original functions. If, for instance, a firewall and IPS functions were implemented to a single device, it would gain the advantages of the IPS to be able to inspect also the packet payloads. However, at the same time it would also inherit the potential vulnerabilities of the more complex interpreting functions than the firewall alone would have had.

References

[horizon] horizon, “Defeating Sniffers and Intrusion Detection Systems”, Phrack Magazine issue 54, 25 December 1998

[Larsen] Larsen, Jason & Haile, Jed, “Understanding IDS Active Response Mechanisms”, URL: http://online.securityfocus.com/infocus/1540, 29 January 2002

[Ranum] Ranum, Marcus, “Intrusion Detection: Challenges and Myths”, URL: http://secinf.net/info/ids/ids_mythe.html, 16 October 2002

[Shankar] Shankar, Umesh & Paxson, Vern, “Active Mapping: Resisting NIDS Evasion Without Altering Traffic”, URL: http://www.cs.berkeley.edu/~ushankar/research/active/activemap.pdf, 6 November 2002

[snort-stream4] Snort developing team, “Snort Advisory: Integer Overflow in Stream4”, URL: http://www.snort.org/advisories/snort-2003-04-16-1.txt, 16 April 2003

Assignment 2 – Network Detects

Detect #1

Anomaly in TCP sequence numbers

1. Source of Trace

The network traffic analysed in this paper was taken from incidents.org's repository of Snort binary log at <http://www.incidents.org/logs/Raw>. The log file used is 2002.9.30.

According to the README file in the same directory with the log files, all packets in the log files were captured by a Snort instance running in binary logging mode. All IP addresses of the protected network have been changed and the IP and TCP checksums have been modified to have incorrect values.

I started my traffic analysis by looking at the MAC addresses. There seemed to be only two unique MAC addresses: 00:00:0c:04:b2:33 and 00:03:e3:d9:26:c0. According to [mac_find], both of these addresses belong to Cisco Systems, Inc. Every frame in the source file had one of these addresses as source address and the other one as destination address. I am assuming that the Snort instance that captured the packets was sitting between two Cisco devices and that there were no other (active) devices in the same segment.

Next I looked at the IP addresses. The frames that had the source MAC address of 00:00:0c:04:b2:33 all shared the same source IP address of 207.166.87.157. The frames coming from 00:03:e3:d9:26:c0 had a wider variety of IP addresses as the destination address. The smallest of them was 207.166.10.149; the largest was 207.166.252.18. I assume that the whole B-class size network 207.166.0.0/16 is routed through the 00:00:0c:04:b2:33 –address. The IP addresses behind the 00:03:e3:d9:26:c0 –address seemed to vary between 4.x.x.x and 217.x.x.x, however not in 207.166.x.x address space.

My conclusion for the network topology thus is the following:

```
Internet -- Cisco:c0 ----- Cisco:33 -- 207.166.0.0/16
                             |
                           Snort
```

I found that at least Andre Cormier had drawn similar conclusion in his analysis. Andre's paper can be found here:

http://cert.uni-stuttgart.de/archive/intrusions/2003/01/msg00121.html

Snort seems to have captured traffic only from a single source IP address leaving the protected network: 207.166.87.157. However, there are several indications that more than one host is sharing the same IP address. The indications that make me to believe this are:

- Multiple TTL values exist in the packets coming from 207.166.87.157. The values are: 123,124,125 and 240.

- As correlation to the TTL values, the IP ID number is zero in all packets that have TTL 240 and non-zero in others.

- Also as correlation to the TTL values, the Type of Service is always 0x00 for TTL values 123 and 125; always 0x10 for TTL 240 and varies a lot for TTL 124.

- Also as correlation to the TTL values, the DF flag is set in every packet that have TTL values of 123, 124 or 125; the DF flag is zero in packets with TTL 240.

I can think of two different possibilities why multiple hosts would seem to share the same IP address:

a) There is a device between Snort and 207.166.x.x network (maybe Cisco:33) that does dynamic network address translation hiding all outgoing traffic behind a single IP address.

b) The README file in the log repository directory says that the IP addresses of the protected network have been modified. It is possible that this modification was done in such a way that all the addresses in the protected network were replaced with a single address.

I do not believe that there would be a proxy device between Snort and 207.166.x.x network. In that case all TTL values should be identical at the place of observation. I am assuming that the case a) (dynamic NAT) is the reason for this traffic behaviour. However, let us keep in mind the possibility for case b) also during our further analysis.

An additional notice of the network is that some of the packets have long total packet lengths. The biggest packet has total length of 6772 bytes -- well over the ethernet MTU value of 1500. None of the captured packet payloads exceeds 1500 bytes, though, which indicates that the snaplen parameter was set to 1500 bytes when Snort was running. All packets over 1500 bytes are coming from 207.166.x.x network and none of them are fragmented. I assume that the 207.166.x.x network and the network that Snort is listening have MTU values greater than 1500 and they are therefore something else than ethernet. [Stevens, page 30] lists typical MTU values for different networks. My guess is that the detects have been captured from a 16 Mbs token ring – network.

2. Detect was generated by

Every single packet in the raw log file has been captured by Snort intrusion detection system. I am not aware of the version of neither the Snort nor the rulebase used.

It is important to remember that there have been a lot of packets in the network that have not been captured to the log. Therefore a missing packet does not proof anything else than it did not match the Snort rulebase that was used.

I installed Snort version 1.9.1 and downloaded the default rulebase from www.snort.org. I run the Snort against the log file and got multiple reports of all kind of anomalies. The anomaly I am presenting here, however, was not detected by the Snort version and configuration I had. I hate to say but I detected it purely accidentally when browsing through the log file with Snort, tcpdump and Ethereal.

I noticed that there are cases where a sending host uses the same TCP sequence number in multiple packets within a single TCP connection –- but has different payload in them.

An example of this detect is given here as a tcpdump trace:

```
$ /usr/sbin/tcpdump -nexSr 2002.9.30 'tcp[4:4] = 0x03967f19'

18:08:10.446507 0:0:c:4:b2:33 0:3:e3:d9:26:c0 ip 2974:
207.166.87.157.62514 > 216.136.232.84.http: P 60194585:60197505(2920)
ack 25114938 win 64240 [tos 0x10]
                  4510 0b90 0000 0000 f006 0000 cfa6 579d
                  d888 e854 f432 0050 0396 7f19 017f 393a
                  5018 faf0 0000 0000 696a 517f 8057 8788
                  4175 7556 8a8a 4276 764f 8383 527f 8353
                  <snip>
18:08:10.536507 0:0:c:4:b2:33 0:3:e3:d9:26:c0 ip 1514:
207.166.87.157.62514 > 216.136.232.84.http: P 60194585:60196045(1460)
ack 25116398 win 65535 [tos 0x10]
                  4510 05dc 0000 0000 f006 0000 cfa6 579d
                  d888 e854 f432 0050 0396 7f19 017f 3eee
                  5018 ffff 0000 0000 1c1c 1a1e 1f24 2829
                  2423 252c 2729 2b25 2629 271f 252d 2222
                  <snip>
18:08:11.746507 0:0:c:4:b2:33 0:3:e3:d9:26:c0 ip 2406:
207.166.87.157.62514 > 216.136.232.84.http: P 60194585:60196937(2352)
ack 25169362 win 64808 [tos 0x10]
                  4510 0958 0000 0000 f006 0000 cfa6 579d
                  d888 e854 f432 0050 0396 7f19 0180 0dd2
                  5018 fd28 0000 0000 577d 414c 6839 4158
                  323d 5928 3353 232e 4e25 3050 2831 5221
                  <snip>
18:08:12.016507 0:0:c:4:b2:33 0:3:e3:d9:26:c0 ip 2406:
207.166.87.157.62514 > 216.136.232.84.http: P 60194585:60196937(2352)
ack 25185746 win 64808 [tos 0x10]
                  4510 0958 0000 0000 f006 0000 cfa6 579d
                  d888 e854 f432 0050 0396 7f19 0180 4dd2
                  5018 fd28 0000 0000 2125 2a15 1914 1a18
```

```
                  1022 1c1d 1b18 1419 1612 1613 0f16 130f
                  <snip>
18:08:12.156507 0:0:c:4:b2:33 0:3:e3:d9:26:c0 ip 2406:
207.166.87.157.62514 > 216.136.232.84.http: P 60194585:60196937(2352)
ack 25193938 win 64808 [tos 0x10]
                  4510 0958 0000 0000 f006 0000 cfa6 579d
                  d888 e854 f432 0050 0396 7f19 0180 6dd2
                  5018 fd28 0000 0000 6d49 6b6a 496d 6730
                  554b 2b51 453e 6b5b 295b 5129 5465 9fb4
                  <snip>
18:08:12.266507 0:0:c:4:b2:33 0:3:e3:d9:26:c0 ip 2974:
207.166.87.157.62514 > 216.136.232.84.http: P 60194585:60197505(2920)
ack 25196858 win 64240 [tos 0x10]
                  4510 0b90 0000 0000 f006 0000 cfa6 579d
                  d888 e854 f432 0050 0396 7f19 0180 793a
                  5018 faf0 0000 0000 522b 6053 3469 5c3c
                  7164 467b 6e2e 6856 2666 5022 664d 2568
                  <snip>
18:08:12.436507 0:0:c:4:b2:33 0:3:e3:d9:26:c0 ip 2406:
207.166.87.157.62514 > 216.136.232.84.http: P 60194585:60196937(2352)
ack 25210322 win 64808 [tos 0x10]
                  4510 0958 0000 0000 f006 0000 cfa6 579d
                  d888 e854 f432 0050 0396 7f19 0180 add2
                  5018 fd28 0000 0000 7264 5085 7857 897d
                  4779 6d31 6357 4d7c 735b 7a77 5d72 7061
                  <snip>
```

The following tcpdump flags were used:

- -n to avoid name resolution
- -e to show link layer addresses
- -x to show packets in hexadecimal dump
- -S to show the actual sequence numbers
- -r to read the traffic from a file

The expression ‘tcp[4:4] = 0x03967f19’ tells tcpdump to show only those packets that have TCP sequence number of 60194585 (0x03967f19 in hex). There appeared to be seven packets sharing this sequence number.

All packets have source IP address of 207.166.87.157 and destination IP of 216.136.232.84. So they are coming from the protected network and leaving for the Internet. The MAC addresses confirm this assumption. All of them were sent within a short period of time (within two seconds) and they all share the same source and destination TCP ports. Therefore, by definition, they belong to a single TCP connection.

[RFC 793] is the standard for TCP protocol. According to it there is only a single case when a sending host should keep the same sequence number in multiple packets: when retransmitting data that was not correctly delivered in the first try. The TCP data payload -- or at least the overlapping part of it if the packets are of different length -- must be identical in the retransmitted packet than in the original one.

All of the detected packets clearly have different payloads. The first two bytes of the payload in the seven packets are:

0x696a
0x1c1c
0x577d
0x2125
0x6d49
0x522b
0x7264

None of them indicates of TCP retransmitting. This anomaly got my attention and I decided to give it further analysis.

3. Probability the source address was spoofed

The anomalies in the detect are particularly alarming, since they appear to come from the protected network. I am assuming that the network topology is the one described above. I am not sure if the network has any anti-spoofing mechanism in place but the IP/MAC address correlation gives pretty strong evidence that no spoofing is happening.

It may be possible that the source IP address is not the one that the sending host holds, but at least the packets are coming from the right direction. Since I assume that there is a device doing dynamic NAT between the 207.166.x.x network and the Snort instance capturing the traffic, my guess is that the source IP is modified by the NAT device but not necessarily spoofed by an attacker.

4. Description of attack

I was unable to find any known attack that would cause this type of TCP sequence number anomaly.

Every host controls the TCP sequence numbers it sends independently, without any influence from the other hosts in the network. Therefore there must be something weird in the sending host. Furthermore, since the sender of these crafted packets seems to lie in the protected network, the attention the detect deserves is high.

I'll present some theories why the anomaly might exist in the correlations section below.

5. Attack mechanism

Let us have a closer look at the packets. All of them have the source IP address of 207.166.87.157 and source TCP port of 62514. They all have destination IP of 216.136.232.84 and destination TCP port of 80. Even though the destination port indicates http traffic, the payload is not pure ASCII, since there are bytes having greater value than 0x7f. The amount of data sent from

the client to the server is pretty big: 16708 bytes in these seven packets alone. Unfortunately the http protocol is used also for many other purposes than requesting html documents. The data in the packets looks pretty random, which indicates that the http connection is used to tunnel some unknown protocol. This alone is not necessarily a sign of malicious traffic, just an observation that we do not know for sure what is going on here. The destination address resolves to a name 'f214.mail.yahoo.com' that might indicate the use of some sort of web mail.

All of the packets have IP ID number zero. According to [RFC 791] the ID values "must be unique for that source-destination pair and protocol for the time the datagram will be active in the internet system." The ID numbers are easy to spoof, but I am not aware of any way to force a remote host to stay in one ID number. Thus the sender must have selected the ID numbers. This is also an indicator that says that there is something strange in the sending host.

All of the packets have a common TTL value of 240, a common TOS of 0x10 (minimize delay), none of them are fragmented, none of them have the DF flag set, none of them have IP or TCP options. They all have TCP flags PSH and ACK set and the TCP window size is 64k or thereabouts in every packet. These are all valid values, even though I would have assumed to see the DF flag set since the IP IDs used do not really support fragmentation.

One of the packets has total length of 1500 bytes; all others are well over 2000 bytes long.

6. Correlations

I decided to run another test with tcpdump against the same log file:

```
$ /usr/sbin/tcpdump -nexSr 2002.9.30 'host 207.166.87.157 and host
216.136.232.84 and tcp port 62514 and tcp port 80 and not tcp[4:4] =
0x03967f19'

18:08:10.236507 0:0:c:4:b2:33 0:3:e3:d9:26:c0 ip 946:
207.166.87.157.62514 > 216.136.232.84.http: P 25111126:25112018(892)
ack 60194585 win 8760 (DF)
                  4500 03a4 9782 4000 7c06 c5fa cfa6 579d
                  d888 e854 f432 0050 017f 2a56 0396 7f19
                  5018 2238 14c5 0000 8b57 7e87 4c73 7c6d
                  97a4 4c75 8471 9aa9 6f98 a76b 92a0 4f76
                  <snip>
18:08:11.486507 0:0:c:4:b2:33 0:3:e3:d9:26:c0 ip 946:
207.166.87.157.62514 > 216.136.232.84.http: P 25160278:25161170(892)
ack 60194585 win 8760 (DF)
                  4500 03a4 c082 4000 7c06 9cfa cfa6 579d
                  d888 e854 f432 0050 017f ea56 0396 7f19
                  5018 2238 613f 0000 ab69 91a4 6488 986d
                  91a1 5276 865e 8292 547a 8652 7884 6288
                  <snip>
```

The flags used in tcpdump program are the same than before. The expression tells the tcpdump to show all TCP packets that belong to the same TCP

connection than the seven previous packets but that do not have the same sequence number of 60194585 (0x03967f19 in hex).

As shown above, two new packets were found. These packets have been captured approximately at the same time with the seven previous packets and they have the same IP numbers and TCP ports. Thus they clearly belong to the same TCP connection. What is really strange is that the new packets have the same acknowledge numbers than the previous packets had as sequence numbers -- even though the packets are going to the same direction as the previous ones.

There are also other differences between these two packets and the seven packets analysed before. These two packets have TTL value of 124 (the seven packets have 240); the IP ID number is random, like it should (it is zero in all seven); the DF flag is set (it is unset in the previous packets); TOS is zero (0x10 in previous); TCP window size is 8760 (approximately 64k in previous).

These two new packets differ so much from the seven previous ones that it is hard to believe that they have been sent by the same host. And still they seem to belong to the same TCP connection! This is odd.

I cannot think of any stimulus that would cause a response like this. Therefore I have to assume that this anomaly is created by a host in the protected network, unless I can think of any other means why TCP could behave like this.

If a host repeats the same TCP sequence number while sending new data, the receiver is unable to keep track of the data. I suppose this could be a clever way to confuse firewalls and intrusion detection systems, but it requires a specifically crafted TCP/IP stack both on sending and receiving hosts. If this is the case, we really need to investigate the host in the protected network that is partaking this activity. This would also mean that f214.mail.yahoo.com (the other end of this TCP connection) has been compromised.

Another possibility is that the detect is corrupted somehow. Let us sort all nine packets by the time and list their sequence and acknowledge numbers and the packet total lengths:

```
Frame       Seq        Ack        Packet_length
1           25111126   60194585   932
2           60194585   25114938   2960
3           60194585   25116398   1500
4           25160278   60194585   932
5           60194585   25169362   2392
6           60194585   25185746   2392
7           60194585   25193938   2392
8           60194585   25196858   2960
9           60194585   25210322   2392
```

Let us consider a possibility that the number 60194585 actually is or should be the acknowledge number for all of these packets. The sequence number would then grow from 25111126 to 25210322. That would indicate that starting from the frame #1 a total of 99.196 bytes would have been send by the client just prior the frame #9 was sent. For some unknown reason seven of these packets have their sequence and acknowledge numbers switched.

I browsed through the whole log file and found 11 TCP connections that had the same anomaly. The sequence numbers that repeated in multiple packets in a single connection even though the payload changed are the following:

```
60194585 (0x03967f19 in hex)
776154499 (0x2e432d83 in hex)
1260679045 (0x4b246f85 in hex)
1374708739 (0x51f06403 in hex)
1616916512 (0x60603020 in hex)
2142566233 (0x7fb4f759 in hex)
2659228771 (0x9e809c63 in hex)
3410273829 (0xcb44a625 in hex)
3513873551 (0xd171748f in hex)
3523143865 (0xd1fee8b9 in hex)
4073505461 (0xf2ccc2b5 in hex)
```

I noticed that in every case the packets that had the sequence number anomaly had the IP total length greater or equal than 1500 bytes. For every such connection there are also packets that had the total length smaller than 1500 bytes and which do not have the sequence number anomaly (i.e. the sequence and acknowledge numbers are in opposite order than in the long packets).

These facts tempt me to think that maybe every packet over 1500 bytes is corrupted somehow. And what if also the total length value is corrupted? Who knows if the network MTU is 1500 after all!

There are 835 packets in the log that have the total length value over 1500 bytes. The total packet count in the log is 15021. Thus 5,56% of the packets are big and potentially corrupted.

This number correlates nicely to another piece of information I have experienced before. Some network interface card drivers I have used on Linux may corrupt ethernet traffic up to 5% of the packets they handle. Normally this can be verified by looking at the checksum values -- unfortunately this time the checksums have been deliberately modified and therefore they are not helpful.

If the corruption theory is the right one, then the other differences between the packets may also be explained with the same reason.

The provided data does not give definite answers to the problem. Without further information this is about as far as I am able to speculate. In a real situation I would start inspecting the protected network in order to find the actual host that has sent the packets. Alternatively I might try to capture

similar traffic using different network interface card and driver than the one used this time. I might also want to use completely different capturing tools, such as Solaris with snoop or some hardware traffic analyzer like Fluke. If these methods showed similar behaviour with the TCP sequence numbers, I would be able to eliminate the possibility of having a problem during the capture.

7. Evidence of active targeting

Snort has not captured any packets coming from the IP address 216.136.232.84. No information was found how the host in the internal network would have been compromised. This does not eliminate such a possibility, though.

8. Severity

The severity is a sum of criticality and lethality subtracted by the sum of system and network countermeasures. I evaluate these values as following:

Criticality = 2. The address 207.166.87.157 may be shared by multiple hosts, but this particular one has opened an http connection to f214.mail.yahoo.com. This makes me assume that the client is a workstation.

Lethality = 5. Even though I am suspecting some sort of corruption in the network traffic, the worst case means that the sender's TCP/IP stack has been modified. Only a superuser can do that and I have no reasons to believe that the actual system administrator would do such a trick.

System countermeasures = 1. The system is sending invalid packets to the network. Either the packets are corrupted, or the host has been compromised, in which case the countermeasures have not been good enough.

Network countermeasures = 3. There is no evidence of having a firewall protecting the network, but some other good signs can be detected. First, the IP/MAC address pairs give a strong indication that some sort of anti-spoofing measures are effective in the network. Secondly, if my analysis is correct, there is a device doing dynamic NAT, which makes it harder to contact directly the hosts that are hiding behind the NAT device.

Severity = (2 + 5) – (1 + 3) = 3.

Definitely something to pay more attention to.

9. Defensive recommendation

I have the following defensive recommendations:

a) Inspect the host 207.166.87.157 for possible signs of compromise. If this address is the NAT hide address, consult the logs of the NAT device first in order to find the real host behind this connection.

b) Test the snort instance that captured the packets. Try changing your hardware and device driver or run some other sniffers than snort or tcpdump and compare the results. In normal cases incorrect IP or TCP checksums are strong indications of corrupted traffic -- this time this information was unavailable.

c) Do not forget to use also the manual approach when studying the logs of firewalls and IDS. Often times an experienced network administrator may find "something that just doesn't look right" and that has not been detected by the automated systems.

The advice of testing your tools applies to us all -- there is hardly anything more annoying than a security device that corrupts the logs!

10. Multiple choice test question

A tcpdump trace shows two TCP packets that share the same sequence number. In which case this is against the TCP standard:

a) If both packets also share the same IP ID number

b) If both packets are sent by the same host and they do not belong to the same TCP connection

c) If both packets belong to the same TCP connection, are sent by the same host and have completely different TCP payloads

d) If both packets share the same IP addresses and TCP ports but only one of them is fragmented

Answer is c.

Comments from the community:

I posted the analysis for the detect #1 above to intrusions@incidents.org on 11 April 2003. The analysis can be found from the address: <http://cert.uni-stuttgart.de/archive/intrusions/2003/04/msg00102.html>.

Andrew Rucker Jones commented the analysis on 12 April: <http://cert.uni-stuttgart.de/archive/intrusions/2003/04/msg00123.html>.

Three selected questions from his comments follow, together with my answers posted on 15 April: <http://cert.uni-stuttgart.de/archive/intrusions/2003/04/msg00142.html>

Question #1:

> I really loved Your arguments as to why something behind 207.166.87.157 is doing many-to-one NAT. I have to ask, though, if You remove the packets that You say are corrupted, do You still have those arguments? You say, for instance, that TTL = 240, IP ID = 0, and TOS = 0x10 are all tied together. Those sound like the corrupted packets to me. Are the characteristics of the output packets more homogenous once the corrupted packets are removed?

My answer:

I checked this out. It seemed that out of 15021 packets 1529 have TTL value of 240 and IP ID zero (835 of those have IP total length greater than 1500 bytes). Two of these 1529 packets are inbound and the rest are outbound. Now if we assume that all these are corrupted, then the percentage of corruption would be 10,18%. I have not seen this high corruption percentage before, but I suppose it could be possible.

Even if we remove all TTL=240 packets out, there are still three different TTL values (123, 124 and 125) in the packets sent by 207.166.87.157. I also looked at the TCP window size and it seemed to vary between 5455 and 65189. No significant correlation in window sizes compared to the TTL values, though. I find it difficult to understand why the TTL would vary at all for outbound packets that are sent by the same host -- therefore I still assume that some device is doing dynamic NAT. The evidence is not as strong anymore after removing all of the supposedly corrupted packets, however.

Question #2:

> *> If a host repeats the same TCP sequence number*
> *> while sending new data, the receiver is unable to*
> *> keep track of the data. I suppose this could be a*
> *> clever way to confuse firewalls and intrusion*
> *> detection systems, but it requires a specifically*
> *> crafted TCP/IP stack both on sending and receiving*
> *> hosts.*
>
> *Not necessarily. On the receiving end, it might just reassemble by overwriting the old data. What surprises me (should that be the case), is that the PUSH flag is set. The way i understand it, once that's set, the previous data couldn't be overwritten.*

My answer:

Once the application has received the byte from the TCP, no further packets can overwrite it. This applies whether there is PSH flag set or not -- the use of

PSH flags makes it more likely that the application receives the data before the next segment arrives. For that reason I still believe that the receiver needs a modified TCP stack to be able to receive any other data after the first segment, if the sequence numbers do not change.

Question #3:

> *Network countermeasures = 3. There is no evidence*
> *of having a firewall protecting the network, but*
> *some other good signs can be detected. First, the*
> *IP/MAC address pairs give a strong indication that*
> *some sort of anti-spoofing measures are effective*
> *in the network.*

...or simply that no one tried spoofing in the time frame You analyzed.

My answer:

An excellent point. Given that the dynamic NAT evidence is not that strong anymore either, I am considering to decrease the Network countermeasures value to 2.

Detect #2

Broadcast source IP address

```
12:43:06.876507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 ip 57:
255.255.255.255.31337 > 207.166.120.90.515: R 0:3(3) ack 0 win 0
 4500 002b 0000 0000 0f06 ae17 ffff ffff
 cfa6 785a 7a69 0203 0000 0000 0000 0000
 5014 0000 633f 0000 636b 6f00 0000
```

1. Source of Trace

The trace for the detect #2 is the same as in detect #1 above, the Snort binary log file 2002.9.30 downloaded from <http://www.incidents.org/logs/Raw>.

I concluded with the detect #1 that the trace included some corrupted frames. In addition, the README file in the same directory with the log files states that all IP and TCP checksums have been modified and are incorrect. Also the IP addresses of the protected network have been changed in the trace.

As with the previous detect, my conclusion for the network topology is the following:

```
Internet -- Cisco:c0 ----- Cisco:33 -- 207.166.0.0/16
                             |
                           Snort
```

I believe now that all packets in the trace that have the IP total length value greater than 1500 bytes are corrupted. Therefore the network that Snort listens to does not necessarily have MTU value greater than 1500.

Furthermore, my conclusion with detect #1 was that every IP packet that has all of the following characteristics is corrupted:

- IP ID is zero
- IP TTL is 240
- IP checksum is zero
- TCP checksum is zero

This means that 1529 (10,18%) packets are corrupted. I assume in this analysis that the rest of the packets are intact.

There are no packets in the trace that would come behind the Cisco:c0 router and have a source address from 207.166.x.x address space. Similarly no packets with destination address within that address space are routed through Cisco:33 router. This IP/MAC address correlation may mean that there is some anti-spoofing technique in place in the network. It may also mean that no spoofing just happens during the time of observation. It is worth to note, however, that even if the incoming packets with the internal source IP

addresses are dropped in the perimeter router, the 255.255.255.255 address still seems to get through.

2. Detect was generated by

The trace is a binary Snort log file. Thus the detect was generated by Snort intrusion detection system. The exact rule that triggered the alert is unknown, as is the alert message. The detect has several characteristics that typically trigger alerts in intrusion detection systems, however. I'll explain these characteristics in the attack description chapter below.

3. Probability the source address was spoofed

The IP address of 255.255.255.255 is reserved for limited broadcast use and it should appear only as destination address [Stevens, p45]. Either some network device is broken or the packets are crafted. I lean strongly toward the latter theory.

The question of spoofed IP address is strongly tied to another question: whether the detect is a stimulus or a response. Let us speculate this shortly.

If the detect is a response, there must exists a stimulus that causes such a response. The detect has a unicast destination IP address. This address could have been used as source IP address in the stimulus. However, the detect has a broadcast IP address as source address. Normally no network device would send a packet out with this address.

Even if a network device would be broken and would use the stimulus' broadcast destination address as the source address in the response, there are other things that do not support that. The detect consists of TCP packets which have RST and ACK flags set. Furthermore, each packet has three bytes of TCP payload. Normally, whenever a TCP connection is terminated with RST, no payload is included to the packet that has the RST flag set. And as a final comment, all packets of the detect have the same IP ID number (zero), which is a strong sign of crafted packets.

The conclusion is that the detect is a stimulus and it uses spoofed IP address. It appears to come from the external network, according to the MAC addresses. Theoretically the packets could have been sent from the same segment where the Snort is listening and with spoofed source MAC addresses, but there is no evidence that would support this theory.

4. Description of attack

The packets were sent to 18 different host, all located in the protected network. None of the hosts received more than one packet. All packets were similar -- up to the IP ID number (zero). Every packet's TTL value is 15, which

indicates that most likely they have been sent by the same host. The packets are TCP RST packets with the destination port 515 (printer).

The first packet was detected at 12:43 and the last at 02:27. The shortest delay between two packets was 39 seconds and the longest was 3 hours and 6 minutes. No pattern was discovered from the packet timings.

The packets had several characteristics that made them as interesting detects:

1) The source IP address 255.255.255.255 is the first and foremost anomaly that got my attention. Since it breaks the IP standard, it has a strong probability of getting caught by an IDS. It is worth to notice that the packet has already penetrated through the external router, which indicates that there are no ingress filters to stop these packets.

2) The source TCP port 31337 deserves some attention, since many trojan horses, such as Back Orifice, use it. The purpose of using this port is unclear, however, since due to the spoofed source IP address the sender is unable to receive the returning packets.

3) The IP ID number remains unchanged (zero) in all of these packets.

4) There is the same payload in all of the packets (ASCII string “cko”) even though the TCP flag RST was set.

I was unable to find a known attack that would match the detect. I’ll speculate the purpose of these packets in the attack mechanism chapter below.

5. Attack mechanism

As I speculated above, I do not believe the detect being a response. And since the packets have the RST flag set, they should not generate any response either. I can think of three different purposes of why the packets may have been sent.

One possible purpose for the packets is just to generate noise to the network. This can serve multiple purposes: the security administrators may either disable the rule that generates a lot of alerts or at least they may get used to them. Additionally the noise may cause the defenders to look at the wrong direction when a real attack is executed somewhere else.

The source IP address and source port selections seem to support the noise theory, since I fail to see why they would benefit the attacker. If the purpose were to act silently, the attacker should have chosen less alarming values for these fields.

Another possible purpose is a denial-of-service attack against printer service. I couldn't find any printer vulnerability that would match the detect, but surely there might exist one that is not known.

The third possible purpose is to provide a communication path for a trojan horse program. It may be possible that someone is trying to establish a contact with a trojan horse program -- most likely in order to start sending commands to it. The big question of this theory is that how the trojan knows to whom it should answer, since the source IP address of the packet does not give any clue.

6. Correlations

I browsed through the incident.org's mail archive and found Les Gordon's (response) article [Gordon2] speculating similar detect. Les described how the remote-control software Q works and how covert channels can be used to send commands to the Q program.

Les pointed out that a trojan is able to interpret the packet in totally different way than a standard TCP/IP stack. He also speculated that the IP address where the trojan is asked to contact can be encoded to the IP or TCP header fields, such as sequence or acknowledge numbers.

In my detect the sequence and acknowledge numbers are zero in all packets. Also the IP ID numbers are zero. Thus it is not likely that they carry any hidden information. However, the three bytes in the payload may do so. Still, since the payload is only 24 bits long in these packets and the IP address is 32 bit long, some extra information may be needed.

I suppose it is possible to compress a 32-bit IP address to 24 bits or even to a smaller space, however. Not all 2^32 possible IP addresses are valid addresses anyways. Furthermore the trojan may carry a map with 2^24 addresses evenly distributed across the whole IP address space; the 24-bit key might be just a pointer to the right IP address.

This correlation showed that the trojan theory is a possible one. I select it as my candidate suspect, even though the other two theories (noise or DoS attack) are also possible.

7. Evidence of active targeting

No traffic was detected to these targets prior the detect. This is understandable, since none of the three theories of why the packets may have been sent require prior scanning. The most likely suspect, a trojan covert channel, is a scan itself looking for installed and active trojans.

I looked quickly the previous and next day's logs and found that the similar pattern with the source IP 255.255.255.255 repeated in them too.

I believe that the scan is not targeted specifically against the hosts in the 207.166.x.x address space, but that it is a generic scan in the Internet.

8. Severity

Here are my evaluations for the criticality, lethality and system and network countermeasures:

Criticality = 2. The scan is targeted evenly across the address space of the protected network. There is no evidence that these hosts would be particularly important servers. Typically trojan programs tend to infect workstations rather than servers, so that point also limits the criticality.

Lethality = 2. A normal scan would earn only 1 point here. The another point comes from the fact that since the possible trojan inside the protected network seems able to receive commands through a covert channel, we have to assume that an established connection between the trojan and its master may be covert too. The trace does not indicate such a connection, but it is fully possible that the Snort rulebase simply does not catch one.

System countermeasures = 4. A normal system should not be affected by these packets in any way. I decreased one point from the fear of an unknown DoS attack.

Network countermeasures = 1. These packets have clear anomalies and they should not be allowed to enter the network. The perimeter router has clearly failed to block them.

Severity = (2 + 2) – (4 + 1) = -1.

The detect gives no reasons for big worries. The network countermeasures may need a little boost, though.

9. Defensive recommendations

Since there is no evidence of trojan activity beside the external scanning, no immediate damage control is needed.

However, I recommend some tuning to the rulebase of the perimeter firewall/router. There is no need to allow packets with broadcast source addresses to pass the firewall. Furthermore, all unnecessary services should be denied. If there is no clear need to allow the external hosts to contact the protected networks' printer service, for instance, the perimeter defence should block all attempts to do so.

Unfortunately the covert channels are limited only by the imagination of their creators. If there is a trojan program inside, there are really no ways to

completely deny it communicating with the outside world. The obvious methods may be detectable and preventable, but the real solution is to try to prevent the trojans to enter the internal network in the first place.

10. Multiple choice test question

Which one of the following is a valid source IP address?

a) 10.0.0.255

b) 127.0.0.255

c) 240.0.0.255

d) 255.255.255.255

The answer is a. The 10.x.x.x is a private address space and its addresses can be used as host addresses. The address 10.0.0.255 is not a broadcast address if the netmask is larger than /24. The b) is loopback network, c) is multicast and d) limited broadcast.

Detect #3

ACK scan

```
03:17:52.834488 0:3:e3:d9:26:c0 0:0:c:4:b2:33 0800 60:
63.211.17.228.80 > 46.5.180.250.53: . ack 0 win 1400
```

1. Source of Trace

The source for this detect was found from the same place than the two previous ones: at incident.org's Snort binary log repository at <http://www.incidents.org/logs/Raw>. This time I selected almost half a year older log file 2002.5.5.

Again the README file located in the same directory with the log files warns that the IP addresses of the protected network have been modified and therefore the IP, TCP and UDP checksums are not correct. It seemed that the addresses were modified differently back in June 2002 than they were in October 2002 when the trace of my previous two detects was captured. The MAC addresses seemed to be the same which makes me to assume that the network topology where the Snort instance runs has not changed.

Using the same techniques as with the previous detects I got the following network diagram. This time the address space for the protected network is 46.5.x.x.

```
Internet -- Cisco:c0 ----- Cisco:33 – 46.5.0.0/16
                          |
                        Snort
```

I did not see any packets with source IP address of 46.5.x.x coming from the Internet side; nor did I see any packets with source IP anything else than 46.5.x.x leaving the internal network. Again, this does not proof that there would be some anti-spoofing device in place; it only indicates that no spoofing happened that day. It is worth to notice, however, that some spoofed packets were detected in the trace: there are 34 packets with the source IP address of 255.255.255.255. I covered these packets in the previous detect -- here I only remark that this anomaly seems to be constant.

One outbound packet has source IP address 46.5.180.133. All other outbound packets have source address 46.5.180.250. This is different than with the previous detects when there was only a single address that the outbound packets shared as the source address.

Using the same criteria as with the detect #2 above, I dismissed all packets that appeared to be corrupted. All packets that had TTL 240, IP ID zero and both IP and TCP checksums zero are assumed to be corrupted. This also removed all packets supposedly bigger than 1500 bytes.

There appeared to be 2558 corrupted packets out of 5135 packets in the trace. This makes the corruption percentage awfully high: 49,8%. I have never seen this high corruption with any network card; however, since I do not have better guesses I have to stand with the assumption that this is the case.

The single packet with source IP 46.5.180.133 has TTL value of 63, TOS 0x00, non-zero IP ID and window size 32120. The packets with source IP 46.5.180.250 have TTL values 123-125 (also 240 among the corrupted packets), multiple kind of TOS values, non-zero IP ID values and windows sizes 0-65535.

Since it is hard to believe that the routing would change within the internal network for a single host, I assume that the address 46.5.180.250 is used for a hide address in a NAT device that is located between the protected network and the Snort. The rules of the NAT device either do not change the source address of 46.5.180.133 or that address is another hide address used less often than the .250 address. It may also be possible that the host with 46.5.180.133 address somehow is able to communicate with the Internet without routing its traffic through the NAT device -- unlike all other hosts.

2. Detect was generated by

The detect was generated by Snort intrusion detection system running in binary logging mode. Since the log data is in standard pcap format, it can be read with multiple programs, including tcpdump and ethereal.

The following is a tcpdump output of the detect:

```
03:17:52.834488 0:3:e3:d9:26:c0 0:0:c:4:b2:33 0800 60:
63.211.17.228.80 > 46.5.180.250.53: . ack 0 win 1400
      4500 0028 324e 0000 3106 28d2 3fd3 11e4
      2e05 b4fa 0050 0035 0000 0398 0000 0000
      5010 0578 778f 0000 0000 0000 0000
03:17:52.834488 0:3:e3:d9:26:c0 0:0:c:4:b2:33 0800 60:
63.211.17.228.53 > 46.5.180.250.53: . ack 0 win 1400
      4500 0028 324f 0000 3106 28d1 3fd3 11e4
      2e05 b4fa 0035 0035 0000 0399 0000 0000
      5010 0578 77a9 0000 0000 0000 0000
03:17:52.944488 0:3:e3:d9:26:c0 0:0:c:4:b2:33 0800 60:
64.152.70.68.80 > 46.5.180.250.53: . ack 0 win 1400
      4500 0028 af8e 0000 3106 766c 4098 4644
      2e05 b4fa 0050 0035 0000 0215 0000 0000
      5010 0578 43ed 0000 0000 0000 0000
<snip>
```

I snipped the trace after the first three packets to save space; there are total of 29 similar events in the logs: 9 with the destination port of 53, three destination ports 4100 and 17 destination ports 80. The anomaly in all of the packets is that the ACK flag is set but the acknowledge number is zero (bolded in the trace).

3. Probability the source address was spoofed

Let us have a closer look at the source addresses of the packets and try to correlate them somehow. According to the whois databases (arin, ripe and apnic) the packets came from the following organizations/countries:

- First three packets, captured at 03:17:52, came from addresses belonging to Level 3 Communications, Inc., Broomfield, CO, US.

- The next six packets, captured at 08:35:05, came from addresses belonging to UUNET Technologies, Inc, Ashburn, VA, US.

- The next three packets were captured at 15:19:54 – 15:19:55 and they came from Israel. (Bank Leumi Of Israel, NetVision Ltd, Israel)

- The next two packets were captured at 15:40:41 and they came from Sollac, France and USINOR TI, France.

- The next five packets, captured at 20:37:04 – 20:37:31 came from Taiwan (Chunghwa Telecom and NCEN)

- Also the last ten packets came from Taiwan (CCEN), the first six captured at 02:26:43 – 02:27:10 and the last four at 02:50:10 – 02:50:19.

The packets surely do not look like valid TCP packets. The ACK flag is set in all of them, but the acknowledge field value is zero. According to RFC793, whenever the ACK flag is set the acknowledge field tells the next sequence number the sender of this packet expects to receive. Theoretically this value could be zero, but in is terribly unlikely for that to happen in all of these cases.

Furthermore, all of the packets have a very low sequence number value. While all values are valid for sequence numbers, the initial numbers should be selected randomly. It is extremely unlikely that all of these connections would have sequence numbers randomly selected as low as they are in these packets.

The TCP ports are also odd. The only source/destination port combinations that make sense are 80/4100 and 81/4100; the port 4100 would then be the ephemeral port. Even then it is very unlikely that the internal client (or the intermediate NAT device) would have selected the same source port for two different connections.

All these facts combined are strong evidence that the packets have been spoofed. But I have another theory too that might explain the behaviour.

Let us suppose that some ISP load balancer device has not been designed correctly and that it probes the ISP links by sending a crafted packet through all of them. Let us further suppose that all of the places mentioned above

might use this device to balance their outbound traffic between multiple ISP links. Now if one of their clients wanted to connect a computer in the University, the balancer may send a crafted IP packet -- like the ones captured in this detect -- to the server simultaneously through all of the ISPs available. The fact that similar packets are captured approximately at the same time coming from the same general area seems to support this theory.

I agree that this would not be the smartest way to probe the ISP links. I do not know of any ISP balancer that would work like this; still I have seen my share of oddly behaving network devices and I know that this could be possible.

The timing of the packets in the capture makes me to assume the ISP balancer theory and therefore I do not believe that the packets would be spoofed (except by the balancer).

4. Description of attack

If my theory of the ISP load balancers is right, the detect is not an attack but traffic from poorly configured, designed or implemented network devices.

5. Attack mechanism

I'll try to speculate here why I do not believe the detect to be an attack.

The destination ports 53 and 80 are common attack targets since the applications listening to these ports have a long history of having severe vulnerabilities. However, none of the packets have any payload, so they cannot be targeted against the applications.

While the packets have odd sequence numbers, acknowledge numbers and source/destination port pairs, strictly speaking they do not break TCP or IP standards. Therefore they should not cause denial-of-service to any TCP/IP stack.

The only relevant reason I can think of why an attacker may have send these packets is port scanning. The ACK scan is a well-known but unreliable way to determine the open and closed ports on certain operating systems. This theory requires that the attacker does not spoof the addresses he uses, since he needs to see the possible returning packets.

However, the detect consists of groups of packets, each packet in one group sent almost simultaneously and from the same general area. Furthermore, the scans seem to come from all around the world: US, Europe, Middle East and Asia. It would be more understandable if there were some single ACK scans; the fact that all scans come in groups makes me to doubt the port scan theory.

6. Correlations

I have not seen an ISP balancer to behave like this. However, I have seen other network devices to behave oddly in certain situations.

Symantec's Raptor firewall, for instance, when configured to prevent TCP SYN flooding attacks sent odd packets to the network in order to trying to determine whether the sender of the SYN packet is an alive host or not.

Radware's Linkproof ISP balancer -- at least in the past times -- sent ICMP echo requests to probe the ISP availability.

The detect here certainly does not match these two devices or at least to the behaviour I have experienced with them. Still they show that some devices send odd packets even when they work like expected.

7. Evidence of active targeting

Since there is no attack, there cannot be active targeting either. The packets have been sent to their targets purposefully, but not with a malicious purpose.

8. Severity

Here are my evaluations for the criticality, lethality and system and network countermeasures:

Criticality = 2. The packets are targeted against the servers of the protected network. There is no evidence that these services would actually be listening on the target hosts, the TCP packets do not indicate that a connection would have been established.

Lethality = 1. Even at the worst case the packets would be performing port scanning. And I believe them to be non-malicious.

System countermeasures = 5. A normal system should not be affected by these packets in any way.

Network countermeasures = 1. A statefull firewall should drop the TCP packets that claim to belong to a non-existing connection. Since Snort has captured the packets, the perimeter defence has failed this task.

Severity = (2 + 1) – (5 + 1) = -3.

Not a big deal. I recommend using a statefull firewall as the perimeter defence, though.

9. Defensive recommendations

As already stated, my recommendation is to implement tighter perimeter defence, using for instance a statefull firewall that would drop all odd and out-of-connection packets out.

10. Multiple choice test question

The acknowledge number is always zero if

a) the TCP connection has been established but no data has been transferred

b) both SYN and ACK flags are set

c) the TCP packet is being retransmitted

d) None of the answers above

The correct answer is d.

Assignment 3 - Analyze This

Files used in the analysis

alert.030401.gz	scans.030401.gz	OOS_Report_2003_04_02_24924 (contains oos logs for 2003-04-01)
alert.030402.gz	scans.030402.gz	OOS_Report_2003_04_03_9924 (contains oos logs for 2003-04-02)
alert.030403.gz	scans.030403.gz	OOS_Report_2003_04_04_29217 (contains oos logs for 2003-04-03)
alert.030404.gz	scans.030404.gz	OOS_Report_2003_04_05_3459 (contains oos logs for 2003-04-04)
alert.030405.gz	scans.030405.gz	OOS_Report_2003_04_07_31826 (contains oos logs for 2003-04-05)

Executive Summary and Defensive Recommendations

The University's IDS produces a lot of logs: 237.332 alerts, 1.416.554 scans and 9.396 out-of-spec packets were collected during five days from 1 April through 5 April 2003. These logs are analyzed here.

A great deal of the logs appeared to be false alarms. Specific recommendations to fine-tune the IDS rules are given in the specific analysis of individual alerts. Still, the University seems to be under constant hammering from the Internet, which the logs clearly show.

Unfortunately, a few hosts in the University look so suspicious, that they have to be assumed to be compromised. The reasons to believe that are detailed in the individual alert analysis below. A strong recommendation is to pay immediate attention to the hosts involved. The security holes in them must be fixed and possible intruders driven out before they can do more harm to the University's network.

Some peer-to-peer program activity was also detected in the network. The dangers involved with such a programs are shortly listed in the peer-to-peer – chapter below. A link to an excellent article about the subject is also provided there.

It seems clear that the University does not have a properly configured perimeter firewall. A good advice and a strong recommendation is to enforce all traffic between the University and the Internet through a firewall with a tight policy to deny everything that is not explicitly allowed. This would serve several purposes. First – it would reduce the amount of false positives of the IDS dramatically. Secondly, a good preventive mechanism is a lot cheaper to maintain than detecting and reacting to incidents that are executed.

The hosts that require immediate attention are the following:

MY.NET.97.66

MY.NET.97.88
MY.NET.98.157
MY.NET.98.184
MY.NET.84.235
MY.NET.105.48
MY.NET.252.166
MY.NET.240.78

The following hosts should also be inspected:

MY.NET.6.63
MY.NET.88.193
MY.NET.105.48
MY.NET.194.223
MY.NET.197.2
MY.NET.203.86
MY.NET.223.46
MY.NET.240.246

Alert Summary

I found an excellent alert summary table used in Les M. Gordon's practical [Gordon] and decided to use the same approach in my work. Les also provided a perl script to help construct the summary table.

All alerts are listed below, in the decreasing order of how many times they occurred during the five days. The count of IP addresses involved is divided to external sources, internal destinations, internal sources and external destinations. The internal addresses contain all addresses in the University's MY.NET.0.0/16 address space. Finally the number of attacks are listed on basis of the direction of the attack: inbound, outbound, only internal or only external.

Alert name	Alerts	Ext Src	Int Dst	Int Src	Ext Dst	Inbound	Outbound	I->I	E->E
SMB Name Wildcard	109244	22873	35727		1	109243			1
Watchlist 000220 IL-ISDNNET-990517	39358	251	324			39358			
High port 65535 udp - possible Red Worm - traffic	18176	153	93	61	157	9459	8717		
spp_http_decode: IIS Unicode attack detected	14163	240	212	525	692	614	13549		
CS WEBSERVER - external web traffic	9393	4262	1		1	9392			1
High port 65535 tcp - possible Red Worm - traffic	8452	72	47	51	72	2263	6189		
Tiny Fragments - Possible Hostile Activity	7587	7	7	1	237	104	7483		
Incomplete Packet Fragments Discarded	5567	64	48	4	5	468	5098	1	
TFTP - Internal TCP connection to external tftp server	3963	30	6	6	28	2036	1927		
TCP SRC and DST outside network	3949	1928			81				3949
MY.NET.30.4 activity	3086	257	1			3086			

External RPC call	2803	16	2748			2803			
spp_http_decode: CGI Null Byte attack detected	2711	15	2	112	117	22	2689		
Null scan!	1609	68	78			1609			
Watchlist 000222 NET-NCFC	1254	90	40			1254			
Queso fingerprint	1174	292	109			1174			
IDS552/web-iis_IIS ISAPI Overflow ida nosize	854	608	700			854			
SUNRPC highport access!	689	65	43			689			
Possible trojan server activity	548	58	30	14	17	134	414		
EXPLOIT x86 NOOP	472	116	95			472			
CS WEBSERVER - external ftp traffic	307	115	1			307			
IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize	305			4	264		305		
MY.NET.30.3 activity	220	44	1			220			
NMAP TCP ping!	214	45	76			214			
connect to 515 from outside	181	24	3			181			
DDOS mstream handler to client	170			1	2		170		
SNMP public access	128	14	13			128			
EXPLOIT x86 setuid 0	117	107	94			117			
NIMDA - Attempt to execute cmd from campus host	114			4	111		114		
IRC evil - running XDCC	85			20	22		85		
EXPLOIT x86 setgid 0	79	73	69			79			
TFTP - Internal UDP connection to external tftp server	75	6	11	9	4	31	44		
EXPLOIT x86 stealth noop	74	16	15			74			
DDOS mstream client to handler	71	27	3			71			
TFTP - External TCP connection to internal tftp server	26	12	1	1	14	12	14		
Notify Brian B. 3.54 tcp	22	19	1			22			
Notify Brian B. 3.56 tcp	22	21	1			22			
Attempted Sun RPC high port access	16	4	3			16			
SMB C access	11	10	8			11			
NETBIOS NT NULL session	8	1	5			8			
FTP passwd attempt	7	7	1			7			
TCP SMTP Source Port traffic	7	1	2			7			
Probable NMAP fingerprint attempt	5	5	5			5			
EXPLOIT NTPDX buffer overflow	3	3	3			3			
EXPLOIT digital unix noop	2	1	1			2			
NIMDA - Attempt to execute root from campus host	2			2	2		2		
RFB - Possible WinVNC - 010708-1	2	1	1	1	1	1	1		
SYN-FIN scan!	2	1	1			2			
Back Orifice	1	1	1			1			
Bugbear@MM virus in SMTP	1					1			
DDOS shaft client to handler	1	1	1			1			
External FTP to HelpDesk MY.NET.53.29	1	1	1			1			

Trin00 password on tcp	1	1	1			1			
Totals:	**237332**	**32027**	**40635**	**816**	**1828**	**186579**	**46801**	**1**	**3951**

Most Frequent Alerts (over 5.000 during five days)

1. SMB Name Wildcard, 109.244 alerts (46,03% of all alerts)
Snort rule: Custom

Since I did not find a Snort rule with this message among the standard or experimental Snort rulebase, I do not know what traffic the rule matches. I assume, though, that all NetBIOS queries trigger the alarm. Furthermore, since all packets have an external source IP address, I assume that the internal NetBIOS queries do not trigger the alert.

Les M. Gordon [Gordon] noticed in his practical that this alert occurred 58.295 times during a six-day period in June 2002. Johnny Calhoun [Calhoun] counted 61.249 alerts during five days in January 2003. Back in June 2002 the majority of the traffic was internal; In January 2003 most of the traffic originated from outside. The trend clearly seems to be that the external scans have increased dramatically in time. There are a few reports of vulnerabilities in Microsoft Windows and Samba that might explain the increased interest to these services:

CERT vulnerability note VU#958321, Samba contains a remotely exploitable stack buffer overflow, published 13 Dec 2002
http://www.kb.cert.org/vuls/id/958321
Also CAN-2002-1318

CERT advisory CA-2003-03, Buffer overflow in Windows locator service, published 23 Jan 2003
http://www.cert.org/advisories/CA-2003-03.html
Also CAN-2003-0003

CERT advisory CA-2003-08, Increased activity targeting Windows shares, published 11 March 2003
http://www.cert.org/advisories/CA-2003-08.html

CERT vulnerability note VU#298233, Samba contains buffer overflow in SMB/CIFS packet fragment reassembly code, published 17 March 2003
http://www.kb.cert.org/vuls/id/298233
Also CAN-2003-0085

CERT vulnerability note VU#267873, Samba contains multiple buffer overflows, published 10 Apr 2003
http://www.kb.cert.org/vuls/id/267873
Also CAN-2003-0201, CAN-2003-0196

The last note does not explain the alerts of the selected dates, since the note was published after they occurred. It may affect even more scans in the future, however.

Recommendations:

Block the incoming NetBIOS traffic in the perimeter firewall. If the NetBIOS is needed with some peers, the recommended way is to build a VPN tunnel between the University and those sites. Ensure that all hosts with samba service have the newest samba software -- the older ones have serious vulnerabilities. Finally ensure that the password policy used with the samba shares requires strong passwords. The CERT advisory CA-2003-08 tells about a worm that spreads by guessing samba passwords.

2. Watchlist 000220 IL-ISDNNET-990517, 39.358 alerts (16,58%)
Snort rule: Custom

This is a custom alert. Based on the traffic that has triggered the alert, it seems to capture traffic that comes from 212.179.0.0/16 network. The port numbers indicate that the majority of the traffic is KaZaA, web browsing and traffic to ports 3162 and 3249. These last two ports do not map to well-known services, which makes me to think that probably they are customized peer-to-peer software ports or even customized trojan horse ports.

Since I do not for what purpose the rule has been created, I find it difficult to evaluate how serious the alerts are.

Recommendations:

If the Israel ISP, whose IP addresses these are, does not need access to the University's network, block them out in the perimeter firewall. If they do need access, limit it to the minimum with the same firewall. This should reduce the amount of logs significantly.

3. High port 65535 udp - possible Red Worm - traffic, 18176 alerts (7,66%)
Snort rule: Custom

Red Worm, also known as Adore, spreads using vulnerabilities in LPRng, rpc-statd and BIND. Once it has infected a host, it opens a backdoor to listen TCP port 65535. A good analysis of the worm is available at the SANS's reading room by J. Anthony Dell [Dell].

This rule watches for UDP traffic, not TCP, and therefore cannot detect the Red Worm. The amount of traffic having UDP port 65535 is so high, however, that it is hard to believe that it has happened to be used as random ephemeral port this often. There are 9459 alerts for inbound traffic and 8717 for outbound.

The majority (89%) of these packets are sent or received by MY.NET.201.58. This host seems to communicate a lot with the external hosts 66.42.68.210 and 4.46.32.83 (ports 5121 and 5122) and host 68.110.8.98 (port 4121). The UDP ports 5121 and 5122 are used by a Neverwinter Nights –game, made by BioWare Corp. [Bioware]. My guess is that this game triggers the majority of the alerts. I am not sure, though, that what application uses UDP port 4121; the IP address involved with that traffic belongs to Cox Communications Inc, Atlanta, US.

Recommendations:

The alert name is misleading. Since the alert does not capture Red Worm traffic, its purpose is questionable. My recommendation is to remove it.

4. spp_http_decode: IIS Unicode attack detected, 14163 alerts (5,97%)
Snort rule: snort preprocessor http_decode

As Tod Beardsley [Beardsley] described in his practical, this alert is generated by Snort http_decode preprocessor when it sees unicode encoded '\', '/' or '.' characters on common http ports. The idea is that Microsoft's Internet Information Server (IIS) has a sad history of accepting requests encoded in unicode that it would not accept otherwise. Tod concludes in his paper that Nimda and other worms have infected multiple hosts in the University.

However, Les Gordon [Gordon] points out in his practical that the extended character sets used in Korea, Japan and some other sites trigger this alert as well. It turned out that approximately 70% of the alerts actually had the destination address in Korea, China or Taiwan.

Michael Wisener [Wisener] found a posting on snort-users mailing list, sent by Joe Steward [Steward], that many of these alerts may be due to URL encoded binary data that the website cookies send.

Given these facts the alerts do not seem to be reliable. I give the same recommendation as Michael and others to turn off the unicode checking from the snort http preprocessor, as explained in the Snort FAQ [snort-faq].

5. CS WEBSERVER – external web traffic, 9393 alerts (3,96%)
Snort rule: Custom

This rule seemed to trigger alert for all http traffic that comes from the external network. All of the packets shared the same internal destination: MY.NET.100.165. 1306 packets (13,9%) came from a single source address 216.39.48.2 that belongs to AltaVista Company. My guess is that AltaVista is updating their web search databases.

It is interesting to notice that this internal host has drawn quite a lot external fire. In addition to the 9392 web traffic alerts it received 307 packets with "CS WEBSERVER - external ftp traffic" alerts, 67 "SMB Name Wildcard" alerts, 101 "Watchlist 000222" and 61 "Watchlist 000220" alerts, 8 IIS Unicode alerts,

6 Queso Fingerprints, 4 NMAP TCP pings, a single IIS ISAPI overflow alert, a single TCP high port 65535 alert, some scanning and total of 23 out-of-spec packets.

Recommendations:

None of the alerts, scans or OOS packets are particularly alarming, though. If the host's purpose is to be an external web server, it will for sure get more attention than an average host -- exactly what happens here. I recommend to check that the host has all of the available patches installed, however, and that it is configured properly. Furthermore, some of the alerts should never be triggered, since a properly configured perimeter firewall's task is to drop all unnecessary connection attempts and single packets before they arrive to the internal network.

6. High port 65535 tcp - possible Red Worm - traffic, 8452 alerts (3,56%)
Snort rule: Custom

As noted with UDP 65535 above, the Red Worm uses TCP, not UDP, for the backdoor communication protocol. These backdoor connection attempts should be captured by this rule.

The majority of the alerts here, however, show connections that have TCP port 65535 bound to the external host's address and the MY.NET hosts use some other port. I looked for connections where the external host would have a likely ephemeral port (1024 or higher) and the internal host has the port 65535. I found eight hosts that fit this criteria:

MY.NET.6.63
MY.NET.88.193
MY.NET.105.48
MY.NET.194.223
MY.NET.197.2
MY.NET.203.86
MY.NET.223.46
MY.NET.240.246

My recommendation is to have a look of these hosts for possible signs of Red Worm infection.

7. Tiny Fragments – Possible Hostile Activity, 7587 alerts (3,20%)
Snort rule: SID 522 (the closest match)

This rule triggers an alert for all packets that have the MF (more fragments) flag set and are smaller than 25 bytes. This might happen if a packet gets fragmented multiple times during its travel. However, more often than not the small fragments are signs of some kind of evasion or other malicious activity. Therefore these alerts require attention.

The vast majority of the alerts are outbound – and from a single host: MY.NET.240.78. This host alone sent 7483 tiny fragments to 237 different

hosts. The same host also scanned 874 hosts and used 228 different scan types to do that!

104 inbound packets triggered the same rule. This is not alarming, since it can be explained as the Internet noise. The fragments themselves should not be able to damage the recent versions of operating systems; their primary function typically is to evade older firewalls or intrusion detection systems.

Recommendations:

Investigate the internal host MY.NET.240.78 for possible compromise. Implement a perimeter firewall to drop all malicious and incomplete fragments before they enter the protected network.

8. Incomplete Packet Fragments Discarded, 5567 alerts (2,35%)
Snort rule: Custom

This rule catches the fragmented packets that Snort is unable to reassemble due to a missing fragment. There may be several reasons for this: the missing fragment may have been lost in the network, it may have been routed differently, or the IDS may have been too busy and missed the fragment for that reason. It is also possible that the incomplete fragments have been send purposefully – maybe to launch a DoS attack or just to generate noise to the network. A conservative amount of these packets would be understandable as normal background noise. However, 5060 packets (90,9%) are sent by a single host MY.NET.252.166 for a single destination 63.210.47.23. I was unable to guess the purpose for these packets, therefore an investigation of the internal host is recommended.

Other Interesting Alerts

9. TCP SRC and DST outside network, 3949 alerts
Snort rule: Custom

This rule alerts whenever a packet is seen that has both source and destination address outside of the MY.NET address space. Let us look at the packets that have triggered the alert.

255 packets (6,5%) have either source or destination address from a private IP address space. My assumption is that there are hosts in the University that have been configured to use private addresses. Either there is a device doing NAT for these addresses before they leave the University network, or the hosts have been configured wrongly.

Furthermore, 1745 packets (44,2%) have the source IP address of 207.46.134.190 and the destination address of 216.251.32.98. The source address resolves to microsoft.com; it actually appears to be Microsoft's main web site address. The destination address belongs to an ISP named InternetNamesForBusiness.com. Since there should be no reason why

Microsoft would route these packets through the University, I suspect that the source address is spoofed.

Recommendations:

If the private addresses are valid addresses in the network, modify the snort rule to recognize them as internal addresses.

Consult the snort binary log to find the MAC address of the spoofed packets. This gives indications of what host has sent packets with external source address. The investigation may require consulting of other logs too, such as the MAC address tables of the switches in the network.

Implement anti-spoofing rules to the perimeter firewall / router to block all outbound packets that do not have source IP address belonging to the University's address space. In the same place, block all incoming packets whose source address is within the University's address space. Make sure that all traffic between the University and the Internet goes through the firewall.

10. IDS552/web-iis_IIS ISAPI Overflow ida nosize, 854 alerts
Snort rule: Custom

This snort rule is taken from the whitehats.com's web site, to capture exploits against Microsoft Internet Information Server.

Total of 700 internal hosts have received the exploit attempt from a total of 608 external hosts. No evidence was found that any of the attempts would have lead to a compromise.

Recommended actions:

These alerts are pure noise when targeted against servers that have been patched against the exploit. I recommend to check the patch levels of all servers through the regular auditing processes and to ensure that the perimeter firewall accepts only the desired connections to the internal network.

11. IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize, 305 alerts
Snort rule: Custom

No doubt, this rule is a modified version of the previous one to capture the outbound exploits against the IIS ISAPI extension. It is also a lot more interesting, since it alarms of the malicious activity launched from the internal network.

Four internal hosts were found to send packets that triggered this alert. The hosts are:

Count	IP

219	MY.NET.97.66
49	MY.NET.97.88
23	MY.NET.98.157
14	MY.NET.98.184

Recommended actions:

Check out the internal hosts involved for possible compromise or other misuse.

12. Possible trojan server activity, 548 alerts
Snort rule: Custom

This rule seemed to trigger whenever either the source or destination port was 27374. According to [doshelp] this port is associated with Sub-7 trojan horse.

The vast majority of the alerts turned out to be false alarms, where the port 27374 was used as normal ephemeral port. A single internal host, however, was connected 25 times to the port in question and may therefore require some further investigation.

Recommendations:

Inspect the host MY.NET.105.48 for possible compromise. Block all unnecessary ports in the perimeter firewall.

13. DDOS mstream handler to client, 170 alerts
Snort rule: SID 248, SID 250

This rule is intended to catch the communication from an mstream handler program to an attacker's client host. The mstream is a distributed denial-of-service program suite that consists of a client host(s), handler programs and agent programs. The clients send commands to the handlers whose task is to forward them to the agents. The agents are able to generate a large amount of network traffic that is used to paralyze the victim the agents are attacking.

A good article about the mstream is written by David Dittrich et al [Dittrich].

One internal host seems to be involved with the mstream: MY.NET.84.235. There are various versions of mstream around and they use different ports; an interesting detail is that this particular host triggers the alert with two different mstream ports: 12754 (140 counts) and 15104 (30 counts).

Recommended actions:

Check out this host for possible infection. Again, block all unnecessary ports in the perimeter firewall.

14. DDOS mstream client to handler, 71 alerts
Snort rule: SID 247, SID 249

This is a reverse rule to the one above; it catches the communication from the attacker's client to the mstream handler program.

Three internal hosts have triggered this alert. However, one of them seems to be a clear false alarm. The other two consist of MY.NET.84.235 that triggered also the previous alert, as well as MY.NET.105.48.

I recommend checking out the host MY.NET.105.48 in addition to the other host that was already mentioned with the previous alert.

15. NIMDA – Attempt to execute cmd from campus host, 114 alerts
Snort rule: Custom

This is also an interesting detect, since it claims to find malicious activity launched from the internal network.

Four internal hosts triggered the alert. The hosts are MY.NET.97.66 (86 alerts), MY.NET.98.157 (12 alerts), MY.NET.97.88 (10 alerts) and MY.NET.98.184 (6 alerts). They appear to be the same hosts that triggered the ISAPI Overflow ida INTERNAL nosize –alert above. In addition, two of them (98.157 and 97.66) also generated another NIMDA alert: Attempt to execute root from campus host.

Recommended actions:

These four hosts are highly suspicious and should be inspected immediately.

Top Talkers

The following tables list the top talkers. The 'Combined Top 10' –tables list hosts with the most alerts, scans and OOS events combined to a single number, further categorized as internal sources, external sources, internal destinations and external destinations.

Combined Top 10 Internal Src

Count	IP
546219	MY.NET.132.23
348775	MY.NET.210.182
283439	MY.NET.178.19
23339	MY.NET.97.66
9522	MY.NET.240.78
7886	MY.NET.201.58
7321	MY.NET.97.88
6906	MY.NET.97.202
6473	MY.NET.88.229
6327	MY.NET.98.126

Combined Top 10 External Src

Count	IP
32156	218.68.216.47
9907	212.179.101.68
9122	212.179.48.2
6992	172.186.87.8
5655	217.21.114.148
5393	66.42.68.210
5241	217.21.114.154
4055	63.250.195.10
3607	62.65.192.30
3544	216.173.52.200

Combined Top 10 Internal Dst

Count	IP

10012	MY.NET.100.165
9951	MY.NET.207.194
8918	MY.NET.220.150
8540	MY.NET.201.58
3236	MY.NET.195.67
3090	MY.NET.30.4
2397	MY.NET.217.38
2271	MY.NET.236.42
1900	MY.NET.196.161
1783	MY.NET.6.7

Count	IP
822419	157.156.0.106
7239	200.43.46.36
6768	139.55.51.186
5060	63.210.47.23
5056	66.42.68.210
4716	68.97.167.18
4326	208.63.167.105
4113	67.35.48.144
3578	65.80.69.96
3364	139.142.201.55

Combined Top 10 External Dst

The following tables list the top internal and external scanners:

Top 10 Internal Scanners

Count	IP
546203	MY.NET.132.23
348775	MY.NET.210.182
283433	MY.NET.178.19
23033	MY.NET.97.66
7224	MY.NET.97.88
6874	MY.NET.97.202
6345	MY.NET.88.229
6327	MY.NET.98.126
5539	MY.NET.194.223
5049	MY.NET.204.110

Top 10 External Scanners

Count	IP
32155	218.68.216.47
6992	172.186.87.8
5655	217.21.114.148
5235	217.21.114.154
4049	63.250.195.10
3543	216.173.52.200
2835	217.59.215.194
2653	61.133.3.146
2076	64.5.44.143
1795	62.65.192.30

Here are the top internal and external host sending packets that generated alerts:

Top 10 Internal Host generating alerts

Count	IP
7886	MY.NET.201.58
7483	MY.NET.240.78
5061	MY.NET.252.166
2501	MY.NET.224.90
1487	MY.NET.86.110
1418	MY.NET.226.206
1088	MY.NET.54.210
770	MY.NET.97.48
662	MY.NET.153.145
628	MY.NET.152.20

Top 10 External Host generating alerts

Count	IP
9907	212.179.101.68
9122	212.179.48.2
5393	66.42.68.210
2384	212.179.102.138
2291	24.66.182.171
2063	4.46.32.83
1812	62.65.192.30
1783	212.179.85.46
1745	207.46.134.190
1534	212.179.35.118

Here are the top internal and external OOS-packet senders:

Top 10 Internal OOS senders

Count	IP
49	MY.NET.12.4

8	MY.NET.12.2
7	MY.NET.236.186
2	MY.NET.83.161
2	MY.NET.253.2

2	MY.NET.104.113
1	MY.NET.252.14
1	MY.NET.183.31
0	
0	

Top 10 External OOS senders

Count	IP
1099	68.54.93.181
612	209.191.132.40
356	66.140.25.157
162	148.63.151.3
159	61.114.222.241
152	62.142.15.248
147	212.244.86.66
136	80.26.5.150
136	216.95.201.23
119	216.95.201.34

Peer-to-Peer Users

The following is a list of probable peer-to-peer (P-P) file sharing users. These programs are used to search and download files, usually illegal copies of software, music and video, across the P-P network. Additionally, these programs have had a sad history of containing several security holes, as well as them have been used to spread worms and trojan programs. A good article of the dangers of the P-P programs can be read by Michael Hurwicz at Network Magazine [Hurwicz].

Gnutella users:

IP	Count
MY.NET.211.26	18
MY.NET.247.94	3
MY.NET.194.223	2
MY.NET.252.78	1
MY.NET.222.98	1
MY.NET.207.10	1
MY.NET.195.67	1

KaZaA users:

IP	Count
MY.NET.235.202	206
MY.NET.228.126	26
MY.NET.226.162	25
MY.NET.217.222	14
MY.NET.250.226	13
MY.NET.225.142	11
MY.NET.210.190	9
MY.NET.233.146	8
MY.NET.193.1	8
MY.NET.224.154	7
MY.NET.206.254	6
MY.NET.247.174	5
MY.NET.237.114	5
MY.NET.233.146	5
MY.NET.217.190	5
MY.NET.207.198	5
MY.NET.233.46	4
MY.NET.211.82	3
MY.NET.88.225	2
MY.NET.250.186	2
MY.NET.239.46	2
MY.NET.228.70	2
MY.NET.194.13	2
MY.NET.97.103	1
MY.NET.249.134	1
MY.NET.235.122	1
MY.NET.233.146	1
MY.NET.233.146	1

Napster users:

IP	Count
MY.NET.244.182	20
MY.NET.253.102	18
MY.NET.205.198	6
MY.NET.208.66	5

MY.NET.206.74	2
MY.NET.253.106	1
MY.NET.241.78	1
MY.NET.208.186	1
MY.NET.201.218	1

Link Graph

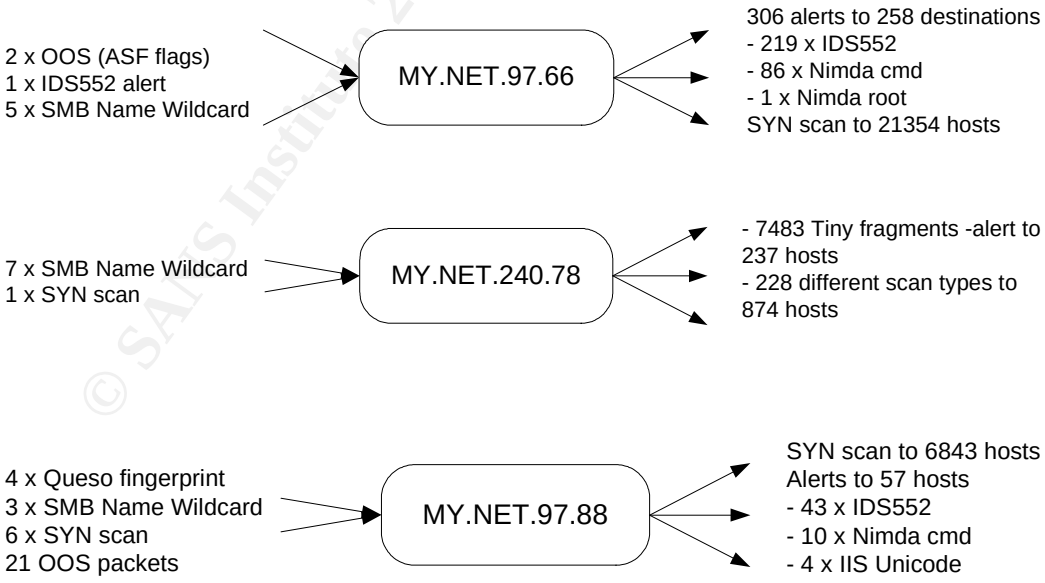
I decided to study the hosts that triggered the highest total amount of alerts, scans and OOS reports, either as sender or receiver. Since one internal and one external host happened to be active in both sending and receiving these events of interest, I ended up with 19 internal and 19 external hosts out of my four Top 10 Combined –tables.

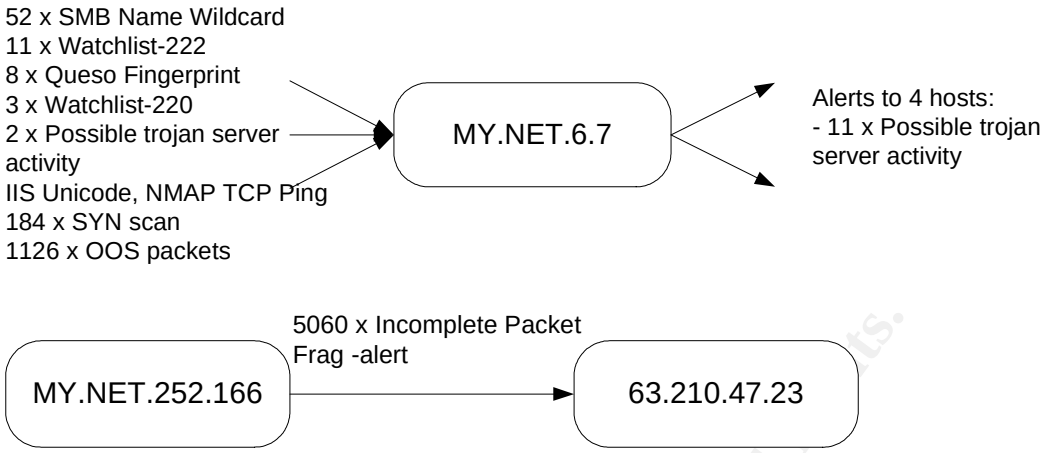
I found the link graphs I drew as useful tools during my analysis. They helped me to correlate the major portion of the logs together by showing the players who created the logs. The contents of the graphs are analyzed above and I do not repeat the work here; however I first made the graphs and then concluded the analysis using the information visible in them.

It is important to understand that only the hosts generating a lot of logs ended to these graphs. They do not necessarily represent the most alarming cases; those I have analyzed above with the interesting alerts.

Graphs 1-5: Suspicious internal hosts

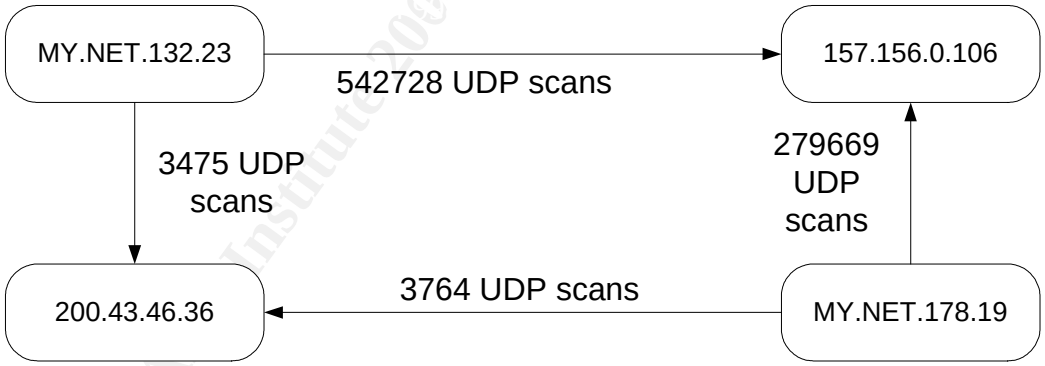
These internal hosts have sent a lot of suspicious traffic and therefore should be inspected for possible compromise or other misuse.

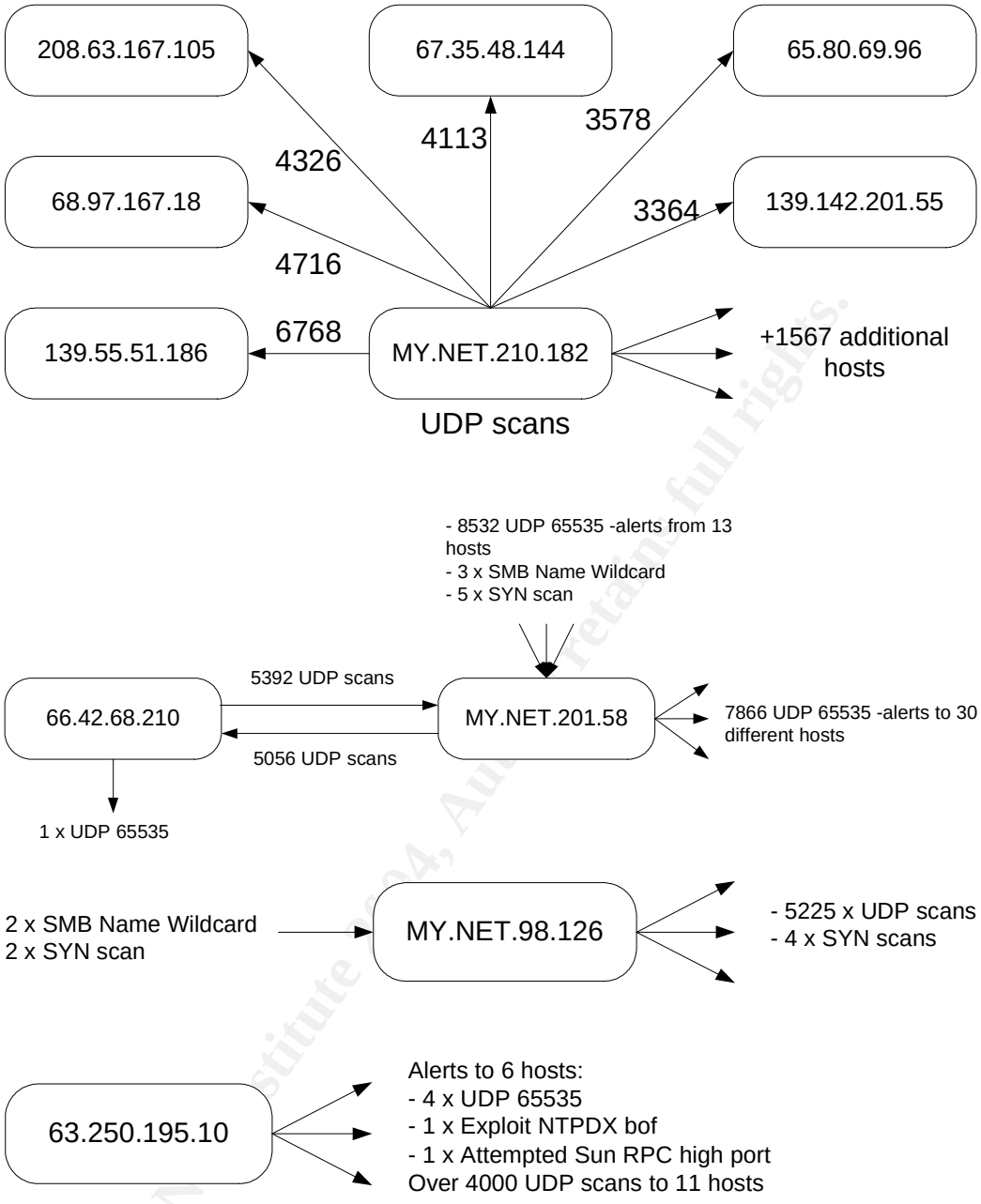




Graphs 6-10: UDP scans from internal hosts

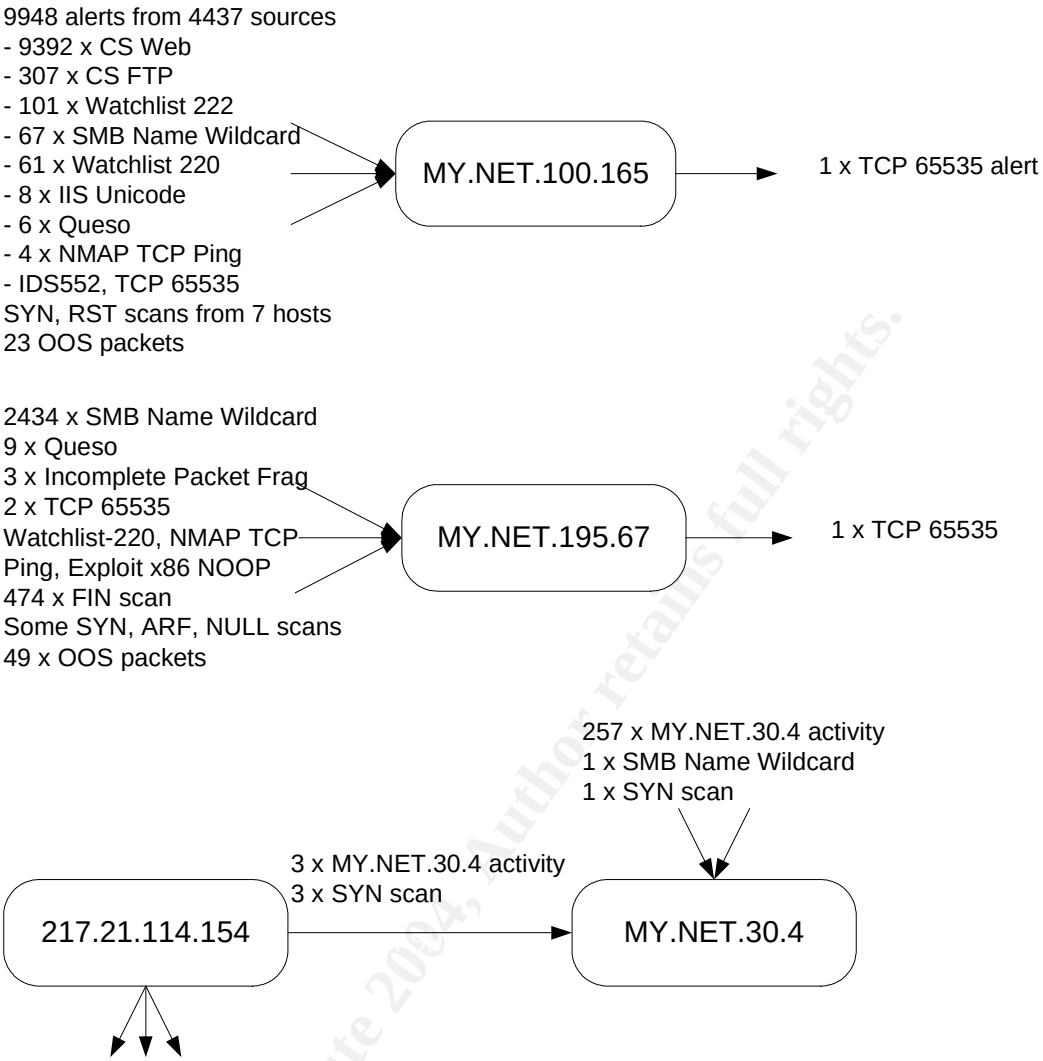
Here are 5 internal hosts and one external that perform heavy UDP scanning and/or raise UDP high port 65535 –alerts. The two internal hosts in the first graph (MY.NET.132.23 and MY.NET.178.19) seem to scan random UDP ports. The UDP traffic of the rest of the hosts in the graphs is much more narrower, including ports associated with peer-to-peer programs. I recommend to check out the internal hosts shown in the first graph and to warn the users of the other hosts about the dangers involved with P-P programs.





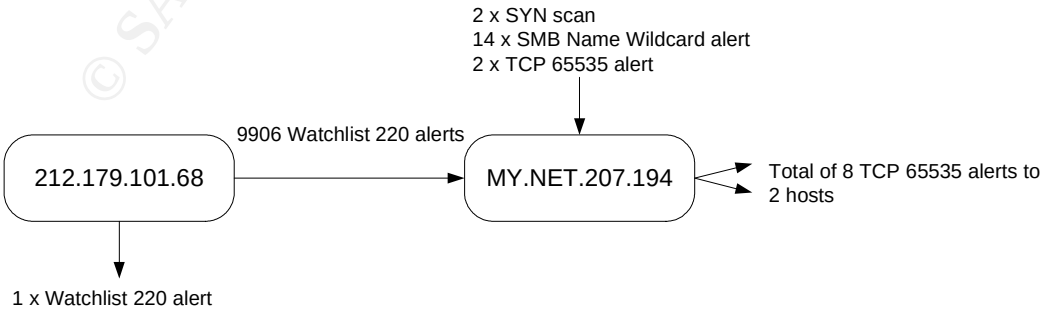
Graphs 11-13: Internal hosts that draw external fire

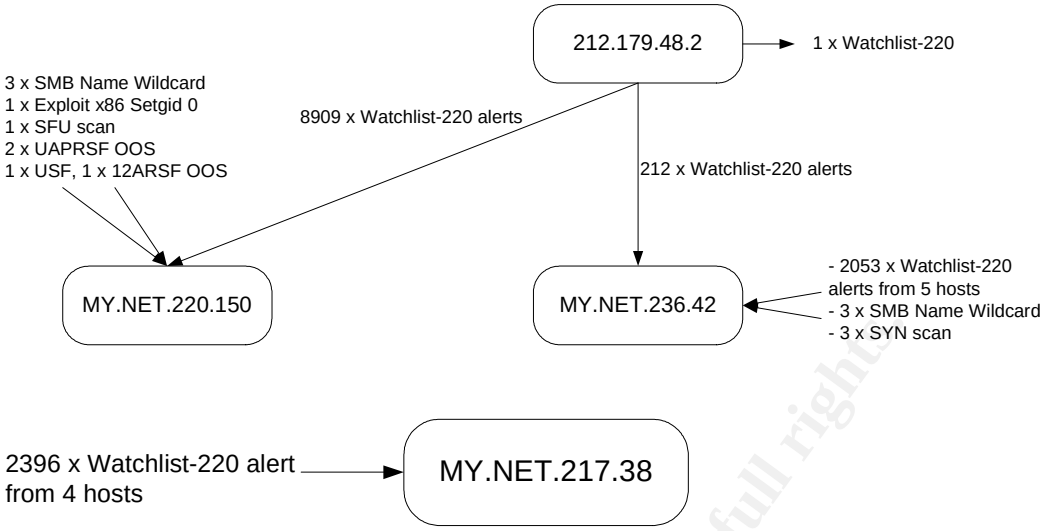
These three internal hosts got significantly more external attention than the average internal hosts. No evidence shows compromise, though. Still it is a good idea to keep an eye on these hosts. I recommend checking that their configuration and hardening is properly done. I also recommend checking the firewall settings to filter all unnecessary traffic out.



Graphs 14-16 Watchlist 000220 alerts

These hosts seemed to be heavily involved with the custom alert Watchlist 000220.





Selected Five External Hosts

The following five external hosts/networks were linked to malicious activities or other network events that are good to confirm with their owners:

IP address	Reason to contact	WHOIS information
212.179.101.68 & 212.179.48.2	These hosts triggered the most Watchlist 000220 -alerts	Netnum: 212.179.0.0 - 212.179.0.255 Netname: REDBACK-EQUIPMENT Mnt-by: INET-MGR Descr: BEZEQINT-EQUIPMENT Country: IL Admin-c: MR916-RIPE Tech-c: ZV140-RIPE Status: ASSIGNED PA Remarks: please send ABUSE complains to abuse@bezeqint.net Remarks: INFRA-AW Notify: hostmaster@bezeqint.net Changed: hostmaster@bezeqint.net 20021020 Source: RIPE
216.251.32.98	1745 packets were sent to this host – supposedly from Microsoft's address space	OrgName: InternetNamesForBusiness.com OrgID: INFB Address: 500 East Broward Boulevard, Suite 1700 City: Ft. Lauderdale StateProv: FL PostalCode: 33394 Country: US NetRange: 216.251.32.0 - 216.251.47.255 CIDR: 216.251.32.0/20 Updated: 2001-04-09 TechName: InternetNamesForBusiness.com TechPhone: +1-954-463-3080 TechEmail: admin@internetnamesforbusiness.com
66.42.68.210	This address was heavily involved with UDP 65535 traffic with MY.NET.201.58	OrgName: Pac-West Telecomm, INC. OrgID: PWTI Address: 1776 W. March Lane Address: Suite 250 City: Stockton StateProv: CA

		PostalCode: 95207 Country: US NetRange: 66.42.0.0 - 66.42.127.255 CIDR: 66.42.0.0/17 RegDate: 2000-11-10 Updated: 2002-11-15 TechHandle: ZP86-ARIN TechName: Administrator TechPhone: +1-800-722-9378 TechEmail: admin@mdsg-pacwest.com
218.68.216.47	A heavy SYN scanner	inetnum: 218.67.128.0 - 218.69.255.255 netname: CHINANET-TJ descr: CHINANET Tianjing province network descr: China Telecom descr: A12,Xin-Jie-Kou-Wai Street descr: Beijing 100088 country: CN admin-c: CH93-AP tech-c: HZ19-AP mnt-by: MAINT-CHINANET mnt-lower: MAINT-CHINANET-TJ changed: hostmaster@ns.chinanet.cn.net 20010820 status: ALLOCATED PORTABLE source: APNIC person: Chinanet Hostmaster address: No.31 ,jingrong street,beijing address: 100032 country: CN phone: +86-10-66027112 fax-no: +86-10-66027334 e-mail: hostmaster@ns.chinanet.cn.net e-mail: anti-spam@ns.chinanet.cn.net nic-hdl: CH93-AP mnt-by: MAINT-CHINANET changed: hostmaster@ns.chinanet.cn.net 20021016 source: APNIC
172.186.87.8	Another scanner	OrgName: America Online OrgID: AOL Address: 8619 Westwood Center Drive Address: Suite 200 City: Vienna StateProv: VA PostalCode: 22182 Country: US NetRange: 172.128.0.0 - 172.191.255.255 CIDR: 172.128.0.0/10 NetName: AOL-172BLK AbuseName: Abuse AbusePhone: +1-703-265-4670 AbuseEmail: abuse@aol.net

Analysis Process

I started the analysis, as recommended, by studying the other GCIA student's work. I found Les M. Gordon's and Tod A. Beardsley's practical as ones with high quality and professionalism in their analysis methods and results. There were others too, of course, but the list would grow too long to mention them all here.

As to the analysis process itself, I first concatenated all alarms, scans and OOS messages to single files (alarms-all, scans-all and oos-all). The OOS

logs consisted of multiple lines per log entry, so I run it through Emacs and several shell- and awk-scripts to suit my needs. I removed all corrupted entries and processed the log to contain a single event in a single line, different fields separated by commas. Similarly I formatted the alerts and scans too and got three new files: alert-formatted, scans-formatted and oos-formatted. These three files were the base for all my further analysis.

Next I used Les Gordon's perl script to produce the Alert Summary table I have in the beginning of this analysis. After studying the alerts I counted the top talkative players and decided to draw link graphs of them. The top-ten lists and the link graphs helped me to go through the analysis and to find the important relationships between the communicating hosts.

I selected the alerts I wanted to analyze and processed them one by one. I tried to find correlations from the other's work when possible. After all analysis was done I wrote the executive summary and listed the recommended actions to the beginning to the assignment.

References

[Bioware] BioWare Corp, “Neverwinter Nights Technical FAQ”, URL: http://nwn.bioware.com/support/techfaq.html, referred 25 April 2003

[Beardsley] Beardsley, Tod, “Intrusion Detection and Analysis: Theory, Techniques and Tools”, GCIA Practical Assignment, 8 May 2002

[Calhoun] Calhoun, Johnny, “Intrusion Detection: In-Depth Analysis”, GCIA Practical Assignment, 8 January 2003

[Dell] Dell, J. Anthony, “Adore Worm – Another Mutation”, URL: http://www.sans.org/rr/threats/mutation.php, 6 April 2001

[Dittrich] Dittrich, David et al, “The mstream distributed denial of service attack tool”, URL: http://staff.washington.edu/dittrich/misc/mstream.analysis.txt, referred 25 April 2003

[doshelp] Doshelp.com, “Trojan and Remote Access Service Ports”, URL: http://doshelp.com/trojanports.htm, referred 25 April 2003

[Gordon] Gordon, Les, “Intrusion Analysis – The Director’s Cut!”, GCIA Practical Assignment, 22 November 2002

[Gordon2] Gordon, Les, “RE: Attack or Virus with 255.255.255.255 Spoofed IP address”, URL: http://cert.uni-stuttgart.de/archive/intrusions/2003/02/msg00011.html 3 Feb 2003

[Hurwicz] Hurwicz, Michael, “Emerging Technology: Peer-To-Peer Networking Security”, URL: http://www.networkmagazine.com/shared/article/showArticle.jhtml?articleId=8703302&pgno=1, 6 February 2002

[mac_find] Coffer.com, “Vendor/Ethernet MAC Address Lookup”, URL: http://coffer.com/mac_find/ Referred 25 April 2003

[RFC791] Postel, J. Internet Protocol, Request for Comments 791 (also STD0005), URL: ftp://ftp.rfc-editor.org/in-notes/rfc791.txt, 1 September 1981

[RFC793] Postel, J. Transmission Control Protocol, Request for Comments 793 (also STD0007), URL: ftp://ftp.rfc-editor.org/in-notes/rfc793.txt, 1 September 1981

[snort-faq] Snort IDS FAQ, "I am getting too many IIS Unicode attack detected…" URL: http://www.snort.org/docs/faq.html#4.17, referred 25 April 2003

[Stevens] Stevens, Richard, TCP/IP Illustrated, Volume 1, Addison Wesley 1994

[Steward] Steward, Joe, "Re: [Snort-users] What defines a IIS Unicode Attack?" URL: http://marc.theaimsgroup.com/?l=snort-users&m=97665860614029&w=2, 12 December 2000

[Wisener] Wisener, Michael, "Intrusion Detection in Depth: Finding the Needle", GCIA Practical Assignment, 28 January 2003

Last Updated: July 21st, 2017

Upcoming Training

SANSFIRE 2017	Washington, DC	Jul 22, 2017 - Jul 29, 2017	Live Event
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 Virginia Beach 2017	Virginia Beach, VA	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS Adelaide 2017	Adelaide, Australia	Aug 21, 2017 - Aug 26, 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 Fall 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 Berlin 2017	Berlin, Germany	Oct 23, 2017 - Oct 28, 2017	Live Event
San Diego Fall 2017 - SEC503: Intrusion Detection In-Depth	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	vLive
SANS San Diego 2017	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Seattle 2017	Seattle, WA	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Paris November 2017	Paris, France	Nov 13, 2017 - Nov 18, 2017	Live Event
Community SANS Pensacola SEC503	Pensacola, FL	Nov 27, 2017 - Dec 02, 2017	Community SANS
SIEM & Tactical Analytics Summit & Training	Scottsdale, AZ	Nov 28, 2017 - Dec 05, 2017	Live Event
SANS Cyber Defense Initiative 2017	Washington, DC	Dec 12, 2017 - Dec 19, 2017	Live Event
SANS Security East 2018	New Orleans, LA	Jan 08, 2018 - Jan 13, 2018	Live Event
SANS Las Vegas 2018	Las Vegas, NV	Jan 28, 2018 - Feb 02, 2018	Live Event
SANS Dallas 2018	Dallas, TX	Feb 19, 2018 - Feb 24, 2018	Live Event
SANS OnDemand	Online	Anytime	Self Paced
SANS SelfStudy	Books & MP3s Only	Anytime	Self Paced