Global Information Assurance Certification Paper

GCIA Practical Assignment Version 3.3
For Intrusions Detection In-Depth Course (Track 3)
By Erik Montcalm
Submitted on 2003-11-12

Part 1: Describe the State of Intrusion Detection
An Intrusion Detection Challenge: Reviewing and Comparing IDS Systems

Executive Summary:

Intrusions Detection Systems are now part of the set of standard defensive tools used by most security professionals. But how do people make decisions about which one to buy and if it will prove to be effective in your specific environment? This shows that traditional IDS testing is not a good way of measuring value of the system, it mostly only measures performance and detection correctness. Also, it is the author of this paper's contention that this type of testing is necessary but not sufficient to make an educated decision on a product. Other testing factors will be suggested and the importance of live testing on the actual network where the IDS system would be installed will be discussed.

1.0 Introduction:

Intrusions detection systems have now matured to the point where they are not considered new technology anymore. Several decision makers would like to invest in this type of technology to insure an additional layer of security for their network. But with the total cost of ownership of an IDS infrastructure being fairly high, these decision makers usually want to buy the best of breed solution for their needs. What criteria can these decision makers use? With so many flavours of IDS (host based, network based, rules based, anomaly detectors, etc) being offered by multiple vendors, how is one supposed to determine which one to buy? Several groups have tried to come up with a solution to this problem and provide a framework to benchmark or test Intrusion Detection Systems. Are these tests and benchmark sufficient to understand what a specific IDS does or not do? Our contention is that a more in-depth review, separate from the testing and benchmarking process is necessary and desirable before any conclusion can be made.

2.0 Test Categories:

Before we can discuss the relevance of certain types of IDS testing methodologies, we need to define some categories.

Several of these categories were obtained from the NIST report titled “An Overview of Issues in Testing Intrusion Detection Systems” (1)
These categories have been modified (mostly generalized) in order to fit the scope of this paper. New categories were then added.

2.1 Ad-hoc Testing

Reviews that do not seem to follow any type of testing parameters or standards fall into the ad-hoc testing category. This might give a useful “first impression” but that is about the extent of their usefulness. These review are usually pretty short. An example of this type of review is the SC magazine review of the Lancope StealthWatch IDS (2)

2.2 Quantitative Testing:

Quantitative testing usually implies that some tests were run and that a specific value is obtained as a result. These results can then easily be compared between IDS systems. The white paper entitled Experiences Benchmarking IDS Systems (3) by Marcus Ranum is a good example of this type of approach. Another approach is to outsource IDS testing to a company like Network Analysis Services who will customize tests to fit the needs of your company(6)

2.2.1 IDS Performance Benchmarking:

IDS performance benchmarking is the act of testing the correctness and reliability of the IDS system under certain types of load. Usually the metric verified is the number of packets dropped at a certain throughput. A few years ago, several IDS systems had issues fully analyzing a gigabit stream.

2.2.2 IDS Correctness Testing:

IDS correctness testing verifies if the detection engine of the IDS can detect all type of attacks on monitored networks. This usually involves trying to evade the IDS using several techniques and verifying that the IDS still picks it up.

2.2.3 IDS False Positive Testing:

A well-known issue with Intrusion Detection Systems is that they generate a large number of alerts that do not actually denote a security issue. This “noise” is known as a false positive. Standard IDS testing methodology usually includes a component that counts the total number or ratio of these.

2.2.4 Problems with Quantitative Testing

These approaches can tell you what an IDS does wrong or badly, but passing all of these tests is a pre-requisite for correct functionality, not an indication that the system adds value.

For example, performance testing a car tells you that the car will NOT break at 200km/h on the highway (performance), will NOT turn left when you ask it to turn right (correctness) and will NOT tell you that you are out of gas when in fact you are not (False positive testing). All of these tests do not tell you if the car is the right one for you or if of the car adds value to the product compared to other manufacturers. Will this car make you life easier or more complicated? Will it enable your employee to be more productive or will it be a drain on resources?

2.3 Qualitative Testing

Qualitative testing means that the result of a test is not a measurable value, but some other result that is usually based on the judgement of the person administering the test.

2.3.1 Ease of Deployment & Administration

This is the ease (or difficulty) of installing and maintaining the IDS system being reviewed.

2.3.2 Feature list

This is the set of features that the IDS vendor claims to support in their product. In an ideal world, this would be easy to compare and measure. But unfortunately, not every IDS feature is implemented in the same way and not everybody agree on what defines a certain feature.

2.3.3 Problems with Qualitative Testing:

Qualitative IDS tests usually rely a lot on the person performing the tests. For example, if the reviewer thinks that the IDS is easy to remotely administer, then that goes in the review. This is the case even if another person might think the opposite.

3.0 Other categories we would like to see:

In order to improve the way IDS systems are tested, it is our contention that more qualitative tests are necessary. This makes testing less scientific and adds a certain responsibility on the shoulders of the people performing these tests. Unfortunately, we feel that the following categories are necessary even if they cannot be easily measured.

3.1 Integration With Process

Process is arguably the most important thing in the Incident Lifecycle. Having a well-established process allows Intrusion Analysts, Incident Handlers and various manager and administrators to perform all necessary steps every time an incident occurs. Sylvain Randier makes a really good case for this in his paper about the IDS process (4). It is therefore very important that the IDS product being evaluated integrates well with the established processes of an organization.
If a specific incident requires a special type of escalation, does the IDS console allow for the analyst to automate this?
Short of having this ability, the IDS system should at the very least not hinder the user from following the process.

3.2 Learning Curve for Analysts/Sysadmin

The IDS system might be the neatest things since sliced bread with a very long list of features. But if everybody that is to use the IDS system requires 1 month of training or seminars, then it is almost useless. The learning curve for users and administrator should be carefully evaluated as this might add significant cost to the Total Cost of Ownership of the IDS.

3.3 Tuning Abilities

Every IDS vendor offers some tuning ability to eliminate background noise and false positives. The real question becomes how easy and automated is the process.

3.4 Integration with other products

With Meta-IDS systems like ArcSight and Intellitactics offering to add an integration layer above traditional IDS and several vendors offering IDS plug-ins for specific things (policy,correlation,vulnerability scans etc), the ability to send (and sometimes receive) alerts from other products is increasingly important. The problem is that sometimes IDS vendors do not want their product to interoperate with other products because this vendor might want their installed base to buy the module from them.

4.0 *Problems preventing full IDS testing*

4.1 Real World

Developing a complete and adequate IDS testing methodology is nearly impossible for several reasons. The first reason being that complete IDS testing cannot happen in a lab according to top IDS experts(5). Several usage conditions only occur in the wild and can never be replicated. For example, our criteria of Process Integration, Learning Curve and Tuning ability are particularly hard to reproduce in a lab.

4.2 Costs

All of these factors are made worse by the fact that IDS deployments, maintenance and training are fairly expensive. So deploying multiple IDS infrastructures from different vendors until the correct product is found is usually not an option for most organizations.

4.3 Time

Another reason that correct and complete IDS testing is very hard is that to do a good job, one would have to run the IDS in a real-world situation for an extended period of time as several conditions only occur after prolonged usage (huge Databases, upgrade scenarios, new deployments, etc)

5.0 Recommendations

If all of this is true, how should we review Intrusion Detection Systems?
The solution is probably a mix of tests that are currently being done with a limited deployment that could adequately test all of the issues mentioned above.
For example, a corporation would need to deploy enough sensors, aggregation servers, plug-in modules and databases to make sure that all elements of the IDS are tested, but this deployment does not need to be enterprise wide to avoid excessive costs. As Marcus Ranum states in an interview (5)

"Before you buy any product, test it on your network as it would be deployed operationally. Lots of things that look good on paper are much less effective in the real world than you'd expect. The only way to see if you'll be comfortable with a system is to try it first hand!"

This will allow testing of a lot of "intangibles" like User Interface, learning curve, ratio of manual errors, speed of access to information and integration with Incident Handling Process. This should lead to buying an IDS that actually fits your needs.If vendors and buyers could agree on this way of doing things, then maybe the IDS industry can avoid its ill-fated destiny that the Gartner group predicts for it (7)

Works Cited

(1)Mell,Lippman.Haines and Zissman.An overview of Issue in Testing Intrusion Detection Systems. NIST, **June 2003**
http://csrc.nist.gov/publications/nistir/nistir-7007.pdf

(2) Jeff Bankster. StealthWatch Intrusion Detection Appliance.Scmagazine . SCMagazine.August 2001
http://www.scmagazine.com/scmagazine/2001_08/review/review1.html

(3)Marcus Ranum. Experiences Benchmarking Intrusion Detection Systems NFR. December 2001
http://www.snort.org/docs/Benchmarking-IDS-NFR.pdf

(4)Sylvain Randier, Process Issues (GCIA Practical White Paper).
SANS 2003
http://www.giac.org/practical/GCIA/Sylvain_Randier_GCIA.pdf

(5) Gene Schultz. Interview with top 3 IDS experts
Information Security Bulletin, May 2000
http://www.chi-publishing.com/portal/backissues/pdfs/ISB_2000/ISB0504/ISB0504GS.pdf

(6)<Example of IDS testing as Service Offering>
Network Analysis Systems Corporation Web. Service Offering Section
http://www.netansys.co.uk/services/ids.html

(7)Gartner Group.June11,2003
http://www3.gartner.com/5_about/press_releases/pr11june2003c.jsp

(8) General Reference, IDS reviews
http://www.neohapsis.com

Part 2 : Network Detects

Detect1: Welchia is still alive and VERY healthy

Welchia/Nachi Worm Pings:

Context:

After I started running snort at home, I quickly discovered that I was getting hit from the outside multiple times per second by ICMP requests that were generating the “ICMP PING CyberKit 2.2 Windows” snort Rule (SID 483). Some basic research showed that this was a fairly well documented case of Welchia/Nachia worm looking for targets. I found this interesting because Welchia has been around for at least 2 months now, but end users are still being hammered by this.

Here are 4 consecutive Alerts logged by Snort).

[**] [1:483:2] ICMP PING CyberKit 2.2 Windows [**]
[Classification: Misc activity] [Priority: 3]
11/02-08:59:42.234783 65.95.179.178 -> 65.94.176.XXX
ICMP TTL:121 TOS:0x0 ID:59987 IpLen:20 DgmLen:92
Type:8 Code:0 ID:768   Seq:50857 ECHO
[Xref => http://www.whitehats.com/info/IDS154]

[**] [1:483:2] ICMP PING CyberKit 2.2 Windows [**]
[Classification: Misc activity] [Priority: 3]
11/02-08:59:55.885556 65.95.182.52 -> 65.94.176.XXX
ICMP TTL:121 TOS:0x0 ID:29101 IpLen:20 DgmLen:92
Type:8 Code:0 ID:512   Seq:52905 ECHO
[Xref => http://www.whitehats.com/info/IDS154]

[**] [1:483:2] ICMP PING CyberKit 2.2 Windows [**]
[Classification: Misc activity] [Priority: 3]
11/02-09:00:11.517493 65.93.34.177 -> 65.94.176.XXX
ICMP TTL:119 TOS:0x0 ID:35940 IpLen:20 DgmLen:92
Type:8 Code:0 ID:768   Seq:55213 ECHO
[Xref => http://www.whitehats.com/info/IDS154]

[**] [1:483:2] ICMP PING CyberKit 2.2 Windows [**]
[Classification: Misc activity] [Priority: 3]
11/02-09:00:14.677723 65.94.134.214 -> 65.94.176.XXX
ICMP TTL:125 TOS:0x0 ID:28549 IpLen:20 DgmLen:92

Type:8 Code:0 ID:1024 Seq:929 ECHO
[Xref => http://www.whitehats.com/info/IDS154]

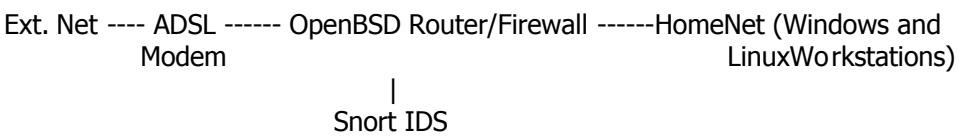
Associated packets:

08:59:42.234783 0:5c:ea:53:0:0 0:5e:0:21:45:0 7901 114:
7065 415f b3b2 415e b078 0800 d900 0300
c6a9 aaaa aaaa aaaa aaaa aaaa aaaa aaaa
aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa
aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa
aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa
aaaa 7c11 df93 594e 4f54 0000 6b45 b90d
a53f fafd
08:59:55.885556 0:5c:71:ad:0:0 0:5e:0:21:45:0 7901 114:
e689 415f b634 415e b078 0800 d200 0200
cea9 aaaa aaaa aaaa aaaa aaaa aaaa aaaa
aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa
aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa
aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa
aaaa 0000 df93 594e 4f54 0000 6b45 b90d
a53f fafd
09:00:11.517493 0:5c:8c:64:0:0 0:5e:0:21:45:0 7701 114:
6158 415d 22b1 415e b078 0800 c7fc 0300
d7ad aaaa aaaa aaaa aaaa aaaa aaaa aaaa
aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa
aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa
aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa
aaaa c0af 3001 415e b078 0035 dac2 b7a5
c931 3a03
09:00:14.677723 0:5c:6f:85:0:0 0:5e:0:21:45:0 7d01 114:
1411 415e 86d6 415e b078 0800 9b09 0400
03a1 aaaa aaaa aaaa aaaa aaaa aaaa aaaa
aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa
aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa
aaaa a aaa aaaa aaaa aaaa aaaa aaaa aaaa
aaaa c0af 3001 415e b078 0035 dac2 b7a5
c931 3a03

1-Source of Trace:

These logs were found on the external OpenBSD system that acts as the primary firewall/router for my home DSL connection (1.5 Mb, PPPoE). These detects were found on the external interface of this system.

Simplified Network Layout:



2- Detect was generated by:

The Detect was generated by the SNORT lightweight Intrusion Detection System version 1.8.6 (I know, I should upgrade to a more recent version)

3- Probability the source address was spoofed

If this activity is indeed Welchia, then a lot of the activity would need to be from real source IPs in order for this ICMP reconnaissance to be useful to the worm. The goal of this reconnaissance is to find new targets, it needs to see the reply in order to build the target list.

But several reports about this worm (including the initial reports on www.incidents.org) mention that some of the traffic is spoofed.

I can only guess at the worm writer's intention, but the worm probably tries to spoof some source IPs in order to overload the senses of anybody looking at various logs (so that the real IPs do not immediately stand out).

Most of the Source IPs in my snort logs are in the range 65.92.X.X to 65.95.X.X which makes them inside my service Providers network (Bell Nexxia is the networking arm of Bell Sympatico who supplies me with DSL service.

ARIN WHOIS lookup for the range in question:

Search results for: 65.93.34.177

```
Bell Canada BELLNEXXIA-10 (NET-65-92-0-0-1)
                                  65.92.0.0 - 65.95.255.255
Bell Nexxia (High Speed) HSLON-CA (NET-65-93-0-0-1)
                                  65.93.0.0 - 65.93.63.255

# ARIN WHOIS database, last updated 2003-11-01 19:15
```

```
# Enter ? for additional hints on searching ARIN's WHOIS
database.
```

Some other source adresses are also present, be they also seem to be part of the Sympatico network (other ranges, like 67.69.211.X)

4- Description of the Attack

Welchia will try and find potential victims to infect by sending out ICMP echo requests to IP addresses that it constructs using a semi-random algorithm. This is the part that is seen in my logs.

Once the worm finds hosts to infect, it then sends the exploit code to obtain access to the Windows System. It then proceeds to infect the system (See attack mechanism section).

Here is the CVE entry for the exploit that the Nachi/Welchia worm uses.

Name	CAN-2003-0352 (under review)
Description	Buffer overflow in a certain DCOM interface for RPC in Microsoft Windows NT 4.0, 2000, XP, and Server 2003 allows remote attackers to execute arbitrary code via a malformed message, as exploited by the Blaster/MSblast/LovSAN and Nachi/Welchia worms.

5- Attack Mechanism

There are several steps to a successful Welchia infection, only the first one is represented in my logs. The other steps are listed here for completeness.

The Mcafee (2) web site has an excellent paper specifically about the MS-RPC worms (Blaster, Welchia and variants) and how to find them in our logs

Discovery:

According to both the Symantec and McAfee web sites, the discovery algorithm works as follows (3)

1. Selects the victim IP address in two different ways: The worm uses either A.B.0.0 from the infected machine's IP of A.B.C.D and counts up, or it will construct a random IP address based on some hard-coded addresses.

   After selecting the start address, the worm counts up through a range of Class B-sized networks; for example, if the worm starts at A.B.0.0, it will count up to at least A.B.255.255.

2. Sends an ICMP echo request, or PING, to check whether the constructed IP address is an active machine on the network.

3. Once the worm identifies a machine as being active on the network, it will either send data to TCP port 135, which exploits the DCOM RPC vulnerability, or it will send data to TCP port 80 to exploit the WebDav vulnerability.

Exploit:

Infection

Syamntec mentions that to find infected hosts, you need to "Look for a ping, then traffic on port 135/tcp, 666-to-765/tcp, and then 69/udp, like this"

So after choosing which IPs to ping, Welchia will go-ahead and ping that list of IPs. It will then try to send the exploit code to port 135/tcp (RPC). If successful, one should see some other ports involved in traffic like 69/udp (tftp, to transfer the worm) and several other ports that can be used as a rootshell.

My logs do not show any of this as my firewall is not configured to reply to anything at all. It just appears like an unused IP to Welchia when it does its initial ICMP scanning and hence does not get any of the exploit code or other activity.

Summary:

In my specific case, the ICMP packets appear to be the stimulus that goes without a response.

ICMP is being used for reconnaissance, but the service that the worm wants to attack is RPC-DCOM (not shown in the logs because the attack never gets to that stage)

The service has some VERY well known vulnerabilities and exposures and some VERY well documented worms that automatically exploit these vulnerabilities.

6- Correlations

I could find a lot of correlations for this type of activity, but none of the specific sources I found were reported to security correlation sites like Dshield or MyNetWatchman.

But several mailing mailing lists and other sources confirm my interpretation of the logs (Symantec, McAfee, various Snort mailing lists)

The most obvious correlation is on the official Symantec write-up about these RPC worms referenced above:

The Symantec example ping-request detects looks exactly like mine:

```
11:47:47.576542 169.254.56.166 > 169.254.189.84: icmp: echo request
0x0000 4500 005c 599d 0000 8001 970c a9fe 38a6 E..\Y.........8.
0x0010 a9fe bd54 0800 fa51 0200 a658 aaaa aaaa ...T...Q...X....
0x0020 aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa ................
0x0030 aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa ................
0x0040 aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa ................
0x0050 aaaa aaaa aaaa aaaa aaaa aaaa ............
```

The Michael Filtcraft reply to questions about his practical detects posted to the intrusions@incidents.org on 2003-11-05 also talks about Welchia and its pings:

"Yes, there was other traffic from the source IP. The attacking IP ping'd the destination IP, to which the dst ip replied. This ICMP request caused Snort to fire a "ICMP PING CyberKit 2.2 Windows" alert. A strong indicator the source is infected with the Welchia worm.

Further analysis of the ICMP request show a ping payload of 64 bytes of "0xaa" data. A typical signature of a Welchia ping request."

7- Evidence of Active Targeting

There is no evidence that my network was specifically targeted by this type of activity. The fact that most of these source are inside of Bell Nexxia's network is probably due to a combination of factors.

A) As seen in the Attack Mechanism Section, this worm does a lot of its reconnaissance by scanning its own Class B subnet, so it makes a lot of sense that a lot of the sources would be from my class B.
B) The fact that I am not seeing many attacks from non- Bell Nexxia attackers is probably an indication that some type of filtering for this is happening at Bell Nexxia Perimeter (filtering ICMP with 'aa''s in the packet is not hard). Several reports from various ISPs in the states

mention that this is also occurring (only seeing source IPs from inside of their ISP.

8- Severity

Criticality = 5

System is a single point of failure for my home LAN and is therefore a very critical box.

Lethality = 2

Not really an attack, can be considered reconnaissance. This reconnaissance can lead to a Denial of Service condition, but this is not likely anymore.

System Countermeasures = 5

All windows systems on my home network automatically download and install critical patches from the WindowsUpdate site. In any case, the pings themselves or any other packets would be stopped at the router/firewall as I do not have any open ports. (I disabled SSH after the last OpenSSH vulnerability).

Network Countermeasures = 1

My firewall drops ICMP. This insures that I am not receiving any exploit packets, but this does not help if the flood of incoming ICMP create a DOS condition on my DSL connection. This wasted bandwidth is somewhat low at the moment, but we can imagine what it could have been like during the initial outbreak.

Severity = (5+2) –(5+1) = 1

*Note, my evalution for this severity would probably have been a lot higher during the initial outbreak when:

a) A lot more systems were probably sending out the pings
b) The patches were not as widely deployed
c) The network effects were not quite as well understood

9-Defensive recommendations

Patch all Windows with MS03-26 and MS03-07 (not to mention ALL critical patches). This will make sure that no matter what else occurs, the windows machines are not vulnerable to the actual worm.

As far as the DOS condition, you can use a filtering router with an ACL to drop ICMP at the router, but this might not be a good idea in all situations. The ideal solution is to drop ICMP where a series of "AA" is present, but not all devices can do this.

Larger customers with a good relationship with their ISPs might get some help with filtering this traffic before it reaches their network. (ACLs, etc)

10-Multiple Choice Test Question

How does the Welchia/Nachi worm try and choose which IPs to ping?

a) It tries every sequential IP in its Class A network
b) It tries random IPs on the internet until it finds a host
c) It will construct a start IP based on the infected machines IP and then try the whole class B network
d) It will choose a starting a random IP address based on some hard-coded list and try the whole class B network.
e) Could be C or D
f) None of the above
g) All of the above

Answer E

References:

1-
http://archives.neohapsis.com/archives/snort/2003-08/0643.html2-

Detecting Traffic due to RPC Worms. Symantec Corportation
http://securityresponse.symantec.com/avcenter/venc/data/detecting.traffic.due.to
pc.worms.html

Detect2: Gamespy Arcade Custom UDP pings

Something was being very active on the network and creating some lag when I thought no heavy load was occurring. The Switch and HUB lights were going insane, so snort was run to find ALL traffic (using the –v –d switches with no configuration file specified. TCPDUMP could have just as easily been used.

Several thousand of these logs were collected over a 20-minute span.
The only thing that differs is the sources. Several hundred different sources were involved, but the activity stayed the same.

11/02-15:44:07.861120 65.94.176.XXX:57116 -> 12.209.92.35:13139
UDP TTL:127 TOS:0x0 ID:26538 IpLen:20 DgmLen:60
Len: 40
91 01 00 01 04 86 00 00 00 00 00 00 00 00 00 00 ....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....

11/02-15:44:08.220628 65.94.176.XXX:55646 -> 141.150.14.193:13139
UDP TTL:127 TOS:0x0 ID:26540 IpLen:20 DgmLen:60
Len: 40
91 01 00 01 04 87 00 00 00 00 00 00 00 00 00 00 ....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....

11/02-15:44:08.238689 141.150.14.193:13139 -> 65.94.176.XXX:13139
UDP TTL:110 TOS:0x0 ID:33722 IpLen:20 DgmLen:60
Len: 40
91 01 00 01 52 6D 00 00 00 00 00 00 00 00 00 00 ....Rm..... .....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....

11/02-15:44:08.267764 141.150.14.193:13139 -> 65.94.176.XXX:55646
UDP TTL:110 TOS:0x0 ID:33724 IpLen:20 DgmLen:60
Len: 40
91 01 00 02 04 87 52 6E 00 00 00 00 00 00 00 00 ......Rn..... .....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....

11/02-15:44:08.282979 65.94.176.XXX:55646 -> 141.150.14.193:13139
UDP TTL:127 TOS:0x0 ID:26541 IpLen:20 DgmLen:60
Len: 40
91 01 00 03 00 00 52 6E 00 00 00 00 00 00 00 00 ......Rn..... .....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....

11/02-15:44:08.633812 12.209.92.35:13139 -> 65.94.176.XXX:57116
UDP TTL:49 TOS:0x0 ID:22957 IpLen:20 DgmLen:60
Len: 40
91 01 00 02 04 86 04 1D 00 00 00 00 00 00 00 00 ....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....

11/02-15:44:08.642533 65.94.176.XXX:57116 -> 12.209.92.35:13139
UDP TTL:127 TOS:0x0 ID:26542 IpLen:20 DgmLen:60
Len: 40
91 01 00 03 00 00 04 1D 00 00 00 00 00 00 00 00 ....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....

11/02-15:44:08.808061 208.157.148.72:13139 -> 65.94.176.XXX:13139
UDP TTL:52 TOS:0x0 ID:30773 IpLen:20 DgmLen:60
Len: 40
91 01 00 01 01 4A 00 00 00 00 00 00 00 00 00 00 .....J..........
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....

11/02-15:44:09.533437 65.94.176.XXX:57098 -> 24.207.232.62:13139
UDP TTL:127 TOS:0x0 ID:26546 IpLen:20 DgmLen:60
Len: 40
91 01 00 01 04 8A 00 00 00 00 00 00 00 00 00 00 ....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....

11/02-15:44:09.560528 24.207.232.62:13139 -> 65.94.176.XXX:13139
UDP TTL:110 TOS:0x0 ID:5960 IpLen:20 DgmLen:60
Len: 40
91 01 00 01 DF 45 00 00 00 00 00 00 00 00 00 00 .....E..........
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....

11/02-15:44:09.578489 68.211.176.118:13139 -> 65.94.176.XXX:13139
UDP TTL:109 TOS:0x0 ID:5340 IpLen:20 DgmLen:60
Len: 40
91 01 00 01 3B E9 00 00 00 00 00 00 00 00 00 00 ....;...........
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....

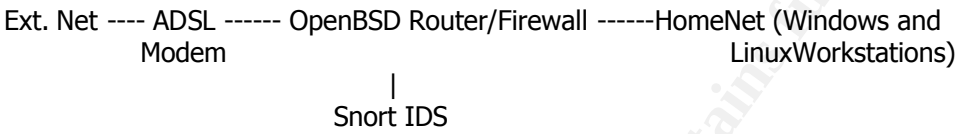
11/02-15:44:09.582785 68.211.176.118:13139 -> 65.94.176.XXX:50560
UDP TTL:109 TOS:0x0 ID:5341 IpLen:20 DgmLen:60
Len: 40

91 01 00 02 04 89 3B EA 00 00 00 00 00 00 00 00 ......;.....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....

1-Source of Trace

The trace was found on my home network, on the external interface of my OpenBSD firewall/router.

Simplified Network Layout:



2- Detect was generated by

Snort in packet capturing mode (option –vd, but without a configuration or rule file) was used to generate these logs. Once the fact that suspicious activity was taking place using the UDP protocol, a BPF filter was used to only capture that kind of activity.

3- Probability the source address was spoofed

The sources address could easily have been spoofed since the interesting detects use the UDP (connectionless) protocol. But given that these detects are most probably a false positive in the Intrusion Detection sense (see below), we can assume that the source addresses are real. Also, the activity seem to have a goal of relaying information and details about gaming sessions(Chatting, GameStats, etc) so spoofing the IPs or blocking them will probably remove some functionality.

4- Description of the Attack

There are no known exploits for this attack, since it is probably a false positive. Several packets are sent to and from the GameSpy Arcade servers. This was easy to track down. The more interesting detects happen to and from at least 20 different source on the internet, mostly high speed home connections (DSL or Cable). These UDP Packets with a destination port of 13139 were seriously

worrying me until some research showed that they are used for the Gamespy Arcade Chat rooms. These rooms are used as a starting point to find friends for on-line games. Turning that functionality off immediately stopped these UDP packets from coming in (although the GameSpy servers are still contacted once in a while with TCP packets to other ports, see below)

5- Attack Mechanism

As soon as you join the chatroom in Gamespy arcade, it looks like everybody in that chatroom starts exchanging UDP packets. Those packets always have a destination port of 13139 and usually have a sourceport of 13139 also.
Once in a while, a packet arrive from a different source port (like 57098).
The cause of this could not be established.

Leaving the chatroom in Gamespy causes this behaviour to immediately Stop. Some light TCP and UDP activity continues to occur, but this is usually on between the Gamespy servers and the user's IP. The traffic is a lot less pronounced and the reduced number of source makes it a lot less worrisome.

6- Correlations

Attacks sources could not be correlated against Internet attack lists like MyNetwatchman or DShield. But several other sources confirm that this is indeed normal behaviour for Gamespy (lists, etc) (1)

<Snipped>

Working Around the Firewall / Proxy
If you are behind a firewall/proxy and are able to change its settings, Arcade needs the following TCP ports open in
order to function:
6667 (IRC)
3783 (Voice Chat Port)
27900 (Master Server UDP Heartbeat)
28900 (Master Server List Request)
29900 (GP Connection Manager)
29901 (GP Search Manager)
13139 (Custom UDP Pings)
6515 (Dplay UDP)

6500 (Query Port)

<Snipped>

7- Evidence of Active Targeting

The targeting seems directed at anybody in a given chat room at a specific point in time. This is what I found in the GameSpy Documentation.
I have no way of actually verifying this for myself, because I would need prior knowledge of everybody's IP in a specific chatroom.

8- Severity

Criticality: 2

The machines being hit by these packets are all systems that are designed to be general purpose workstations or gaming machines. All are useful, but no single one is overly critical.

Lethality: 1

This attack (or false positive) had the effect of making me nervous (I actually though I had been hacked. But this was the only ill effect from these detects.

System Countermeasures: 5

All of my workstations are protected by the free version of ZoneAlarm. They should be configured to drop packets when not part of a known connection. I had authorized Gamespy Arcade as an application that could access the internet, which is why the packets arrived at destination.

Network Countermeasures: 5

My OpenBSD firewall is hardened and configured to drop all sessions not initiated from the inside of my network. If this had been an outside attack, it would have not gotten through.

Severity = (1+6) – (5+5) = -3

*** Note: The fact that this turned out to be a probable false positive does not lessen the learning experience I went through in order to find this out.
It gave me a good grasp of the Incident Handling process, including all of the emotional responses that the unknown can bring. ****

9-Defensive recommendations

I do not recommend any defense against this as it might break the desired behaviour of the application. If you do not like this behaviour, do not use GameSpy Arcade. Many bugs exist in Gamespy and they do not seem to want to fix them (as seen by this mail).

http://www.securityfocus.com/archive/1/344214/2003-11-09/2003-11-15/0

10-Multiple Choice Test Question

How many ports does GameSpy Arcade use for its various features:

a) 3
b) 4
c) 5
d) 6
e) 7
f) none of the above

Answer: F

Gamespy Arcade uses up to 9 different TCP or UDP ports depending on how many features are used.

References:

Gamespy Support Page. Official Web Site
1-http://www.gamespyarcade.com/support/firewalls.shtml

Detect3: Proxy Scan

Note: This detect has been submitted before. I was made aware of this after the fact. I reviewed questions that were asked on the list and tried to answer most o the here (this is where I decided to use the p0f tool, to make sure I had substantial additional contribution to these detects)

The following detects were dated 2003-10-30,

But the downloadable filenames where they were found are called 2003.09.30 (other practical students have already pointed this out)

To find an interesting trace, I put the snort alerts from the files 2002.9.28 to 2002.9.30 into a PostgreSQL database and grouped the Alerts by Signature name and count. 1 IP came back as the top offender for generating 2 different proxy scan alerts and several IPs from the same Class B network immediately followed were also doing smaller scans.

I need to scope the exercise a little bit so only the detect from 2003-10-30 will be discussed.

Output from the DB format of Snort-DB

All proxy scans for 2002-10-30 (aggregate view)

Count	sig_name	Source IP	DestPort(TCP)
5425	SCAN Proxy (8080) attempt	24.90.122.137	8080
5425	SCAN Squid Proxy attempt	24.90.122.137	3128
51	SCAN SOCKS Proxy attempt	216.77.219.225	1080
14	SCAN SOCKS Proxy attempt	216.77.216.104	1080
6	SCAN SOCKS Proxy attempt	216.77.216.150	1080
6	SCAN Squid Proxy attempt	172.184.170.160	3128

Sample of the individual alerts generated by SNORT and logged to the DataBase (sample for each source IP).

Time	Signature	Source	Target	
10/30/2002 9:33	SCAN Proxy (8080) attempt	24.90.122.137	207.166.45.235	8080
10/30/2002 9:33	SCAN Proxy (8080) attempt	24.90.122.137	207.166.45.237	8080
10/30/2002 9:33	SCAN Squid Proxy attempt	24.90.122.137	207.166.45.236	3128
10/30/2002 9:33	SCAN Squid Proxy attempt	24.90.122.137	207.166.45.238	3128
10/30/2002 9:33	SCAN Proxy (8080) attempt	24.90.122.137	207.166.45.239	8080
10/30/2002 9:33	SCAN Squid Proxy attempt	24.90.122.137	207.166.45.233	3128
10/30/2002 9:33	SCAN Proxy (8080) attempt	24.90.122.137	207.166.45.234	8080
10/30/2002 9:33	SCAN Proxy (8080) attempt	24.90.122.137	207.166.45.240	8080

10/30/2002 9:33	SCAN Proxy (8080) attempt	24.90.122.137	207.166.45.245	8080
10/30/2002 9:33	SCAN Squid Proxy attempt	24.90.122.137	207.166.45.244	3128
10/30/2002 9:33	SCAN Squid Proxy attempt	24.90.122.137	207.166.45.243	3128
10/30/2002 9:33	SCAN Proxy (8080) attempt	24.90.122.137	207.166.45.246	8080
10/30/2002 13:21	SCAN Proxy (8080) attempt	65.169.47.29	207.166.116.126	8080
10/30/2002 13:22	SCAN Proxy (8080) attempt	65.169.47.29	207.166.116.126	8080
10/30/2002 13:22	SCAN Proxy (8080) attempt	65.169.47.29	207.166.116.126	8080
10/30/2002 13:22	SCAN Squid Proxy attempt	65.169.47.29	207.166.116.126	3128
10/30/2002 13:22	SCAN Squid Proxy attempt	65.169.47.29	207.166.116.126	3128
10/30/2002 13:22	SCAN Squid Proxy attempt	65.169.47.29	207.166.116.126	3128
10/30/2002 9:17	SCAN Squid Proxy attempt	172.184.170.160	207.166.49.39	3128
10/30/2002 9:17	SCAN Squid Proxy attempt	172.184.170.160	207.166.49.39	3128
10/30/2002 9:17	SCAN Squid Proxy attempt	172.184.170.160	207.166.49.39	3128
10/30/2002 11:45	SCAN Squid Proxy attempt	172.184.170.160	207.166.50.39	3128
10/30/2002 11:45	SCAN Squid Proxy attempt	172.184.170.160	207.166.50.39	3128
10/30/2002 11:46	SCAN Squid Proxy attempt	172.184.170.160	207.166.50.39	3128
10/30/2002 12:46	SCAN Squid Proxy attempt	172.184.170.160	207.166.51.39	3128
10/30/2002 12:46	SCAN Squid Proxy attempt	172.184.170.160	207.166.51.39	3128
10/30/2002 12:46	SCAN Squid Proxy attempt	172.184.170.160	207.166.51.39	3128
10/30/2002 23:00	SCAN Squid Proxy attempt	212.32.4.25	207.166.151.27	3128
10/30/2002 23:00	SCAN Proxy (8080) attempt	212.32.4.25	207.166.151.27	8080
10/30/2002 23:00	SCAN SOCKS Proxy attempt	212.32.4.25	207.166.151.27	1080
10/30/2002 23:00	SCAN SOCKS Proxy attempt	212.32.4.25	207.166.151.27	1080
10/30/2002 23:00	SCAN Squid Proxy attempt	212.32.4.25	207.166.151.27	3128
10/30/2002 23:00	SCAN Proxy (8080) attempt	212.32.4.25	207.166.151.27	8080
10/30/2002 23:00	SCAN Squid Proxy attempt	212.32.4.25	207.166.151.27	3128
10/30/2002 23:00	SCAN Proxy (8080) attempt	212.32.4.25	207.166.151.27	8080
10/30/2002 23:00	SCAN SOCKS Proxy attempt	212.32.4.25	207.166.151.27	1080
10/30/2002 23:00	SCAN SOCKS Proxy attempt	212.32.4.25	207.166.151.27	1080
10/30/2002 23:00	SCAN Squid Proxy attempt	212.32.4.25	207.166.151.27	3128
10/30/2002 23:00	SCAN Proxy (8080) attempt	212.32.4.25	207.166.151.27	8080
10/30/2002 23:00	SCAN Squid Proxy attempt	212.32.4.25	207.166.151.27	3128
10/30/2002 23:00	SCAN Proxy (8080) attempt	212.32.4.25	207.166.151.27	8080
10/30/2002 23:00	SCAN SOCKS Proxy attempt	212.32.4.25	207.166.151.27	1080
10/30/2002 23:00	SCAN SOCKS Proxy attempt	212.32.4.25	207.166.151.27	1080
10/30/2002 23:00	SCAN Squid Proxy attempt	212.32.4.25	207.166.151.27	3128
10/30/2002 23:00	SCAN Proxy (8080) attempt	212.32.4.25	207.166.151.27	8080
10/30/2002 6:32	SCAN SOCKS Proxy attempt	216.77.216.104	207.166.225.208	1080
10/30/2002 6:36	SCAN SOCKS Proxy attempt	216.77.216.104	207.166.17.220	1080
10/30/2002 6:41	SCAN SOCKS Proxy attempt	216.77.216.104	207.166.108.129	1080
10/30/2002 6:45	SCAN SOCKS Proxy attempt	216.77.216.104	207.166.82.111	1080
10/30/2002 6:49	SCAN SOCKS Proxy	216.77.216.104	207.166.165.57	1080

	attempt			
10/30/2002 6:54	SCAN SOCKS Proxy attempt	216.77.216.104	207.166.95.144	1080
10/30/2002 6:58	SCAN SOCKS Proxy attempt	216.77.216.104	207.166.44.119	1080
10/30/2002 7:02	SCAN SOCKS Proxy attempt	216.77.216.104	207.166.71.193	1080
10/30/2002 7:07	SCAN SOCKS Proxy attempt	216.77.216.104	207.166.138.253	1080
10/30/2002 7:11	SCAN SOCKS Proxy attempt	216.77.216.104	207.166.243.38	1080
10/30/2002 7:15	SCAN SOCKS Proxy attempt	216.77.216.104	207.166.7.39	1080
10/30/2002 7:20	SCAN SOCKS Proxy attempt	216.77.216.104	207.166.32.3	1080
10/30/2002 7:24	SCAN SOCKS Proxy attempt	216.77.216.104	207.166.123.38	1080
10/30/2002 7:28	SCAN SOCKS Proxy attempt	216.77.216.104	207.166.246.195	1080
10/30/2002 11:36	SCAN SOCKS Proxy attempt	216.77.216.150	207.166.65.142	1080
10/30/2002 11:40	SCAN SOCKS Proxy attempt	216.77.216.150	207.166.175.163	1080
10/30/2002 11:44	SCAN SOCKS Proxy attempt	216.77.216.150	207.166.238.24	1080
10/30/2002 11:49	SCAN SOCKS Proxy attempt	216.77.216.150	207.166.216.231	1080
10/30/2002 11:53	SCAN SOCKS Proxy attempt	216.77.216.150	207.166.205.192	1080
10/30/2002 11:57	SCAN SOCKS Proxy attempt	216.77.216.150	207.166.199.98	1080
10/30/2002 12:02	SCAN SOCKS Proxy attempt	216.77.216.150	207.166.43.243	1080
10/30/2002 12:06	SCAN SOCKS Proxy attempt	216.77.216.150	207.166.239.247	1080
10/30/2002 12:10	SCAN SOCKS Proxy attempt	216.77.216.150	207.166.85.25	1080
10/30/2002 12:15	SCAN SOCKS Proxy attempt	216.77.216.150	207.166.36.164	1080
10/30/2002 12:19	SCAN SOCKS Proxy attempt	216.77.216.150	207.166.200.6	1080
10/30/2002 12:23	SCAN SOCKS Proxy attempt	216.77.216.150	207.166.24.213	1080
10/30/2002 12:28	SCAN SOCKS Proxy attempt	216.77.216.150	207.166.21.196	1080

1- Source of Trace

This trace was generated by the raw log files from incidents.org.
The dates used were from 2002-10-28 to 2002-10-31 2002-10-30 was chosen at first and the others were mostly used for correlation.

2- Detect was generated by

The alerts were generated by running Snort 2.0.2 with the following command

Snort –c /etc/snort.conf –b –r 2002.9.30 –k none

The snort.conf file contained the default rules and the default configurations except that output to a postgreSQL Database was added to make data manipulation easier.

The –k none option is used so that snort does try and calculate the checksum. Snort would not generate any alerts without this configuration. Several mailing list posts suggested trying this and it worked. I am assuming this would not have been necessary if I had been using the same snort version as was used to capture the logs.

This command was repeated for the raw logs of 2002-09-29, 2002-10-30, 2002- and 10-31.

The following snort rules were identified as being part of the 2 relevant scans

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

alert tcp $EXTERNAL_NET any -> $HOME_NET 1080 (msg:"SCAN SOCKS Proxy
attempt"; flags:S; reference:url,help.undernet.org/proxyscan/;

classtype:attempted-recon; sid:615; rev:3;)
alert tcp $EXTERNAL_NET any -> $HOME_NET 8080 (msg:"SCAN Proxy \(8080\)
attempt"; flags:S; classtype:attempted-recon; sid:620; rev:2;)
```

All of these alerts work in the same way. A packet will trigger one of these alerts if it meets all of these criteria:

- It uses the TCP protocol
- It arrives from the defined external network and is destined to the internal network
-It has the SYN Flag set
-It is destined to one of the following ports: 3128, 1080, 8080

3- Probability the source address(es) spoofed

It is not very likely that the source address was spoofed.

This scan uses the connection-oriented TCP protocol and the attacker was probably looking to get results of the scans. Although spoofing is not very likely in this case, see section 4.1 for the fact that it is possible.

4- Description of the Attack

The attackers are scanning for ports 1080, 8080 and 3128 which usually means looking for an open or vulnerable proxy. Several Trojans use some of these ports, it might be hard to distinguish between a scan for a proxy running on a given port of somebody trolling for trojaned computers.

I also looked at the possibilities that this was part of a RingZero attack, but the timeframe and characteristics of the packets do not make this likely. (see section below, the attacking computers do not seem to all be running the same OS)

4.1 Fingerprinting using p0f (obtaining more information about attackers)

I used a tool called p0f to passively fingerprint the captured packets, and all
Of the sources in the 216.77 class B network share the following trace by p0f

UNKNOWN [1024:49:0:40:.:QA:?:?]

So the Window Size, TTL and MSS are exactly the same.
But it could not give an actual identification of the OS

To me this is an indication that all of the 216.77 addresses scanning for proxies are in fact the same machine.

The practical detect by Mark Bazant talks about how nmap is able to do this.

http://cert.uni-stuttgart.de/archive/intrusions/2003/07/msg00154.html

The 24.90.122.137 IP hit a lot more targets was more active in a 2 hours span.

It was identified by p0f as

24.90.122.137:4277 - Windows 2000 SP2+, XP SP1 (seldom 98 4.10.2222)

Conclusion: The 24.90.122.137 attacker is NOT the same as the other top offender(s) from 216.77.X.X

On the other hand, it is possible that all of the 216.77.X.X attackers are in fact the same computer.

5- Attack Mechanism

The attackers send a single TCP packet with the SYN flag set hopping to get an ACK in return. This would prove that a service is running on that and is trying to complete a 3-way handshake.

6- Correlations

Several CVE number exist for the ISA, Squid, Proxy, Cisco, products that use on of these ports. Several Papers exist talking about looking for open proxies and several other GCIA practicals have been written about this scan
(Alfred Koo is one example)

References (sample, not an exhaustive list)

Name	CVE-2002-0068
Description	Squid 2.4 STABLE3 and earlier allows remote attackers to cause a denial of service (core dump) and possibly execute arbitrary code with an ftp:// URL with a larger number of special characters, which exceed the buffer when Squid URL-escapes the characters.

Name	CVE-2002-0916
Description	Format string vulnerability in the allowuser code for the Stellar-X msntauth authentication module, as distributed in Squid 2.4.STABLE6 and earlier, allows remote attackers to execute arbitrary code via format strings in the user name, which are not properly handled in a syslog call.

Name	CAN-2002-0735 (under review)
Description	Format string vulnerability in the logging() function in C-Note Squid LDAP authentication module (squid_auth_LDAP) 2.0.2 and earlier allows

	remote attackers to cause a denial of service and possibly execute arbitrary code by triggering log messages.
References	• VULN-DEV:20020506 ldap vulnerabilities • URL:http://marc.theaimsgroup.com/?l=vuln-dev&m=102070267500932&w=2 • VULNWATCH:20020506 [VulnWatch] ldap vulnerabilities • URL:http://archives.neohapsis.com/archives/vulnwatch/2002-q2/0053.html • BUGTRAQ:20020506 ldap vulnerabilities • URL:http://online.securityfocus.com/archive/1/271173 • BID:4679 • URL:http://www.securityfocus.com/bid/4679 • XF:squidauthldap-logging-format-string(9019) • URL:http://www.iss.net/security_center/static/9019.php
Phase	Proposed (20020726)
Votes	ACCEPT(2) Cole, Armstrong NOOP(3) Cox, Wall, Foat
Comments	

Micosoft ISA

Name	CVE-2001-0239
Description	Microsoft Internet Security and Acceleration (ISA) Server 2000 Web Proxy allows remote attackers to cause a denial of service via a long web request with a specific type.

Name	CVE-2001-0658
Description	Cross-site scripting (CSS) vulnerability in Microsoft Internet Security and Acceleration (ISA) Server 2000 allows remote attackers to cause other clients to execute certain script or read cookies via malicious script in an invalid URL that is not properly quoted in an error message.

Other proxies :

Wingate (Port 1080)

Name	CVE-1999-0291
Description	The WinGate proxy is installed without a password, which allows remote attackers to redirect connections without authentication.

7- Evidence of Active Targeting

The attacks looks like part of scan that was targeted at proxy ports, most likely part of a wider scan, but this is not possible to prove given the available data. There is no evidence that any host was actively targeted.

No evidence was found that any reply was sent to the attackers and no follow-up probing occurred to indicate any action was taken based on this reconnaissance. There is no evidence that the attackers knew of a proxy in the scan range or that one was found.

8- Severity

severity = (+ lethality) – (system countermeasures + network countermeasures)

Criticality = 4

The attack looks for a critical service that may or may not be vulnerable

Lethality = 1

This is mostly reconnaissance. We need to lookout for follow-ups
No evidence was found that follow-up scans or exploit existed.
It is possible that the evidence is present in subsequent logs, but I cannot base my evaluation on this.

System Countermeasures = 2

My assumption is that this systems are relatively well protected.
I cannot find any evidence of immediate compromise so an average score of 2 was given.

Network Countermeasures = 2

Criticality: (4+1) – (2+2) = 1

Note: Since these logs come from incidents.org,
This estimate is admittedly less precise than the other 2 detects. This is because

9-Defensive recommendations

-Make sure you are not an Open Proxy (do not allow connection from the external interface to elsewhere, only from the internal interface of the proxy)

-Check your logs often for suspicious activity

-Keep you proxy product up-to-date

10-Multiple Choice Test Question

Which are some factors looked at to passively fingerprint
an Operating System.

a)window size
b)time to live
c)maximum segment size
d)don't fragment flag
e)window scaling
f)sackOK flag
g)nop flag
h)declared packet size
i) all of the above

Answer: I)

References:

Michal Zalewski. p0f README.. Version 1.8.3
http://www.stearns.org/p0f/README

Common Vulnerability and Exposure, Mitre Corportation
http://www.cve.mitre.org/cve/

Alfred Koo. Practical Detect for GCIA. July 30, 2003
http://cert.uni-stuttgart.de/archive/intrusions/2003/07/msg00374.html

Mark Bazant . Practical Detect for GCIA. July 2003
http://cert.uni-stuttgart.de/archive/intrusions/2003/07/msg00154.html

11- Question & Answers from intrusions@incidents.org

I did not get any replies from intrusions@incidents.org
My were coworkers were then asked to find questions and I answered.
I posted my detects on 2003-11-07 and had only received 1 reply by 2003-11-12.

Both Comments are from the 2003

A) Comment 1 : Your multiple choice questions is too easy

I originally had posted a question about “Which tools is used to passively fingerprint an OS”. This did not test GCIA level knowledge

I changed the question to reflect the knowledge that GCIA students are expected to have. I asked for the variables that p0f looks at instead.

B) Comment 2: Are the addresses spoofed or not?

I had 2 intruders running the scan. 1 IP seems real, there is a possibility that the other one was spoofed using Nmap. Several sections were slightly modified to make this clearer.

Part 3 : Analyze this

1.0 Executive Summary

These are the results of the Intrusion Detection System audit that your University mandated through SANS/GIAC. Although these results are very preliminary, some interesting conclusions and defensive recommendations can be made.

286170 Alerts, 11699732 Port Scans and 21800 Out Of Spec packets were analyzed using a variety of tools and methods, but full correlation could not be made given that several important pieces of information were kept secret from the auditors at your request.

The following information was found

a) VERY large variety of activity

More investigation is needed to determine if this activity follows University Guidelines for Faculty and staff and/or acceptable use policy.
Examples of activity that might need to be reviewed:

-IRC usage
-Peer 2 Peer Usage

b) Some computers need to be looked at closely

Some highly suspect systems were found and need a closer look (forensics analysis or at the very least be rebuilt).

Computers with MS-Blaster Alert
Computer with TFTP alerts

More detail will be given about these computers in the relevant sections of the report.

The following Defensive recommendation can be made following this preliminary audit:

Recommendation 1: Full Audit

A full log AND policy audit needs to be commissioned with contextual information given to the person performing this audit. This would allow the creation of a

report with a greater level of detail and to check the logs for sign of policy violations as well as system compromise

Recommendation 2: Tighten up the perimeter

Add some layers of protection between users and the Internet.
It almost looks like anybody from the outside can run any attack towards computers on the inside of the network. I would recommend forcing users to navigate through a proxy for all allowed services and to block all other inbound/outbound traffic (TFTP and IRC, for instance)
Placing users behind a firewall and NAT might also help mask the internals of the networks from outside reconnaissance (of which there is quite a bit)

Recommendation 3: Tune Snort

Once this audit of services and computer logs is complete, you need to tune the IDS avoid some false positives. Some of the alerts are not false positives in the pure IDS sense of the word (the packet actually triggered a specific rule). But if the type of traffic that triggered the rule is normal and allowed, that rule should be disabled

2.0 Origin of the Logs

These logs originated from www.incidents.org and were dated from 2003-10-19 to 2003-10-23 inclusively. Three log formats (generated by SNORT) from these dates were analyzed. These logs represent the Alerts (named alert.0310dd), Portscans (named scans.0310dd) and Out of Spec packets (named OOS_Report_2003_10_XX).

alert.031019	scans.031019	OOS_Report_2003_10_19
alert.031020	scans.031020	OOS_Report_2003_10_20
alert.031021	scans.031021	OOS_Report_2003_10_21
alert.031022	scans.031022	OOS_Report_2003_10_22
alert.031023	scans.031023	OOS_Report_2003_10_23

3.0 Traffic and Network Analysis

Several techniques were used to analyse the traffic. The most important of these techniques involves trying to find the services that are supposed to be running inside monitored network. This would give us a good idea of what

kind of alerts should be happening and also allow us to eliminate some alerts as false positives.

Context:
While trying to analyze and correlate the portscans logs to the alert logs, we soon found that the 130.85.0.0/16 subnet is the most frequently occurring subnet. Combine this with the fact that the MY.NET subnet is the one that was generating the most alerts (by far) and you get a pretty strong correlation that 130.85 is in fact MY.NET

Looking up this subnet in WHOIS yields:

Search results for: 130.85.80.51

OrgName: University of Maryland Baltimore County
OrgID: UMBC
Address: UMBC University Computing
City: Baltimore
StateProv: MD
PostalCode: 21250
Country: US

NetRange: 130.85.0.0 - 130.85.255.255
CIDR: 130.85.0.0/16
NetName: UMBCNET
NetHandle: NET-130-85-0-0-1
Parent: NET-130-0-0-0-0
NetType: Direct Assignment
NameServer: UMBC5.UMBC.EDU
NameServer: UMBC4.UMBC.EDU
NameServer: UMBC3.UMBC.EDU
Comment:
RegDate: 1988-07-05
Updated: 2000-03-17

TechHandle: JJS41-ARIN
TechName: Suess, John J.
TechPhone: +1-410-455-2582
TechEmail: jack@umbc.edu

ARIN WHOIS database, last updated 2003-11-09 19:15
Enter ? for additional hints on searching ARIN's WHOIS database.

This is another indication that we are in fact guessing correctly in assuming that 130.85 is the same as MY.NET because the requirements paper listed the monitored subnet as a University. One might wonder what the point of hiding the source IP in the alert files is, if one can just find out with the other files?
Was this done on purpose? Or is this another case of broken script/cron job?

3.1 Services

The alerts and Portscans were checked for patterns in source port and destination port in order to see if this could help us determine what kind of services is running on those computers.

Sylvain Ranier best described the technique to find servers

Summary: The assumption is that a server running on port X will probably have alerts destined to that port or alerts source from that port. Portscans might also be triggered from that source port to multiple IP and Destination ports.

This will give us a list of probable servers.
We cannot really confirm this list without actually probing the network in question so this will not be done. We fully expect to be wrong on more than a few accounts

Here are the results:

Destination port

SourceIP	Count	#Dest	Dest Port
MY.NET.70.49	5	3	21
MY.NET.80.51	2	2	22
MY.NET.80.51	1	1	25
MY.NET.100.13	2	1	25
MY.NET.100.230	18	1	25
MY.NET.24.20	22	3	25
MY.NET.25.10	10	1	25
MY.NET.25.67	3	1	25
MY.NET.25.68	6	2	25
MY.NET.25.69	3	2	25
MY.NET.25.70	8	1	25
MY.NET.25.71	11	4	25
MY.NET.25.73	1	1	25
MY.NET.80.51	1	1	53

MY.NET.84.235	6	2	80
MY.NET.97.150	1	1	80
MY.NET.97.228	1	1	80
MY.NET.75.103	1	1	80
MY.NET.53.20	4	1	80
MY.NET.53.21	2	1	80
MY.NET.70.176	1	1	80

by Source Port

SourceIP	Count	#Dest	Source Port
MY.NET.12.4	3	1	110
MY.NET.60.17	1	1	110
MY.NET.12.6	26	3	25
MY.NET.24.20	2	1	25
MY.NET.100.165	17	3	80
MY.NET.150.83	2	1	80
MY.NET.162.67	5	1	80
MY.NET.24.34	36	15	80
MY.NET.24.44	32	7	80
MY.NET.29.3	13	3	80
MY.NET.5.20	18	5	80
MY.NET.60.14	4	1	80

By removing the IP with a small number of alerts, we get the following probable servers running these services.

Port 25 : SMTP servers

SourceIP	Count	#Dest	Dest Port
MY.NET.24.20	22	3	25
MY.NET.25.10	10	1	25
MY.NET.25.71	11	4	25
MY.NET.12.6	26	3	25

Port 110 : Pop Servers

SourceIP	Count	#Dest	Dest Port
MY.NET.12.4	3	1	110

Port 80 Web Servers

MY.NET.84.235	6	2	80
MY.NET.53.20	4	1	80
MY.NET.53.21	2	1	80
MY.NET.70.176	1	1	80
MY.NET.100.165	17	3	80
MY.NET.24.34	36	15	80
MY.NET.24.44	32	7	80
MY.NET.29.3	13	3	80
MY.NET.5.20	18	5	80

Port 53 : DNS servers

These machines are obviously DNS servers as per the ports scans generated and the WHOIS lookup.

130.85.1.3
130.85.1.4

FTP servers :

SourceIP	**Count**	**#Dest**	**Dest Port**
MY.NET.70.49	5	3	21

4.0 Highest Volume Alerts

The Top 5 issues in terms of volume of alerts will be briefly discussed here. This criteria of alert volume is being used under the assumption that some of the most damaging attacks, reconnaissance and Denial of Service activities will probably be in this category. Investigating these issues first has the additional side-benefit that if a false positive is found, we can then tune the IDS to ignore this alerts. This would greatly help future analyst, as they would have to sort through less data.

Another set of alerts will be discussed in the “Top Talkers” section as the top talkers are not necessarily one and the same.

Here is a table of all Alerts that generated, sorted by activity and volume

Count	#sources	# targets	Signature
199206	884	132248	SMB Name Wildcard
28546	619	959	SMB C access
15606	447	2	MY.NET.30.4 activity
11562	1412	937	EXPLOIT x86 NOOP
7131	1	3	connect to 515 from inside
5726	100	1	MY.NET.30.3 activity
4518	26	111	TCP SRC and DST outside network
3266	4	1830	External RPC call
3172	99	115	High port 65535 tcp - possible Red Worm - traffic
2009	84	327	Possible trojan server activity
1825	102	1498	ICMP SRC and DST outside network
752	150	59	NMAP TCP ping!
494	20	25	SUNRPC highport access!
455	60	54	Null scan!
438	75	79	High port 65535 udp - possible Red Worm - traffic
342	1	6	[UMBC NIDS IRC Alert] IRC user /kill detected
182	4	3	[UMBC NIDS IRC Alert] XDCC client detected attempting to IRC
105	35	6	FTP passwd attempt
103	48	1	[UMBC NIDS] External MiMail alert
84	2	84	Back Orifice
83	17	27	TFTP - Internal UDP connection to external tftp server
74	57	46	Incomplete Packet Fragments Discarded
62	38	24	Tiny Fragments - Possible Hostile Activity
55	7	2	[UMBC NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC
53	10	7	EXPLOIT x86 stealth noop
51	3	13	NETBIOS NT NULL session
38	5	2	DDOS shaft client to handler
37	2	6	[UMBC NIDS IRC Alert] Possible drone command detected.
27	25	18	EXPLOIT x86 setuid 0
26	21	20	EXPLOIT x86 setgid 0
25	8	13	EXPLOIT NTPDX buffer overflow
14	3	2	DDOS mstream client to handler
14	5	1	FTP DoS ftpd globbing
13	3	3	TFTP - Internal TCP connection to external tftp server
12	2	1	[UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected.
11	7	7	TFTP - External UDP connection to internal tftp server
10	3	5	Attempted Sun RPC high port access
10	7	6	RFB - Possible WinVNC - 010708-1
5	1	3	HelpDesk MY.NET.70.49 to External FTP
4	1	2	[UMBC NIDS IRC Alert] K\:line'd user detected
4	4	3	NIMDA - Attempt to execute cmd from campus host
3	2	3	[UMBC NIDS] Internal MSBlast Infection Request
2	2	1	External FTP to HelpDesk MY.NET.53.29
2	1	2	connect to 515 from outside
2	2	1	External FTP to HelpDesk MY.NET.70.50
2	2	2	TFTP - External TCP connection to internal tftp server
2	1	1	Traffic from port 53 to port 123

2	2	2	Probable NMAP fingerprint attempt
2	2	1	External FTP to HelpDesk MY.NET.70.49
1	1	1	IRC evil – running XDCC
1	1	1	Bugbear@MM virus in SMTP
1	1	1	[UMBC NIDS IRC Alert] Possible trojaned box detected attempting to IRC

SMB Name Wildcard

SMB Name Wildcard is a signature that is triggered when a NetBIOS computers tries to request the Netbios NameTable Information from a remote computer. This table shows the top offenders for the attacks.

Source	Count	# of Targets
MY.NET.80.51	115618	115610
MY.NET.150.133	72066	13748
MY.NET.29.2	3100	2147
MY.NET.84.224	1290	5
MY.NET.150.198	474	234
MY.NET.42.9	193	14
MY.NET.17.34	143	5
MY.NET.84.154	141	32
MY.NET.111.65	133	27
MY.NET.150.44	118	61
MY.NET.84.202	116	5
MY.NET.29.3	114	12
MY.NET.162.62	102	5
MY.NET.150.42	99	4

Sample activity for the top offenders of this shows that ALL of the targets are external and mostly sequential by subnet. This tends to show that some script or automated tool is trying to find out a lot of NetBios information. This could be the sign of reconnaissance, system compromise or that some worm/virus infected these machines.

Possible scenerio: a netbios Worm?

sample of the alerts

2003-10-23 10:19:33.963	SMB Name Wildcard	MY.NET.80.51	1036	16.229.19.199	137
2003-10-23 10:19:33.963	SMB Name Wildcard	MY.NET.80.51	1035	52.94.88.177	137
2003-10-23 10:19:34.717	SMB Name Wildcard	MY.NET.80.51	1036	16.229.19.204	137
2003-10-23 10:19:35.167	SMB Name Wildcard	MY.NET.80.51	1036	16.229.19.207	137
2003-10-23 10:19:35.317	SMB Name Wildcard	MY.NET.80.51	1036	16.229.19.208	137
2003-10-23 10:19:36.860	SMB Name Wildcard	MY.NET.80.51	1035	52.94.88.195	137
2003-10-23 10:19:36.860	SMB Name Wildcard	MY.NET.80.51	1036	16.229.19.218	137
2003-10-23 10:19:37.313	SMB Name Wildcard	MY.NET.80.51	1036	16.229.19.221	137
2003-10-23 10:19:37.313	SMB Name Wildcard	MY.NET.80.51	1035	52.94.88.198	137
2003-10-23 10:19:37.463	SMB Name Wildcard	MY.NET.80.51	1035	52.94.88.199	137
2003-10-23 10:19:37.613	SMB Name Wildcard	MY.NET.80.51	1036	16.229.19.223	137
2003-10-23 10:19:37.613	SMB Name Wildcard	MY.NET.80.51	1035	52.94.88.200	137
2003-10-23 10:19:38.213	SMB Name Wildcard	MY.NET.80.51	1036	16.229.19.227	137
2003-10-23 10:19:38.213	SMB Name Wildcard	MY.NET.80.51	1035	52.94.88.204	137
2003-10-23 10:19:38.363	SMB Name Wildcard	MY.NET.80.51	1035	52.94.88.205	137

Recommended action: Closer examination of the top 5 offenders are listed above. They are either compromised, infected or being misused.

SMB C$ Access

This alert is triggered when somebody tries to mount the C$ share that is usually present on unhardened Windows Machines. Administrators are the only people authorized to mount these shares.
In our case, several IP addresses are scanning several subnets inside the University. There is usually no good reason to do this. If this were a legitimate administrative access, there would probably be no need to scan subnets from the outside, as he/she would probably already know which hosts they want to access.

Probable Scenario:

This looks like a classic case of reconnaissance for Administrative shares.
The attacker would run more directed probing or password grinding.

The assumption for the attacker is that if these C$ shares are available from the outside, the administrative user name is probably easy to guess (admin, administrator, etc) and a strong password policy is probably not enforced.

Sample Alerts take from the logs over a span of 2 minutes

SMB C access	12.84.41.64	1576	MY.NET.70.128	139
SMB C access	12.84.41.64	1582	MY.NET.70.146	139
SMB C access	12.84.41.64	1584	MY.NET.70.154	139
SMB C access	12.84.41.64	1588	MY.NET.70.177	139
SMB C access	12.84.41.64	1564	MY.NET.71.230	139

SMB C access	12.84.41.64	1589	MY.NET.70.180	139
SMB C access	12.84.41.64	1596	MY.NET.70.206	139
SMB C access	12.84.41.64	1605	MY.NET.70.235	139
SMB C access	12.84.41.64	1570	MY.NET.70.72	139
SMB C access	12.84.41.64	1572	MY.NET.70.82	139
SMB C access	12.84.41.64	1576	MY.NET.70.128	139
SMB C access	12.84.41.64	1588	MY.NET.70.177	139
SMB C access	12.84.41.64	1565	MY.NET.71.237	139
SMB C access	12.84.41.64	1594	MY.NET.70.197	139
SMB C access	12.84.41.64	1629	MY.NET.69.171	139

Correllations:

http://www.digitaltrust.it/arachnids/IDS339/event.html
http://www.snort.org/snort-db/sid.html?sid=533
http://www.giac.org/practical/GCIA/Andrew_Jones_GCIA.pdf

MY.NET.30.4 activity

This alert is presumably triggered when a host that is not supposed to generate activity actually is. A lot of people from the outside are connecting to this computer on ports 80, 524 and 51443.

Port 524 is used by Novell and should not be going outside the local network

MY.NET.30.4 activity	172.142.110.232	1471	MY.NET.30.4	51443
MY.NET.30.4 activity	172.142.110.232	1471	MY.NET.30.4	51443
MY.NET.30.4 activity	172.142.110.232	1474	MY.NET.30.4	51443
MY.NET.30.4 activity	172.142.110.232	1474	MY.NET.30.4	51443
MY.NET.30.4 activity	172.142.110.232	1474	MY.NET.30.4	51443
MY.NET.30.4 activity	172.142.110.232	1474	MY.NET.30.4	51443
MY.NET.30.4 activity	172.142.110.232	1474	MY.NET.30.4	51443
MY.NET.30.4 activity	172.142.110.232	1471	MY.NET.30.4	51443
MY.NET.30.4 activity	172.142.110.232	1474	MY.NET.30.4	51443
MY.NET.30.4 activity	172.142.110.232	1474	MY.NET.30.4	51443
MY.NET.30.4 activity	172.142.110.232	1474	MY.NET.30.4	51443
MY.NET.30.4 activity	172.142.110.232	1474	MY.NET.30.4	51443

Port 51443 is used by Novell Secure Folder option.

Recommendation:

Take MY.NET.30.4 off the network and rebuild it, unless it is supposed to connect to this external server.

http://www.tek-tips.com/gfaqs.cfm/lev2/3/lev3/19/pid/871/fid/3352
http://archives.neohapsis.com/archives/incidents/2000-10/0221.html

EXPLOIT x86 NOOP

Some NO-OP instruction was found in a packet. This is x86 code that is found in a lot of known exploits. These alerts are mostly from external machines going to internal machines. Several known false positives are documented for this, but there is a possibility that this can be somebody that has already performed reconnaissance and is now trying to exploit some machines on a target list.

EXPLOIT x86 NOOP	12.11.171.7	21874	MY.NET.70.164	135	
EXPLOIT x86 NOOP	12.110.52.132	59809	MY.NET.80.107	135	
EXPLOIT x86 NOOP	12.13.158.98	4295	MY.NET.69.175	135	
EXPLOIT x86 NOOP	12.134.34.92	4722	MY.NET.66.61	135	
EXPLOIT x86 NOOP	12.153.9.9	3288	MY.NET.80.107	135	
EXPLOIT x86 NOOP	12.161.217.145	27876	MY.NET.150.150		135
EXPLOIT x86 NOOP	12.166.203.254	40570	MY.NET.11.9	135	
EXPLOIT x86 NOOP	12.168.149.2	49324	MY.NET.53.31	135	
EXPLOIT x86 NOOP	12.174.232.15	3556	MY.NET.70.235	445	
EXPLOIT x86 NOOP	12.174.232.15	4229	MY.NET.69.224	445	
EXPLOIT x86 NOOP	12.215.187.116	3981	MY.NET.53.31	135	
EXPLOIT x86 NOOP	12.219.244.158	2644	MY.NET.111.156		135
EXPLOIT x86 NOOP	12.22.118.230	12860	MY.NET.11.9	135	
EXPLOIT x86 NOOP	12.223.197.216	1282	MY.NET.190.97	135	
EXPLOIT x86 NOOP	12.223.197.216	1287	MY.NET.190.102		135
EXPLOIT x86 NOOP	12.223.212.39	4672	MY.NET.152.45	135	
EXPLOIT x86 NOOP	12.31.156.167	54088	MY.NET.152.45	135	

Correlations:

http://www.derkeiler.com/Mailing-Lists/securityfocus/focus-ids/2002-04/0041.html

Connect to 515 from inside

This is a very well known exploit for the LPD service that runs on port 515 on most UNIX/Linux older Linux/Unix distributions.

The activity is pretty worrisome because there is no reason for the attacker to connect to a printer on the outside. The Ramen Worm and toolkit also use some exploits for this service

connect to 515 from inside	MY.NET.162.41721	128.183.110.242	515
connect to 515 from inside	MY.NET.162.41721	128.183.110.242	515
connect to 515 from inside	MY.NET.162.41721	128.183.110.242	515
connect to 515 from inside	MY.NET.162.41721	128.183.110.242	515
connect to 515 from inside	MY.NET.162.41721	128.183.110.242	515
connect to 515 from inside	MY.NET.162.41721	128.183.110.242	515
connect to 515 from inside	MY.NET.162.41721	128.183.110.242	515
connect to 515 from inside	MY.NET.162.41721	128.183.110.242	515
connect to 515 from inside	MY.NET.162.41721	128.183.110.242	515
connect to 515 from inside	MY.NET.162.41721	128.183.110.242	515
connect to 515 from inside	MY.NET.162.41721	128.183.110.242	515
connect to 515 from inside	MY.NET.162.41721	128.183.110.242	515
connect to 515 from inside	MY.NET.162.41721	128.183.110.242	515
connect to 515 from inside	MY.NET.162.41721	128.183.110.242	515
connect to 515 from inside	MY.NET.162.41721	128.183.110.242	515
connect to 515 from inside	MY.NET.162.41721	128.183.110.242	515
connect to 515 from inside	MY.NET.162.41721	128.183.110.242	515

Recommended Action: Host MY.NET.162.41 is probably compromised or being misused for malicious activity. Take it off the network for forensics unless it really needs to print on the destination host.

Correlations:

http://lists.insecure.org/lists/incidents/2001/Jun/0240.html
http://www.cert.org/incident_notes/IN-2001-01.html

MY.NET.30.3 activity

Similar to the 30.4 activity except that is only directed to computer on the outside on port 524 which is a Novell port.

MY.NET.30.3 activity	68.57.90.146	1032	MY.NET.30.3	524
MY.NET.30.3 activity	68.57.90.146	1032	MY.NET.30.3	524
MY.NET.30.3 activity	68.57.90.146	1032	MY.NET.30.3	524
MY.NET.30.3 activity	68.55.53.222	1032	MY.NET.30.3	524
MY.NET.30.3 activity	165.247.89.143	2727	MY.NET.30.3	524
MY.NET.30.3 activity	165.247.89.143	2727	MY.NET.30.3	524
MY.NET.30.3 activity	165.247.89.143	2727	MY.NET.30.3	524
MY.NET.30.3 activity	165.247.89.143	2727	MY.NET.30.3	524
MY.NET.30.3 activity	68.55.233.51	63785	MY.NET.30.3	524
MY.NET.30.3 activity	68.55.233.51	63785	MY.NET.30.3	524
MY.NET.30.3 activity	68.55.233.51	63785	MY.NET.30.3	524
MY.NET.30.3 activity	68.55.233.51	63785	MY.NET.30.3	524

Other alerts of interest

The following alerts were considered interesting for various reasons:

a) IRC alerts

A query was run to find all alerts containing the keyword IRC.
Interestingly enough, most of those alerts start with the prefix [UMBC NIDS IRC Alert]. To me, this is an indication that somebody actually paid special attention to these rules and modified the standard SNORT message. Does this mean that the University has had problems with this in the past. What is the IRC policy at UMBC?

Here is a table containing the IRC alerts and their associated counts

342	[UMBC NIDS IRC Alert] IRC user /kill detected
182	[UMBC NIDS IRC Alert] XDCC client detected attempting to IRC
55	[UMBC NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC
37	[UMBC NIDS IRC Alert] Possible drone command detected.
12	[UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected.
4	[UMBC NIDS IRC Alert] K\:line'd user detected
1	[UMBC NIDS IRC Alert] Possible trojaned box detected attempting to IRC
1	IRC evil - running XDCC

Some of these alerts seem particularly worrisome even if the counts are not extremely high. IRC is a brutal environment, and the University needs to make sure that users do not commit crimes on it (Warez, Attacks, Zombies, etc)

Examples of hosts that would need to be examined:

MY.NET.15.198
MY.NET.97.91
MY.NET.80.149
MY.NET.82.79
MY.NET.80.16
MY.NET.81.18
MY.NET.97.236
MY.NET.97.126
MY.NET.97.135
MY.NET.97.21
MY.NET.97.219
MY.NET.29.2
MY.NET.163.249

TFTP connection:

TFTP connections are commonly used to transfer small files or simple transfer from point A to point B. Usually this a way to update routers or other simple devices. But they are also used to transfer worm and virus code as seen in the Recent Blaster/Welchia RPC worms. (See detect#2)

Here is a summary of the TFTP alerts generated by Snort.

83	TFTP - Internal UDP connection to external tftp server
13	TFTP - Internal TCP connection to external tftp server
11	TFTP - External UDP connection to internal tftp server
2	TFTP - External TCP connection to internal tftp server

I see no good reason why TFTP connection should occur between the inside and the outside of the University.

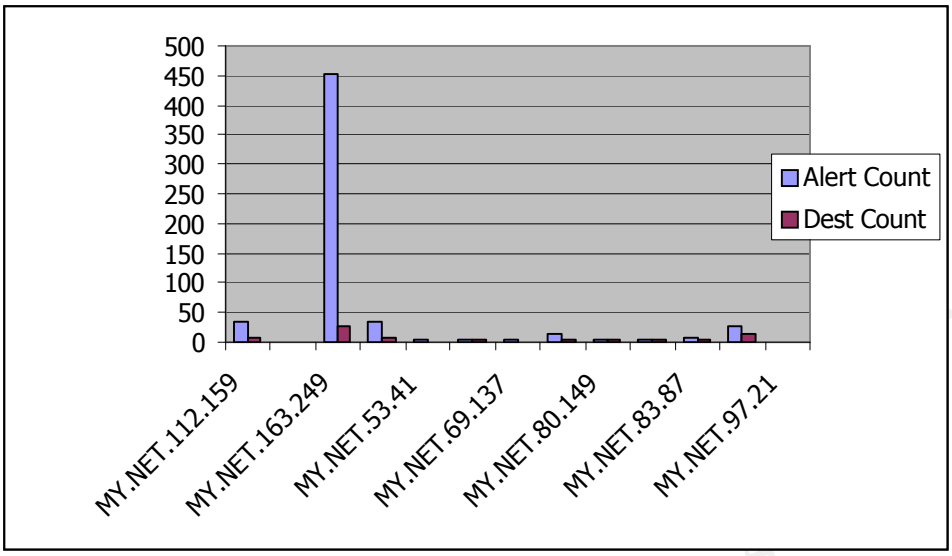
These two source were used as a test case because they had the highest occurrences of TFTP alerts

MY.NET.69.156
MY.NET.153.195

The first host does not seem to have any related activity, but the second host also seems to have MS-Blaster related alerts. This is significant because MS-Blaster and other worms have a TFTP component to download the worm code after being exploited.
These will be discussed in the next section, but I recommend that all computers with TFTP activity be checked for worms or signs of compromise.
This brings us to the link graph that follows.

The graph represents a link between TFTP activity from the outside and further activity from the target hosts.

The targets of the TFTP activity are on the X axis, and an aggregate view of their activity is represented by the 2 bar charts. The blue charts represents the alert count and the red bar represents the number of distinct destinations.



As you can tell from this graph, host MY.NET.112.159 requires further follow up and investigation.

The SQL used to generate this graph is as follows

```
select sourceip,count(*) as '# of alerts',count(distinct destip) as '# of dest'
from alerts
where sourceip in
(select destip from alerts
where alert like '%tftp%' and destip like 'MY.NET.%')
group by sourceip
```

[**UMBC NIDS] Internal MSBlast Infection**

The following computers have generated the MSBlast alert.
This alert seems to trigger on having port 4444 open (known backdoor for Msblaster)

[UMBC NIDS] Internal MSBlast Infection Request MY.NET.163.249 4444 130.67.101.88 4865

This first computer seems to be infected with a lot more than Blaster
It is also generating the following alerts:

Possible trojan server activity	MY.NET.163.249	6667	200.163.61.175	27374
SMB Name Wildcard	MY.NET.163.249	137	81.53.115.246	137

Both of these alerts could be an indication of further compromise
The Possible Trojan server alert uses the Subseven port.

The next computer is also generating several SMB name Wildcard, TFTP alerts as described in the previous section.

2003-10-22 23:37:58.173 [UMBC NIDS] Internal MSBlast Infection Request
MY.NET.153.195 4444 67.30.249.193 4610

5.0 Top Talkers

The top Talkers section only looks at the volume of activity per category of file analyzed.
Results should be fairly similar to the important alerts section because in both cases the volume of logged data was used in order to sort through the data.

Here are the results:

Alerts				Portscans		
Count	SourceIP	Alert	# targets	Count	Source	# targets
115618	MY.NET.80.51	SMB Name Wildcard	115610	2166933	130.85.1.3	85807
72066	MY.NET.150.133	SMB Name Wildcard	13748	1294187	130.85.70.154	285689
7128	MY.NET.162.41	connect to 515 from inside	3	966595	130.85.163.107	966532
4279	169.254.244.56	TCP SRC and DST outside network	4	888185	130.85.84.194	884152
3100	MY.NET.29.2	SMB Name Wildcard	2147	669973	130.85.163.249	586885
2934	68.55.85.180	MY.NET.30.4 activity	2	273705	130.85.42.1	99301
2837	193.114.70.169	External RPC call	1592	213577	130.85.70.129	85732
2743	68.54.91.147	MY.NET.30.4 activity	1	211571	130.85.1.5	21215
1290	MY.NET.84.224	SMB Name Wildcard	5	175961	130.85.80.149	92535

1224 68.57.90.146 MY.NET.30.3 activity 1 171526 130.85.111.72 171525

As we can see, several of the alerts in question are the same as in 4.1
The only alerts we have not already discussed are:

TCP SRC and DST outside Network

This alert occurs when both the source and destination IP are outside of the network (obviously).

What is interesting here is that there are repeated alerts for this, but all of them have the same source, the 2 same destinations.

TCP SRC and DST outside network	169.254.244.56	2476	211.91.144.72 996
TCP SRC and DST outside network	169.254.244.56	2477	218.16.124.131 21

As you can see, the destination 211.91.144.72 is always contacted on port 996 and 218.16.124.131 is always contacted on port 21.

The IP address is obviously spoofed as it resolves to this the LINKLOCAL as seen in the following WHOIS query.

NetRange: 169.254.0.0 - 169.254.255.255
CIDR: 169.254.0.0/16
NetName: LINKLOCAL
NetHandle: NET-169-254-0-0-1
Parent: NET-169-0-0-0-0
NetType: IANA Special Use
NameServer: BLACKHOLE-1.IANA.ORG
NameServer: BLACKHOLE-2.IANA.ORG
Comment: Please see RFC 3330 for additional information.
RegDate: 1998-01-27
Updated: 2002-10-14

OrgTechHandle: IANA-ARIN
OrgTechName: Internet Corporation for Assigned Names and Number
OrgTechPhone: +1-310-823-9358
OrgTechEmail: res-ip@iana.org

Further Investigation is needed to determine who the source actually is and what is actually occurring.

External RPC call

This alert seems to trigger when SunRPC (port 111) is probed from the outside as seen in the logs. SunRPC has several very well-known vulnerabilities.

External RPC call	193.114.70.169 4253	MY.NET.16.16	111
External RPC call	193.114.70.169 4298	MY.NET.16.33	111
External RPC call	193.114.70.169 2590	MY.NET.21.0	111
External RPC call	193.114.70.169 4262	MY.NET.16.19	111

Looking for additional proof of compromised systems, we ran a query for all activity from this subset of the destinations and it returned no results.

It looks like this is just another case of external reconnaissance.

6.0 Out of Spec Packets

Here is the Summary Output of our Perl Script.
It is a count of Packets in all of the OOS files with the associated flag combination. The rest of the output details what source,destinations and ports were generated with a count. Since the output was a lot less informative, it was put in Appendix C

Given the scope of this report, only the top 2 types of flag combinations that occurred the most often will be analyzed.

FLAGS
flags: 12****S* : 21254
flags: ******** : 296
flags: ****P*** : 122
flags: 12***R** : 37
flags: 12*A*R** : 15
flags: **U*P*SF : 7
flags: ***A**SF : 7
flags: ***AP*SF : 4
flags: *2UAPRSF : 3
flags: 1**AP*SF : 2
flags: 12*AP**F : 2
flags: 1**A**SF : 2
flags: 12**PR** : 2
flags: *2U***SF : 2
flags: 12U*P*** : 2
flags: *2***RSF : 2
flags: 12*AP*S* : 2
flags: 12UA*RS* : 2

flags: 12***R*F : 2
flags: 12U*P**F : 2
flags: *2*APRSF : 2
flags: 12UAPR*F : 2
flags: **U***** : 1
flags: 12*A**** : 1
flags: **UA**SF : 1
flags: 12UAPRSF : 1
flags: 12****** : 1
flags: 12**PRS* : 1
flags: 1**A*RSF : 1
flags: *2*A*RSF : 1
flags: *2UAP*SF : 1
flags: **UAPRSF : 1
flags: 12UA*R** : 1
flags: 12UAPRS* : 1
flags: 12*****F : 1
flags: 1**APRSF : 1
flags: 1*****SF : 1
flags: 12*A*RS* : 1
flags: 12UA*RSF : 1
flags: 12**PRSF : 1
flags: 12U***** : 1
flags: *2*AP*SF : 1
flags: *****RSF : 1
flags: *2UA*RSF : 1
flags: 1*U*PRSF : 1
flags: 12**P*SF : 1
flags: 12U***S* : 1
flags: 12**P*S* : 1
flags: *2*A**SF : 1
flags: 12UAP*** : 1
flags: 1***P*SF : 1
**********REPORT
no_tcpopt : 332
tcpopt : 21468
total_packets : 21800

1) 12****S*

As we can see from the report above, the 12****S* flag combination is by far the most common.

This thread on a snort related mailing list explains that this combination is used for ECN (Explicit Network Congestion).

http://archives.neohapsis.com/archives/snort/2002-07/0617.html

While this post deals with the behaviour of ECN.

http://archives.neohapsis.com/archives/snort/2001-01/0409.html

ECN is a protocol (defined in *rfc2481 and rfc2914)* that deals with network congestion and flow control. It is a relatively new way dealing with network congestion and not every TCP stack deals with it correctly.

An interesting side-effect of using reserved bit and flags for ECN,according to that last link (Crist Clark) is that

> " Also remember as ECN come into more use, the threat represented by the reserved bits" also declines. Since more IP stack implementers will need to worry about the reserved bits, there should be better behavior from various IP stacks when confronted with the high-bits set. The primary malicious uses of the bits, fingerprinting and stealth, should become less effective."

Only time will tell if this is true or not.

Recommended Action: Verify that all of the hosts using these flags are actually trying to use ECN. If not, something is wrong.

2) ******** (No flags)

As discussed in Joanne_Schell's GCIA paper,this was probably part of some type of NULL scan that was too slow to trigger an associated alert.
I think that this analysis is fairly accurate.

7.0 WHOIS lookups for source IPs

WHOIS lookups

A sample of WHOIS lookups about some attackers was requested when the audit was commissioned. We also provide the www.dshield.org report on the specific external IP being investigated. You will find the listing below. In order to

determine which attackers to investigate, we built a query to find the Top 10 external attackers.

The Top5 REAL external IPs will then be chosen from the table below.

Count	IP	Alert
4279	169.254.244.56	TCP SRC and DST outside network
2934	68.55.85.180	MY.NET.30.4 activity
2837	193.114.70.169	External RPC call
2743	68.54.91.147	MY.NET.30.4 activity
1224	68.57.90.146	MY.NET.30.3 activity
1124	172.142.110.232	MY.NET.30.4 activity
1023	200.96.13.157	High port 65535 tcp - possible Red Worm – traffic
997	151.196.19.202	MY.NET.30.4 activity
764	209.6.97.168	EXPLOIT x86 NOOP
735	68.55.27.157	MY.NET.30.3 activity

Search results for: 68.55.85.180

Comcast Cable Communications, Inc. JUMPSTART-1 (NET-68-32-0-0-1)
68.32.0.0 - 68.63.255.255
Comcast Cable Communications, Inc. BALTIMORE-A-6 (NET-68-55-0-0-1)
68.55.0.0 - 68.55.255.255

ARIN WHOIS database, last updated 2003-11-09 19:15
Enter ? for additional hints on searching ARIN's WHOIS database.

Your IP (68.55.85.180) does not appear as an attacker in the DShield database.

Search Results for 193.114.70.169

inetnum: 193.114.70.160 - 193.114.70.191
netname: FIRST-PROCUREMENT-ASSOCIATES-LIMITED
descr: FIRST PROCUREMENT ASSOCIATES LIMITED
country: GB

admin-c: JB7221-RIPE
tech-c: AB480-RIPE
status: ASSIGNED PA
notify: ripe-notify@uk.psi.com
mnt-by: PSINET-UK-SYSADMIN
changed: sysadmin@uk.psi.com 19990903
source: RIPE
route: 193.114.0.0/15
descr: EUNETGB-114-AGG
origin: AS1290
mnt-by: PSINET-MNT
changed: network-ripe@uk.psi.com 20021015
source: RIPE
person: John Barke
address: FIRST PROCUREMENT ASSOCIATES LIMITED
address: 1St Andrews House
address: Vernon Gate
address: Derby
address: DE1 1UJ
phone: +44 1332 604 313
nic-hdl: JB7221-RIPE
notify: ripe-notify@uk.psi.com
mnt-by: PSINET-UK-SYSADMIN
changed: sysadmin@uk.psi.com 19990903
source: RIPE

Your IP (193.114.70.169) does not appear as an attacker in the DShield database.

Search results for: 68.54.91.147

Comcast Cable Communications, Inc. JUMPSTART-1 (NET-68-32-0-0-1)
68.32.0.0 - 68.63.255.255
Comcast Cable Communications, Inc. BALTIMORE-A-4 (NET-68-54-80-0-1)
68.54.80.0 - 68.54.95.255

ARIN WHOIS database, last updated 2003-11-09 19:15
Enter ? for additional hints on searching ARIN's WHOIS database.

Your IP (68.54.91.147) does not appear as an attacker in the DShield database.

Search results for: 68.57.90.146

Comcast Cable Communications, Inc. JUMPSTART-1 (NET-68-32-0-0-1)
68.32.0.0 - 68.63.255.255
Comcast Cable Communications, Inc. CHESTERFIELD-2 (NET-68-57-64-0-1)
68.57.64.0 - 68.57.127.255

ARIN WHOIS database, last updated 2003-11-09 19:15
Enter ? for additional hints on searching ARIN's WHOIS database.

Your IP (68.57.90.146) does not appear as an attacker in the DShield database.

OrgName: America Online
OrgID: AOL
Address: 22000 AOL Way
City: Dulles
StateProv: VA
PostalCode: 20166
Country: US

NetRange: 172.128.0.0 - 172.191.255.255
CIDR: 172.128.0.0/10
NetName: AOL-172BLK
NetHandle: NET-172-128-0-0-1
Parent: NET-172-0-0-0-0
NetType: Direct Allocation
NameServer: DAHA-01.NS.AOL.COM
NameServer: DAHA-02.NS.AOL.COM
NameServer: DAHA-07.NS.AOL.COM
Comment: ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
RegDate: 2000-03-24
Updated: 2003-08-08

TechHandle: AOL-NOC-ARIN

TechName: America Online, Inc.
TechPhone: +1-703-265-4670
TechEmail: domains@aol.net

OrgAbuseHandle: AOL382-ARIN
OrgAbuseName: Abuse
OrgAbusePhone: +1-703-265-4670
OrgAbuseEmail: abuse@aol.net

OrgNOCHandle: AOL236-ARIN
OrgNOCName: NOC
OrgNOCPhone: +1-703-265-4670
OrgNOCEmail: noc@aol.net

OrgTechHandle: AOL-NOC-ARIN
OrgTechName: America Online, Inc.
OrgTechPhone: +1-703-265-4670
OrgTechEmail: domains@aol.net

ARIN WHOIS database, last updated 2003-11-09 19:15
Enter ? for additional hints on searching ARIN's WHOIS database.

Your IP (172.142.110.232) does not appear as an attacker in the DShield database.

7.1 Analysis of the WHOIS

Most of these sources seem to be in the range for Major ISPs (1 AOL, 3 comcast).
Only 1 appears to be a company in England.
None of these IPs had been reported to Dshield for abuse/strange activity

8.0 Process used

In order to process the amount of data that was necessary to complete this assignment, several techniques were used:

a) Perl scripts

Perl scripts were developed to turns the alert, portscans and OOS files into 1 CSV (Comma Separated) files. This merger was done as a first step, to make the information more manageable. This gave us 2 VERY large CSV. We could have searched through these files with some tools, but this made data manipulation awkward. This leads us to the following point, the DTS

b) DTS (Microsoft Data Transformation Services)

In order to manipulate the data, it was decided that it needed to go in an SQL database. Since I have access to several power MS-SQL servers at work, the choice to use these was pretty obvious. Microsoft offers a tool that very easily imports and exports data to/from SQL servers. This is called Data Transformation Services (DTS). Although this tool is extremely powerful, the DTS transformation needed to send the CSV file to a database was trivial. It was just a case of running through all the steps.

1) Source: The source of the DTS was defined as the CSV
2) Target: The Target was defined as a table in the DB, if you had not create a table yet, one can be created at this time. We just mapped the same fields as in the Perl scripts (see appendix) and put them in the same order.
3) Transformation: Because the fields of the CSV and the target tables were mapped in the same order, we did not even need to modify defaults here.
4) Run the DTS

Repeat the steps for all of the CSVs

c) SQL

Once a everything was inside a few tables, some simple SQL queries were developed to do some data mining.

These queries were based on some basic building blocks of what to look for: A very large number of queries were run to analyze the data, but most were a combination (subset, ordering, grouping) of the following factors:

-Total Count
-Counts per source IP
-Count per Target IP
-Counts per alert
-Count per source port

-Count per destination port
-# of sources
-# of targets
-Is it internal of external
-Number of distinct ports

After these queries had been run to find some interesting activity, some more standard queries were run to find more specific activity

-Past activity from attacker
-Past activity from target
-Past activity towards target

d) OOS files:

Another Perl Script was used to generate a report about OOS files. The Counts of source IP, destination IP and Various packet combination. The alerts were not put into a DB as the output was not as important as seeing everything that was going on. The Perl script keeps tracks of totals and outputs a summary at the end.

Works Cited

Part 3: Analyze this

(Most of these links were included directly in the text)

GCIA practicals referenced in part 3

http://www.giac.org/practical/GCIA/Sylvain_Randier_GCIA.pdf
www.whitehats.ca/main/members/Herc_Man/Files/Al_Williams_GCIAPractical.pdf
http://www.giac.org/practical/GCIA/Andrew_Jones_GCIA.pdf
http://www.giac.org/practical/GCIA/Bruce_Auburn_GCIA.pdf

Correllations:

Port 524: compromised machine with ndsd. Jans Hector.Oct 2000
http://archives.neohapsis.com/archives/incidents/2000-10/0221.html

How do I get NetWare 6 Web Services to work?.Tek-tip website
http://www.tek-tips.com/gfaqs.cfm/lev2/3/lev3/19/pid/871/fid/3352
I
ECN,Phil Wood, July 2002. Snort Mailing list
http://archives.neohapsis.com/archives/snort/2002-07/0617.html

massive lpr exploit attempt. Pavel Lozhkin.July2001.Incidents mailing list
http://lists.insecure.org/lists/incidents/2001/Jun/0240.html

Incident Note IN-2001-01.CERT.January 2001
http://www.cert.org/incident_notes/IN-2001-01.html

SHELLCODE x86 NOOP. Kevein Butters. April 2002Focus IDS list
http://www.derkeiler.com/Mailing-Lists/securityfocus/focus-ids/2002-04/0041.html

Signature References

http://www.digitaltrust.it/arachnids/IDS339/event.html
http://www.snort.org/snort-db/sid.html?sid=533

Whois lookup

www.arin.net

Appendix A

Credits, help and acknowledgment

In addition to the other GCIA practicals and sources already credited, the following people helped me in one way or another in doing this practical

Xavier Guilbault for debugging and helping with some Per.l

Tom Chmielarski for general help and using his Databases.

Richard Noel, Philippe Lafontaine, Mitchell Choiniere and Daeman Stewart for proofreading and suggestion.

Appendix B :Perl Scripts

Perl Script for transferring ALERTS to a CSV file and portscans to CSVs
Bit and pieces from the GCIA practicals cited above were used but the whole thing ended up re-written.
The regular expression was taken directly from Al Williams' practical

```
#!/usr/bin/perl
# Convert alert.DDDDDD files to csv
#my $file = shift;
my $dir = shift;
my $mask = shift;
opendir(DIR,$dir) or die " Could not open dir : $!";
my @files = readdir DIR;
closedir DIR;
my $file;
foreach $file (@files) {
    if($file =~ /$mask/) {
            print STDERR "Parsing $file\n";


            open(FILE,"<",$file);
            while(<FILE>) {
                    next unless m/^\d/;
                    next if m/spp_portscan/;
                    chomp;
                    ($date_time,$alert,$addrs) = split(/\s+\Q[**]\E\s+/);
                    ($source, $dest) = ($addrs =~ m/(.*)\s+->\s+(.*)/);
                    ($date,$time) = split(/-/,$date_time);
                    ($source_ip, $source_port) = split(/:/,$source);
                    ($dest_ip, $dest_port) = split(/:/,$dest);
                    print
"$date,$time,$alert,$source_ip,$source_port,$dest_ip,$dest_port\n";
            }

            close(FILE);
    }
}
```

2nd script

```
#!/usr/bin/perl
# Convert scans.DDDDDD files to csv
my %months = ("Mar" => 3,
              "Oct" => 10);
my $dir = shift;
my $mask = shift;
opendir(DIR,$dir) or die " Could not open dir : $!";
my @files = readdir DIR;
closedir DIR;
my $file;
foreach $file (@files) {
    if($file =~ /$mask/) {
              print STDERR "Parsing $file\n";
              open(FILE,"<",$file);
              while(<FILE>) {
                        next unless m/^[A-Z]/;
                        chomp;
                        ($month,$day,$time,$source,$dir,$dest,$proto,$flags) = split;
                        $month = $months{$month};
                        $date = sprintf("%02d/%02d", $month, $day);
                        ($src_ip,$src_port) = split(/:/,$source);
                        ($dst_ip,$dst_port) = split(/:/,$dest);
                        print
"$date,$time,$src_ip,$src_port,$dst_ip,$dst_port,$proto,$flags\n";
              }
    }
}
```

3rd script to analyze OOS files

```
#!/usr/bin/perl

use strict;
my $dir = shift;
my $mask = shift;
if($dir eq "") {
    print "usage: $0 dir mask\n";
    print "if you want it in a file... well do:\n";
    print "$0 . OOS_ > file_to_write \n";
    print "TADAM!\n";
```

```
    exit(0);
}

opendir(DIR,$dir) or die "INVALID DIR !!!";
my @allfiles = readdir DIR;
closedir DIR;
my $file;
my %records;
$records{"total_packets"} = 0;
my %sources;
my %targets;
my %targetsNports;
my %sourcesNports;
my %source_ports;
my %target_ports;
my %flags;

my $line = 2;
my ($date,$source_ip,$source_port,$dest_ip,$dest_port);
my ($date_time,$alert,$tmp,$addrs,$source,$dest,$time,$month,$day);
my ($proto,$ttl,$tos,$id,$iplen,$dmglen,$frag);
my ($flags,$seq,$ack,$win,$tcplen);
my ($tcpopt);

foreach $file (@allfiles) {
    if($file =~ /$mask/) {
            print STDERR "Reading $file...";
            open(FILE,"<",$file) or die "Invalid file...";
            $line = 2;

            while(<FILE>) {
                    if(/=\+=\+=\+=\+=\+=\+/) { $line = 0; }
                    elsif($line >= 2) {
                            if($line == 2) {
                                    ($date_time,$source,$tmp,$dest) = split(/\s+/);
                                    ($date,$time) = split(/-/,$date_time);
                                    ($month,$day) = split(/\//,$date);
                                    ($source_ip, $source_port) = split(/:/,$source);
                                    ($dest_ip, $dest_port) = split(/:/,$dest);
    #                               print "2003-$month-$day
$time,$source_ip,$source_port,$dest_ip,$dest_port\n";
                            } elsif ($line == 3) {
                                    ($proto,$ttl,$tos,$id,$iplen,$dmglen,$frag) =
split(/\s+/);
                                    my @ttl = split(/:/,$ttl);
                                    $ttl = $ttl[1];
```

```
                                        my @tos = split(/:/,$tos);
                                        $tos = $tos[1];
                                        my @id = split(/:/,$id);
                                        $id = $id[1];
                                        my @iplen = split(/:/,$iplen);
                                        $iplen = $iplen[1];
                                        #print "$proto $ttl $tos $id $iplen $dmglen
$frag\n";
                                } elsif ($line == 4) {
                                        my @tmp = split(/\s+/);
                                        $flags = $tmp[0];
                                        $seq = $tmp[2];
                                        $ack = $tmp[4];
                                        $win = $tmp[6];
                                        $tcplen = $tmp[8];
                                        #if($flags eq "") {
                                        #       print STDERR "$line at $. and
".$records{"total_packets"}."\n";
                                        #}
                                        #print "$flags $seq $ack $win $tcplen\n";
                                } elsif ($line == 5) {
                                        if(/\d+/) { $tcpopt = 1; } else { $tcpopt =0;}
                                }
                        }
                        if($line == 0) {

                                # On a fini de parser le packet, alors on update les
records,
                                # tu n'as qua ajouter un field dans le hash table et
suivre la
                                # meme methode que pour les autres stats et tu peux
avoir ce
                                # que tu veux.

                                if($date eq "") { print STDERR "ERR date at $.\n";}
                                if($dest_ip eq "") { print STDERR "ERR dest ip at
$.\n"; }
                                if($source_ip eq "") { print STDERR "ERR source ip at
$.\n"; }
                                if($flags eq "") { print STDERR "ERR flags at $.\n"; }

                                #,$source_ip,$source_port,$dest_ip,$dest_port);
#               my
($date_time,$alert,$tmp,$addrs,$source,$dest,$time,$month,$day);
#               my ($proto,$ttl,$tos,$id,$iplen,$dmglen,$frag);
#               my ($flags,$seq,$ack,$win,$tcplen);
```

```
#               my ($tcpopt);
#               my %targetsNports;
#my %sourcesNports;

                                my $rec_key = "Source : $source_ip";

                                if( $sources{$rec_key} == undef ) {
                                        $sources{$rec_key}=1;
                                } else {
                                        $sources{$rec_key}++;
                                }


                                $rec_key = "Source with dst port :
$source_ip\:$dest_port";

                                if( $sourcesNports{$rec_key} == undef ) {
                                        $sourcesNports{$rec_key}=1;
                                } else {
                                        $sourcesNports{$rec_key}++;
                                }

                                $rec_key = "Dest with dst port : $dest_ip\:$dest_port";

                                if( $targetsNports{$rec_key} == undef ) {
                                        $targetsNports{$rec_key}=1;
                                } else {
                                        $targetsNports{$rec_key}++;
                                }

                                $rec_key = "Dest : $dest_ip";

                                if( $targets{$rec_key} == undef ) {
                                        $targets{$rec_key}=1;
                                } else {
                                        $targets{$rec_key}++;
                                }

                                $rec_key = "Dest port : $dest_port";

                                if( $target_ports{$rec_key} == undef ) {
                                        $target_ports{$rec_key}=1;
                                } else {
                                        $target_ports{$rec_key}++;
                                }
```

```
                        $rec_key = "Source port : $source_port";

                        if( $source_ports{$rec_key} == undef ) {
                                $source_ports{$rec_key}=1;
                        } else {
                                $source_ports{$rec_key}++;
                        }



                        $rec_key = "flags: $flags";

                        if( $flags{$rec_key} == undef ) {
                                $flags{$rec_key}=1;
                        } else {
                                $flags{$rec_key}++;
                        }

                        if($tcpopt) {
                                if ($records{"tcpopt"} == undef ) {
                                        $records{"tcpopt"} = 1;
                                } else {
                                        $records{"tcpopt"}++;
                                }
                        } else {
                                if ($records{"no_tcpopt"} == undef) {
                                        $records{"no_tcpopt"} = 1;
                                } else {
                                        $records{"no_tcpopt"}++;
                                }
                        }
                        $records{"total_packets"} ++;
                }
    #       print "2003-$month-$day
$time,$alert,$source_ip,$source_port,$dest_ip,$dest_port\n"              ;

                $line++;
        }
        print STDERR "Done\n";
        close(FILE);
    }

}

# On print le rapport pour le plus grand plaisir de nous tous.
print "\nReport \n";
```

```
my @keys;
my $key;

@keys = sort {
        $sources{$b} <=> $sources{$a}
} keys %sources;

print "\nSOURCES \n";
foreach $key ( @keys ) {
    print "$key : $sources{$key}\n";
}

print "*" x 50;

@keys = sort {
        $targets{$b} <=> $targets{$a}
} keys %targets;

print "\nTARGETS \n";
foreach $key ( @keys ) {
    print "$key : $targets{$key}\n";
}

print "*" x 50;

@keys = sort {
        $source_ports{$b} <=> $source_ports{$a}
} keys %source_ports;

print "\nSOURCE PORTS \n";
foreach $key ( @keys ) {
    print "$key : $source_ports{$key}\n";
}

print "*" x 50;

@keys = sort {
        $target_ports{$b} <=> $target_ports{$a}
} keys %target_ports;

print "\nTARGET PORTS \n";
foreach $key ( @keys ) {
    print "$key : $target_ports{$key}\n";
}

print "*" x 50;
```

```
@keys = sort {
        $targetsNports{$b} <=> $targetsNports{$a}
} keys %targetsNports;

print "\nTARGETS WITH target PORTS \n";
foreach $key ( @keys ) {
    print "$key : $targetsNports{$key}\n";
}

print "*" x 50;

@keys = sort {
        $sourcesNports{$b} <=> $sourcesNports{$a}
} keys %sourcesNports;

print "\n SOURCE WITH TARGET PORTS \n";
foreach $key ( @keys ) {
    print "$key : $sourcesNports{$key}\n";
}

print "*" x 50;

@keys = sort {
        $flags{$b} <=> $flags{$a}
} keys %flags;


print "\nFLAGS \n";
foreach $key ( @keys ) {
    print "$key : $flags{$key}\n";
}

print "*" x 50;

print "REPORT\n";
foreach my $key ( sort keys %records ) {
    print "$key : $records{$key}\n";
}
print "\n";
```

Appendix C
OOS packet summary
(This is only a sample as the full report is over 300 pages long)

SOURCES
Source : 217.174.98.145 : 1142
Source : 195.111.1.93 : 1130
Source : 212.16.0.33 : 1038
Source : 158.196.149.61 : 973
Source : 194.67.62.194 : 792
Source : 82.82.64.209 : 685
Source : 213.23.46.99 : 682
Source : 195.208.238.143 : 472
Source : 195.14.47.202 : 454
Source : 200.77.250.50 : 437

TARGETS
Dest : MY.NET.111.52 : 7867
Dest : MY.NET.12.6 : 4114
Dest : MY.NET.100.165 : 1672
Dest : MY.NET.69.181 : 1504
Dest : MY.NET.24.44 : 1407
Dest : MY.NET.75.240 : 839
Dest : MY.NET.84.143 : 734
Dest : MY.NET.24.34 : 471
Dest : MY.NET.100.230 : 327
Dest : MY.NET.6.7 : 282
Dest : MY.NET.12.4 : 260
Dest : MY.NET.112.159 : 243
Dest : MY.NET.60.38 : 219
Dest : MY.NET.112.152 : 184
Dest : MY.NET.60.39 : 168
Dest : MY.NET.60.16 : 107
Dest : MY.NET.29.66 : 106
Dest : MY.NET.150.133 : 82
Dest : MY.NET.84.198 : 79
Dest : MY.NET.111.61 : 60
Dest : MY.NET.99.38 : 53

SOURCE PORTS
Source port : 80 : 40
Source port : 20 : 23
Source port : 14976 : 23
Source port : 3931 : 15
Source port : 25 : 14
Source port : 47153 : 13
Source port : 52627 : 12

TARGET PORTS
Dest port : 25 : 13446
Dest port : 80 : 4194
Dest port : 8887 : 1489
Dest port : 4662 : 1255
Dest port : 113 : 406
Dest port : 110 : 246
Dest port : 1214 : 90
Dest port : 6881 : 56
Dest port : 6883 : 41
Dest port : 3264 : 26
Dest port : 443 : 26
Dest port : 22 : 26
Dest port : 21 : 21
Dest port : 18753 : 20

TARGETS WITH target PORTS
Dest with dst port : MY.NET.111.52:25 : 7867
Dest with dst port : MY.NET.12.6:25 : 4114
Dest with dst port : MY.NET.100.165:80 : 1672
Dest with dst port : MY.NET.69.181:8887 : 1489
Dest with dst port : MY.NET.24.44:80 : 1405
Dest with dst port : MY.NET.75.240:25 : 839
Dest with dst port : MY.NET.84.143:4662 : 727
Dest with dst port : MY.NET.24.34:80 : 457
Dest with dst port : MY.NET.6.7:80 : 281
Dest with dst port : MY.NET.100.230:113 : 258
Dest with dst port : MY.NET.12.4:110 : 246
Dest with dst port : MY.NET.112.159:4662 : 243
Dest with dst port : MY.NET.60.38:25 : 213
Dest with dst port : MY.NET.112.152:4662 : 184
Dest with dst port : MY.NET.60.39:25 : 159
Dest with dst port : MY.NET.29.66:80 : 106
Dest with dst port : MY.NET.60.16:25 : 99

Dest with dst port : MY.NET.150.133:1214 : 81
Dest with dst port : MY.NET.84.198:4662 : 78
Dest with dst port : MY.NET.100.230:25 : 69
Dest with dst port : MY.NET.99.38:6881 : 53
Dest with dst port : MY.NET.24.35:80 : 51
Dest with dst port : MY.NET.60.14:80 : 45
Dest with dst port : MY.NET.29.3:80 : 33
Dest with dst port : MY.NET.75.3:25 : 28
Dest with dst port : MY.NET.83.109:3264 : 26
Dest with dst port : MY.NET.12.7:443 : 23
Dest with dst port : MY.NET.84.180:6883 : 23
Dest with dst port : MY.NET.60.17:25 : 23
Dest with dst port : MY.NET.111.140:80 : 22
Dest with dst port : MY.NET.25.67:113 : 22

SOURCE WITH TARGET PORTS
Source with dst port : 217.174.98.145:25 : 1142
Source with dst port : 195.111.1.93:80 : 1130
Source with dst port : 212.16.0.33:25 : 1038
Source with dst port : 158.196.149.61:25 : 973
Source with dst port : 194.67.62.194:25 : 792
Source with dst port : 82.82.64.209:8887 : 685
Source with dst port : 213.23.46.99:8887 : 682
Source with dst port : 195.208.238.143:25 : 472
Source with dst port : 195.14.47.202:25 : 454
Source with dst port : 200.77.250.50:25 : 437
Source with dst port : 62.29.135.2:25 : 431
Source with dst port : 66.225.198.20:25 : 406
Source with dst port : 216.220.105.4:80 : 72
Source with dst port : 35.8.2.57:113 : 70
Source with dst port : 213.23.48.69:8887 : 70
Source with dst port : 204.92.128.11:25 : 68
Source with dst port : 62.121.89.3:80 : 67
Source with dst port : 81.21.202.98:80 : 67
Source with dst port : 204.92.158.14:25 : 65
Source with dst port : 204.92.158.11:25 : 62
Source with dst port : 193.219.49.20:25 : 57
Source with dst port : 200.208.2.21:25 : 55
Source with dst port : 69.50.131.181:25 : 53
Source with dst port : 216.95.201.30:25 : 50
Source with dst port : 193.219.1.48:6881 : 50
Source with dst port : 199.184.165.135:25 : 49
Source with dst port : 193.252.22.27:25 : 48
Source with dst port : 195.19.254.35:25 : 48
Source with dst port : 81.57.41.9:25 : 47
Source with dst port : 204.60.93.234:80 : 46

Source with dst port : 66.48.78.13:25 : 46
Source with dst port : 204.92.158.12:25 : 45
Source with dst port : 216.95.201.27:25 : 44
Source with dst port : 192.115.133.133:4662 : 41
Source with dst port : 216.190.181.192:80 : 39
Source with dst port : 193.41.64.2:80 : 38
Source with dst port : 80.202.102.223:80 : 37
Source with dst port : 200.21.87.165:80 : 35
Source with dst port : 80.77.40.62:25 : 34
Source with dst port : 81.31.166.251:80 : 34
Source with dst port : 207.6.138.228:80 : 34
: 195.71.9.198:18753 : 20

****
FLAGS
flags: 12****S* : 21254
flags: ******** : 296
flags: ****P*** : 122
flags: 12***R** : 37
flags: 12*A*R** : 15
flags: **U*P*SF : 7
flags: ***A**SF : 7
flags: ***AP*SF : 4
flags: *2UAPRSF : 3
flags: 1**AP*SF : 2
flags: 12*AP**F : 2
flags: 1**A**SF : 2
flags: 12**PR** : 2
flags: *2U***SF : 2
flags: 12U*P*** : 2
flags: *2***RSF : 2
flags: 12*AP*S* : 2
flags: 12UA*RS* : 2
flags: 12***R*F : 2
flags: 12U*P**F : 2
flags: *2*APRSF : 2
flags: 12UAPR*F : 2
flags: **U***** : 1
flags: 12*A**** : 1
flags: **UA**SF : 1
flags: 12UAPRSF : 1
flags: 12****** : 1
flags: 12**PRS* : 1
flags: 1**A*RSF : 1
flags: *2*A*RSF : 1

flags: *2UAP*SF : 1
flags: **UAPRSF : 1
flags: 12UA*R** : 1
flags: 12UAPRS* : 1
flags: 12*****F : 1
flags: 1**APRSF : 1
flags: 1*****SF : 1
flags: 12*A*RS* : 1
flags: 12UA*RSF : 1
flags: 12**PRSF : 1
flags: 12U***** : 1
flags: *2*AP*SF : 1
flags: *****RSF : 1
flags: *2UA*RSF : 1
flags: 1*U*PRSF : 1
flags: 12**P*SF : 1
flags: 12U***S* : 1
flags: 12**P*S* : 1
flags: *2*A**SF : 1
flags: 12UAP*** : 1
flags: 1***P*SF : 1
**********REPORT
no_tcpopt : 332
tcpopt : 21468
total_packets : 21800