Global Information Assurance Certification Paper

GCIA Practical Assignment

Cisco IDS Sensors: The Evaluation

By Michael Bernstein

Version 3.4, Option 1 Part 1
Submitted: March 14, 2004

Part 1 – Describe the State of Intrusion Detection

Evaluating Cisco Secure Intrusion Detection System (CSIDS) Sensors

Introduction

This paper is focused on evaluating Cisco intrusion detection sensors for implementing a Cisco Secure Intrusion Detection System (CSIDS). Detection sensors and management consoles comprise a real-time CSIDS. This paper's focus is solely on evaluating sensor components of the CSIDS so that readers will feel confident about designing and implementing sensors for CSIDS installations. Moreover, the Cisco Secure Policy Manager (CSPM), Cisco Secure IDS UNIX Director, and CiscoWorks VPN/Security Management Solution (VMS) management consoles will not be discussed. Properly laying out sensors is a complex task that requires full knowledge of sensor options in order to maximize leverage in network security while reducing costs through reuse of routers and firewalls to provide detection capabilities. Intrusion detection sensors are the essence of all IDSs. That is the reason this paper is so heavily focused on them. We begin by describing the six sensors. Proceeding, selection considerations will be discussed prior to actually exploring an actual implementation scenario. This will provide readers with useful knowledge to design and implement Cisco Secure Intrusion Detection Systems, which lead the market in IDS technology.

Sensor Overview

Appliance, Network Module, IDSM, Router, Firewall, and Host sensors are the six types of sensors offered by Cisco for implementation in a CSIDS. All are designed to detect, report, and terminate unauthorized activity throughout networks. All of these sensors are primarily designed for Ethernet LAN segments and the Network Module for WAN links. However, some sensors offer optional interfaces for diverse LAN technologies such as Token Ring and FDDI. The examples in this paper are all based on Ethernet since it's the most common. All sensors employ attack signature libraries as their primary mechanism to detect network misuse, which is any activity that violates a security policy. For maximum visibility, it's wise to deploy sensors on every LAN segment. Various sensors should be employed to obtain maximum security from a full range of detection capabilities of multiple devices and all inclusive network coverage. Additionally, diverse sensor types should be implemented to gain a full range or coverage, test the detection capabilities of sensors, and server the purposes of log correlation. Sensors perform real-time monitoring of network packets and report activity to management consoles for analysis. Next is a description of each of the six IDS sensors.

Appliance Sensors (4200 Series)

Cisco 4200 series appliance sensors are a steadily evolving hybrid system. There are four models currently offered: IDS-4215, IDS-4235, IDS-4250, and IDS-4250XL. Appliance sensors operate off of the proprietary IDS Sensor Software. The latest release version is 4.1. This software is extremely powerful, providing protection against known and unknown attacks. Here is a list of features and detection capabilities the Cisco IDS Sensor Software Version 4.1 offers.[28]

Detects [28]
Worms (exploits with propagation mechanisms)
Exploits
DOS type Attacks (i.e: Smurf, Trinoo, TFN, SYN Floods)
Application layer attacks
Reconnaissance Activity (i.e: ping sweeps, port scans)
Misuse (activities that violate the security policy)
Malicious P2P file sharing tools (Napster, Kazaa, iMesh) [36]

Features [28]
Stateful pattern recognition
Protocol analysis
Traffic anomaly detection
Protocol anomaly detection
Layer 2 signature engine (protects against ARP spoofing techniques)
IP fragment reassembly
TCP stream reassembly
Anti-IDS evasion protection
Deobfuscation techniques
Protocol Monitoring (IP,TCP,UDP,ICMP, etc)
Stateful decoding of application-layer protocols
(HTTP, SMTP, DNS, RPC, NetBIOS, NNTP, TELNET)
Prevents attacks from executing
Session Sniping
Shun intruders by adding ACLs to routers and modifying PIX policies
Flexible Policy Language (TAME: Threat Analysis Micro Engine)
Automatic Signature and Application Upgrades (uses staging server)

The Cisco IDS Sensor Software currently supports over 1,000 attacks signatures. A listing of all signatures by ID number and software release version can be viewed at the Cisco Secure Encyclopedia website at the following URL: http://www.cisco.com/go/csec/

Appliance sensors are primarily used to monitor Ethernet LAN and WAN subnets. A single sensor is capable of monitoring multiple subnets and handling various levels of throughput, from 80 Mbps to fully saturated gigabit links. Specifications of the four appliance sensors will be covered in the sensor selection section.[30]

Network Module Sensor

The Network Module Sensor (NM-CIDS-K9) is designed for installation in the following routers: 2600XM, 3660, or 3700 series. It utilizes the same IDS 4.1 sensor software as used in appliance sensors. The unique function of this sensor is that it's designed to monitor WAN traffic before packets cross the WAN boundary, from WAN to LAN or WAN to WAN, for example. The way it works is the router copies packets received from the WAN link, from the router back plane to the Network Module Sensor for inspection, prior to forwarding them to their destined output interfaces. This allows for more efficient inspection of WAN traffic right as it comes off the WAN when compared with appliance sensors which must wait until WAN traffic crosses the network boundary into its configured LAN/WAN Ethernet subnet. This sensor option supports T1 and T3 links.[9]

IDSM Sensor

This sensor is designed specifically for Catalyst 6500 Series Multilayer switches. The current model, IDSM-2, has up to 650 Mbps of inspection throughput for monitoring VLANs configured in switches through SPAN (Switchport Analyzer) sessions or VACLs (VLAN Access Control Lists). Traffic from multiple VLANs can be captured as long as the bandwidth doesn't exceed 650 Mbps or else the module will drop packets. The IDSM-2 utilizes the same IDS 4.1 sensor software as the appliances and network module. It is an extremely scalable solution that allows numerous modules to be stacked for aggregated inspection throughput. Reporting is streamlined since the length alarm packets have to travel is reduced to the distance between the IDSM module connection to the switch back plane and the management console interface. This protects packets from being sniffed and naturally or maliciously corrupted in transit by one half.[8]

Router Sensors

Cisco routers are designed predominantly for routing but a subset of IDS functions are available for Cisco routers in an IOS image called the Firewall Feature Set. This feature set converts a Cisco router into a router sensor. The feature set is available from Cisco's website with a CCO login for the following routers and route modules: Cisco 800, uBR900, 1400, 1600, 1700, 2500, 2600, 3600, 7100, 7200, 7400, 7500 Series, Multilayer Switch Feature Cards (MSFC,MSFC2) for Catalyst 6500 Series switches, and the Route Switch Module (RSM) for Catalyst 5000 Series switches.[4]

The signature library contains 59 signatures for the most common attacks. Signatures are grouped into four categories: Info Atomic, Info Compound, Attack Atomic, and Attack Compound. "Info" represents information gathering attempts. "Attack" represents either a DoS or illegal command execution. "Atomic"

represents simple patterns such as access to a port. "Compound" represents complex patterns that define multiple packet events for exploit detection.[7]

Signature List

1000 IP options-Bad Option List (Info, Atomic)
1001 IP options-Record Packet Route (Info, Atomic)
1002 IP options-Timestamp (Info, Atomic)
1003 IP options-Provide s,c,h,tcc (Info, Atomic)
1004 IP options-Loose Source Route (Info, Atomic)
1005 IP options-SATNET ID (Info, Atomic)
1006 IP options-Strict Source Route (Info, Atomic)
1100 IP Fragment Attack (Attack, Atomic)
1101 Unknown IP Protocol (Attack, Atomic)
1102 Impossible IP Packet (Attack, Atomic)
2000 ICMP Echo Reply (Info, Atomic)
2001 ICMP Host Unreachable (Info, Atomic)
2002 ICMP Source Quench (Info, Atomic)
2003 ICMP Redirect (Info, Atomic)
2004 ICMP Echo Request (Info, Atomic)
2005 ICMP Time Exceeded for a Datagram (Info, Atomic)
2006 ICMP Parameter Problem on Datagram (Info, Atomic)
2007 ICMP Timestamp Request (Info, Atomic)
2008 ICMP Timestamp Reply (Info, Atomic)
2009 ICMP Information Request (Info, Atomic)
2010 ICMP Information Reply (Info, Atomic)
2011 ICMP Address Mask Request (Info, Atomic)
2012 ICMP Address Mask Reply (Info, Atomic)
2150 Fragmented ICMP Traffic (Attack, Atomic)
2151 Large ICMP Traffic (Attack, Atomic)
2154 Ping of Death Attack (Attack, Atomic)
3040 TCP - no bits set in flags (Attack, Atomic)
3041 TCP - SYN and FIN bits set (Attack, Atomic)
3042 TCP - FIN bit with no ACK bit in flags (Attack, Atomic)
3050 Half-open SYN Attack/SYN Flood (Attack, Compound)
3100 Smail Attack (Attack, Compound)
3101 Sendmail Invalid Recipient (Attack, Compound)
3102 Sendmail Invalid Sender (Attack, Compound)
3103 Sendmail Reconnaissance (Attack, Compound)
3104 Archaic Sendmail Attacks (Attack, Compound)
3105 Sendmail Decode Alias (Attack, Compound)
3106 Mail Spam (Attack, Compound)
3107 Majordomo Execute Attack (Attack, Compound)
3150 FTP Remote Command Execution (Attack, Compound)
3151 FTP SYST Command Attempt (Info, Compound)
3152 FTP CWD ~root (Attack, Compound)
3153 FTP Improper Address Specified (Attack, Atomic*)
3154 FTP Improper Port Specified (Attack, Atomic*)
4050 UDP Bomb (Attack, Atomic)
4100 Tftp Passwd File (Attack, Compound)
6100 RPC Port Registration (Info, Atomic*)
6101 RPC Port Unregistration (Info, Atomic*)
6102 RPC Dump (Info, Atomic*)
6103 Proxied RPC Request (Attack, Atomic*)
6150 ypserv Portmap Request (Info, Atomic*)

6151 ypbind Portmap Request (Info, Atomic*)
6152 yppasswdd Portmap Request (Info, Atomic*)
6153 ypupdated Portmap Request (Info, Atomic*)
6154 ypxfrd Portmap Request (Info, Atomic*)
6155 mountd Portmap Request (Info, Atomic*)
6175 rexd Portmap Request (Info, Atomic*)
6180 rexd Attempt (Info, Atomic*)
6190 statd Buffer Overflow (Attack, Atomic*)
8000 FTP Retrieve Password File (Attack, Atomic*)

The CBAC (Context-based Access Control) engine of the Firewall Feature Set is responsible for enabling IDS functionality through "ip inspect" commands. When there's a signature match, a router sensor can send an alarm, drop packets, and or perform session sniping using RST packets for TCP and ICMP unreachables for UDP. Another active response mechanism is shunning which is when an external sensor commands a router to dynamically add access lists to its configuration to block hostile host attempts.[7]

Firewall Sensors

Cisco PIX firewalls, like Cisco routers, were not designed for intrusion detection. The main function of the PIX is to perform high volume packet filtering, but natively shares the same 59 attack signature library used in the IOS Firewall Feature Set.[5] The difference is that a specialized image or feature set isn't required. It's built into the PIX OS.[5] Like Cisco Routers, PIX firewalls can shun hostile hosts on command.[6] The following PIX models can be used as firewalls sensors: 501, 506/506E, 515/515E, 520, 525, and 535.[5]

Host Sensors

Cisco offers two host based sensors for desktop and server intrusion detection. They are the Host IDS (HIDS) sensor and the Cisco Security Agent. Both are designed for Windows and Solaris operating systems to provide protection against known and unknown attacks.[35; 36] Proceeding installation, both perform an automated security audit to determine what is considered normal activity for the host.[35; 36] If detected activity falls out of the range of what is considered normal, the host sensor will alarm. Both sensors can detect a plethora of attacks including buffer overflows, privilege escalation, specialized web attacks, and others.

The Cisco Security Agent was released a little over a year ago whereas the Cisco Host IDS has been around for approximately three years or so. During this time period Cisco has improved their host based intrusion detection idea and built a more comprehensive IDS solution into the Cisco Security Agent. The Cisco Security Agent works in the same fashion as the Cisco HIDS, sitting between the kernel and applications, but it is better at intercepting operating system calls and detecting unacceptable behavior. Both of these agents compete with the operating system and applications for CPU cycles.[36]

Specific intrusion detection and prevention for attacks on Microsoft IIS and Solaris Apache web server platforms are offered by both solutions. However, the Cisco HIDS provides additional protection for Solaris SPARC Planet Web Server v4.0, v4.1, and Netscape Enterprise Server v3.6.[3]

Selection Considerations

Determining factors in electing sensors for CSIDS installations are network security requirements, budget, and the decision to use existing hardware. Appliance sensors indisputably make a better network sensor than router or firewall sensors. The router and firewall sensor are the two most similar options because they use the same 59 signatures. They are low cost alternatives and supplements to CSIDS installations.

The IDS Sensor Software provides an enormous attack signature library, superior IDS instructions, is dedicated to performing ID, and has the inspection throughput to handle multiple subnets. The PIX OS and Firewall Feature Set attack signature libraries are sparse, considerable overhead is added, and inspection throughput is limited. However, if the budget prohibits the expense of an appliance sensor and data doesn't have a high classification/sensitivity, and the external threat factor is low, then companies may decide to convert existing Cisco routers and PIX firewalls into sensors. This provides limited IDS capabilities. Compare 59 attack signatures to over 1,000 in the IDS Sensor Software. A savvy hacker can circumvent a single router or firewall sensor.

Advantages of Router/Firewall sensors
convert router or PIX where it would be impractical to deploy other sensors
take advantage of existing Cisco routers and PIXs
supplement CSIDS with key positioned router senors
good for log correlation
reduces expenditures
adds value to existing assets
gain increased intrusion detection visibility
PIX and routers have shunning defenses
PIX can command routers and route modules in multilayer switches to shun hosts and networks

Disadvantages of Router/Firewall sensors
detection processing is slower for router and firewall sensors even with robust hardware
IOS is designed for routing, PIX OS designed for packet filtering (neither for IDS)
detection processing is competing with PIX security policing and VPN tunnels
detection processing is competing with IOS routing, NAT, and other processes
routers can't command other routers or PIXs to shun hostile hosts

Suggestions for implementing Router/Firewall sensors
tune configuration before adding sensor functions in order to limit additional CPU overhead
monitor CPU usage by issuing a "show proc" from privileged exec mode
tune PIX rulesets and router ACLs (remove unnecessary rules, put higher used rules first)

It is wise to supplement CSIDS installations with router and firewall sensors. Care must be taken, especially in larger networks, in the selection of routers and PIXs for the job. The main factors to consider are network topology, hardware comparison, and device load. Considerations should also include CPU speed, memory, interface speed, device throughput, running processes (i.e. NAT, ACLs, QoS, VPN tunnels, VoIP, etc.), number of connections, and overall utilization. Routers and PIX firewalls can have their memory upgraded if necessary to support IDS functions. Additionally, the location of a router or PIX is essential for consideration as an IDS sensor. This topic will be covered in the Placement and Implementation Scenario sections. For now, here is a table detailing hardware specifications to aid in the selection of a router or firewall sensor.

Cisco Router	CPU Speed	Max Memory	Max Eth Int	References
800 Series	33 MHz	48 MB	10 Mb	10
1700 Series	48 MHz	128 MB	100 Mb	11
2500 Series	20 MHz	16 MB	10 Mb	12
2600 Series	40-160 MHz	64-256 MB	1 Gb	13
3600 Series	80-225 MHz	64-256 MB	1 Gb	14; 15
7200 Series	150-700 MHz	32 MB - 1 GB	1 Gb	16; 17; 18
7500 Series	400 MHz	1 GB	1 Gb	19
MSFC (Cat6K)	200 MHz	128 MB	None	20
MSFC2 (Cat6K)	300 MHz	256 MB	None	20
RSM (Cat5K)	400 MHz	128 MB	None	21; 22

Cisco PIX	Supported Interfaces	CPU Speed	Max Memory	Max Eth Interface	Max Cleartext Throughput	References
501	1E + 4FE	133 MHz	16 MB	10 Mb out	60 Mbps	23
506E	2E	300 MHz	32 MB	10 Mb	100 Mbps	24
515E	6FE	433 MHz	64 MB	100 Mb	188 Mbps	25
525	8FE or 3Gb	600 MHz	256 MB	1 Gb	330 Mbps	26
535	10FE or 9Gb	1 GHz	1 GB	1 Gb	1.7 Gbps	27

The first requirement of determining whether a router or PIX is suited for conversion into a sensor is location. If the location is suited, implement “ip inspect” commands on the PIX or download the Firewall Feature Set image for the router and configure. Check utilization early on after deployment and upgrade memory as needed. Refer to the above table for maximum memory specifications. Additional throughput can be gained for devices supporting higher throughput interfaces, for example upgrading a PIX 525 with FE interface to gigabit interfaces is highly recommended. Check the table above for devices that support gigabit. PIX 525 and 535 can probably benefit from incorporating IDS as well as 7000 series routers. These devices have robust hardware and support gigabit interfaces. On the other hand, it may be a good idea to incorporate PIX firewalls as sensors over routers to combine access control and IDS into a single device, since these functions usually go together. Criteria to follow when deciding whether a router or PIX should be converted into a sensor is available/maximum supported interface type/speed, CPU speed, and available/maximum supported memory. All of these factor into the inspection throughput gained from the device. Without adequate inspection throughput, false negatives occur, operational efficiency suffers, and the worst that can happen is that devices are put over their operational limits. Inspection

throughput is a variable factor based on all of the running processes on the devices, configurations (light or heavy, optimized or non-optimized), since bandwidth will be shared across interfaces with routing and filtering.

The following table is a list of the higher end sensors including the appliances, network module, and IDSM that operate off of the Cisco IDS Sensor Software with their inspection throughput and best uses.

IDS Sensor Software v4.1 devices	Inspection Throughput	# Signatures	Ideal For	References
IDS-4215	80 Mbps	1000+	Multiple T1 subnets	31
IDS-4235	250 Mbps	1000+	Multiple T3 subnets	32
IDS-4250	500 Mbps	1000+	Gb subnets & backbone switches	33
IDS-4250XL	1000 Mbps	1000+	Fully saturated gigabit subnets	34
IDSM-2	650 Mbps	1000+	Catalyst 6500 Series switches	1
Net Module NM-CIDS-K9	45 Mbps	1000+	WAN links	9

All are great choices for sensors. The only device from this list that has a downside is the network module sensor. Like the Firewall Feature Set of the Cisco router and IDS functions of the PIX OS, overhead is added. A Cisco 3660 or a 3700 series router will perform better with the network module sensor than a 2600XM. Definitely upgrade memory to 256MB on any of these three routers, which is the allowable maximum.[37]

The IDSM-2 should be introduced to CSIDS implementations that contain Catalyst 6500 Series switches. They are more efficient than all the appliance sensors except for the top of the line 4250XL. IDSM-2 blades are streamlined to reduce latency from packet travel, since they capture packets directly through the switch back plane and originate packets within the module for faster communication to the management console.[1]

Placement Considerations

Understanding sensor placement is paramount. Two rudimentary configurations will be explained. The first is sensor placement in front of a firewall and the second is sensor placement behind a firewall. Understanding these two configurations will help the reader understand how to place sensors in various network locations.

In Front of Firewall

IDS sensor placement in front of the firewall with a control interface on the protected side allows the analyst to monitor all ingress and egress traffic. However, a downside to this implementation is that the sensor cannot detect attacks originating from the inside and targeting the internal network because the packets don't pass through the firewall.[2]

Behind Firewall

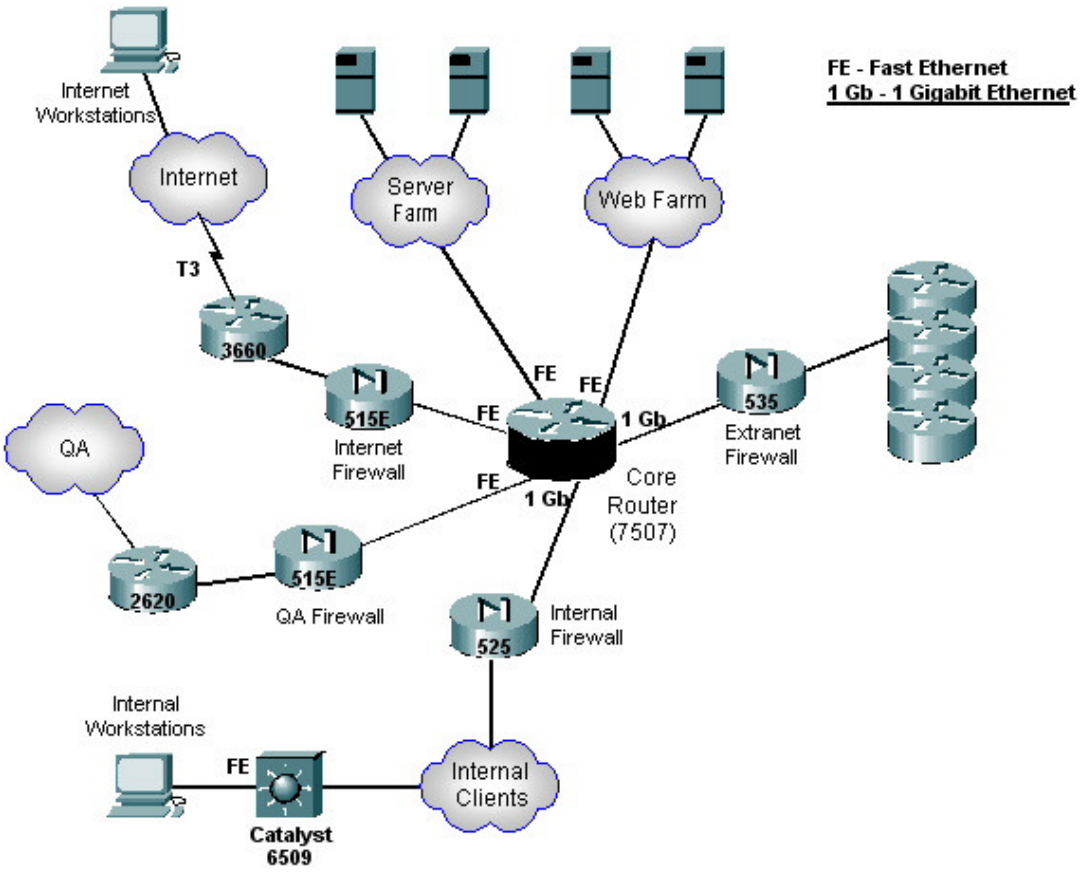
In this configuration, the sensor can detect internal attacks unlike the previous configuration. However, the sensor will no longer be able to detect ingress attacks that the firewall is filtering.[2]

The point of this section is to inform the reader of the necessity of having multiple sensors. Without multiple sensors, networks will be blind to attacks in segments that don't have sensors. The examples above demonstrate a single sensor implementation in a network with two segments, an internal and external side. Most networks, from small to enterprise scale, will have multiple segments and the engineers responsible for implementation will need to take into consideration the visibility gained and lost from implementing sensors in given regions.

Implementation Scenario

Implementing sensors is a difficult task. Placing a sensor in one specific area could blind sight another area. Multiple sensors are required for full network coverage. The goal is full detection visibility and minimal expenditures on intrusion detection hardware and software. PIXs and Cisco routers can easily be converted to sensors to save money and provide intrusion detection in hard to

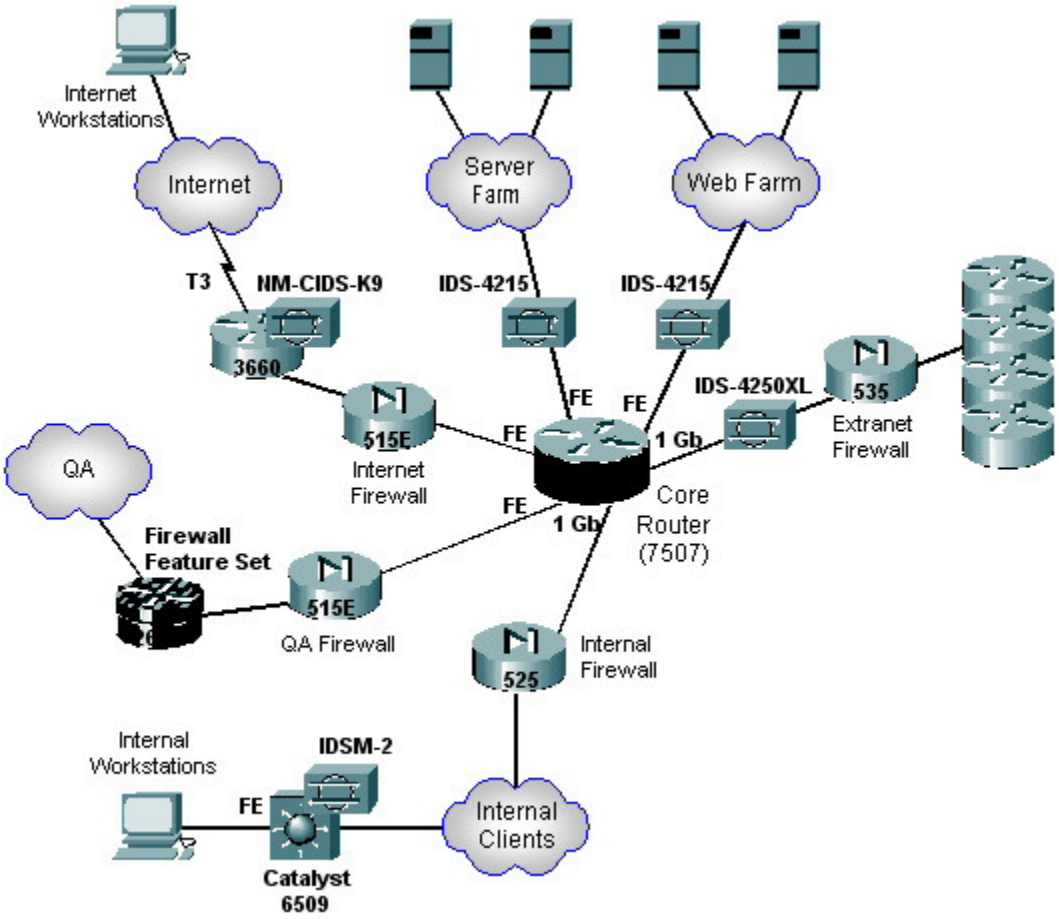
reach areas. Take a look at the following network diagram and think about the most viable sensor implementation considering topology and existing hardware.



The above network requires a robust CSIDS implementation. They would like to use existing hardware for IDS as necessary without compromising security. To understand the network topology and resources, begin by analyzing the network structure. There is a T3 connecting this organization to the Internet. A Cisco 3360 is the border router connected to a PIX 515E. There are two DMZ subnets, respectively Server Farm and Web Farm. Internet workstations, internal workstations, QA, and Extranet clients use resources in these subnets. All traffic flow is through the perspective of the core router, a Cisco 7507. There's a PIX 515E and Cisco 2620 router connecting a QA network, and a PIX 535 connecting Extranets clients and servers. A PIX 525 filters access to and from internal clients.

Since the requirement includes the use of existing equipment, it makes sense to deploy an IDSM-2 blade in the Catalyst 6509 and a NM-CIDS-K9 in the 3660 border router. QA, on the other hand, can most likely suffice with their Cisco 2620 router upgraded to the Firewall Feature Set image. Appliance sensors will do justice for the three remaining regions. Since there are multiple extranets to business partners, it makes sense to deploy a robust appliance sensor because many outside users and application sensors will be connecting internally. The

segment supports gigabit bandwidth so we'll use an IDS-4250XL for scalability. The two remaining subnets, Server and Web farms, can be secured with a lower-end appliance sensor, specifically the IDS-4215. Bandwidth should be measured on these two segments, and if 80 Mbps is exceeded, the sensor should be upgraded to an IDS-4235. Below is the finished layout with the implementation of the described sensors.



Conclusion

This paper has discussed Cisco's sensor security product lines, the most significant components of the Cisco Secure Intrusion Detection System (CSIDS). Implementation is subjective to the topology of the network, security requirements, and whether existing Cisco routers and switches are to be employed as sensors. If care isn't taken when designing a CSIDS, areas will go overlooked and can lead to possible network and system compromise. If an IDS can't detect attacks, then it is useless. If routers and PIX firewalls are to be employed as sensors, attempt to reduce overhead by streamlining configurations. Remove unnecessary commands and services. Upgrade devices with faster interfaces and add additional memory. Additionally, if the

network has a Catalyst 6500 Series switches, greatly consider implementing the IDSM-2 blade for seamless switching and intrusion detection, eliminating delay. Use this paper as a resource for designing and implementing CSIDS sensors.

References

[1] Cisco Systems. "Catalyst 6500 Series Intrusion Detection System Module Installation and Configuration Note Version 3.0(5)." 2002. URL: http://www.cisco.com/univercd/cc/td/doc/product/iaabu/csids/idsm/idsm_2/13074_04.htm

[2] Cisco Systems. "Cisco Intrusion Detection System Sensor Installation and Safety Note." 2002. URL: http://www.cisco.com/univercd/cc/td/doc/product/iaabu/csids/sensor/7016_04.htm

[3] Cisco Systems. "Data Sheet: Cisco IDS Host Sensor." 2002. URL: http://www.cisco.com/warp/public/cc/pd/sqsw/sqidsz/prodlit/hid25_ds.htm

[4] Cisco Systems. "Cisco IOS Software: Firewall Feature Set." 2002. URL: http://www.cisco.com/warp/public/cc/pd/iosw/ioft/iofwft/prodlit/fire_ds.htm

[5] Cisco Secure Internet Security Solutions, Part 1. Cisco Press, 2001. URL: http://networking.earthweb.com/netsecur/article.php/10952_883401_3

[6] Cisco Systems. "IDS PIX Shunning Using Cisco IDS UNIX Director. 2004. URL:http://www.cisco.com/en/US/products/hw/vpndevc/ps2030/products_configuration_example09186a0080145270.shtml

[7] Cisco Systems. "Configuring Cisco IOS Firewall Intrusion Detection System. URL:http://www.cisco.com/en/US/products/sw/iosswrel/ps1835/products_configuration_guide_chapter09186a00800ca7c6.html

[8] Cisco Systems. "Cisco Catalyst 6500 Intrusion Detection System (IDSM-2) Services Module." URL: http://tecun.cimex.com.cu/tecun/software/Soporte_tecnico_Redes/Cisco/catalyst/idsm2_qa.pdf

[9] Cisco Systems. "Cisco Intrusion Detection System NM for Cisco 2600 & 3700 Routers: Design Implementation Guide." URL: http://www.cisco.com/en/US/netsol/ns340/ns394/ns171/ns292/networking_solutions_design_guidance09186a00801cf9fc.html

[10] Cisco Systems. "Cisco 800 Series ISDN Routers." URL: http://www.cisco.com/en/US/products/hw/routers/ps380/products_data_sheet09186a008008872c.html

[11] Cisco Systems. "Release Notes for the Cisco 1700 Series Routers for Cisco IOS Release 12.2(4)XL. June, 2002. URL:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122relnt/1700/rn1700xl.pdf

[12] Kahlon. "Upgrade Configuration for Cisco Model: 2500 Series." URL: http://www.kahlon.com/rn1120_Cisco_2500_Series.html

[13] "Cisco 2600 Series – Modular Access Router Family." Cisco Product Catalog, August 2002. URL: http://www.osws.com/pdf/cisco%202600.pdf

[14] Kahlon. "Upgrade Configuration for Cisco Model: 3600 Series Router." URL: http://www.kahlon.com/rn22686_Cisco_3600_Series_Router.html

[15] Kahlon. "Upgrade Configuration for Cisco Model: 3660 Single Port Fast Ethernet Router." URL: http://www.kahlon.com/rn23222_Cisco_3660_Single_Port_Fast_Ethernet_Router.html

[16] "Cisco 7200 Series Router Architecture." 2001. URL: http://tecun.cimex.com.cu/tecun/software/Soporte_tecnico_Redes/Cisco/7200/arch_7200_5810.pdf

[17] "Cisco Network Processing/Services Engine – NPE/NSE (7200 Series)." URL: http://www.megacomp.fi/cisco_network_processing.html

[18] Cisco Systems. "Cisco 7200 Series Router Architecture. Document ID: 5810." January 07, 2004. URL: http://www.cisco.com/en/US/products/hw/routers/ps341/products_tech_note09186a0080094ea3.shtml

[19] "Route Switch Processor 16 for the Cisco 7500 Series Router." URL: http://www.interlinkweb.com/systemics/assets/product_images/cisco/rsp16_ds.pdf

[20] Cisco Systems. "Data Sheet: Catalyst 6000 Family Multilayer Switch Feature Card MSFC2." July, 2002. URL: http://www.cisco.com/warp/public/cc/pd/si/casi/ca6000/prodlit/msfc2_ds.htm

[21] Cisco Systems. "Troubleshooting the Catalyst 5000 Route Switch Module (RSM) and InterVLAN Routing. September 25, 2002. URL: http://www.cisco.com/warp/public/473/56.html

[22] "Catalyst 5000/5500 RSM Memory." 2004. http://www.memoryx.net/memx/cat50routswi.html

[23] Cisco Systems. "Data Sheet: Cisco PIX 501 Security Appliance." URL: http://www.cisco.com/en/US/products/hw/vpndevc/ps2030/products_data_sheet09186a0080091b18.html

[24] Cisco Systems. "Data Sheet: Cisco PIX 506E Security Appliance." URL: http://www.cisco.com/en/US/products/hw/vpndevc/ps2030/products_data_sheet09186a0080091b13.html

[25] Cisco Systems. "Data Sheet: Cisco PIX 515E Security Appliance." URL: http://www.cisco.com/en/US/products/hw/vpndevc/ps2030/products_data_sheet09186a0080091b15.html

[26] Cisco Systems. "Cisco PIX 525 Security Appliance Data Sheet." URL: http://www.cisco.com/en/US/products/hw/vpndevc/ps2030/products_data_sheet09186a0080091b09.html

[27] Cisco Systems. "Data Sheet: Cisco PIX 535 Security Appliance." URL: http://www.cisco.com/en/US/products/hw/vpndevc/ps2030/products_data_sheet09186a008007d05d.html

[28] Cisco Systems. "Cisco IDS Sensor Software Version 4.0." URL: http://www.cisco.com/en/US/products/sw/secursw/ps5052/ps4966/index.html

[29] Enterprise Networks & Servers. "Cisco Offers Bundle of Security Products and Enhancements." July, 2003. URL: http://www.enterprisenetworksandservers.com/monthly/art.php/166

[30] Cisco Systems. "Cisco IDS 4200 Series Sensors: Introduction." URL: http://www.cisco.com/en/US/products/hw/vpndevc/ps4077/index.html

[31] Cisco Systems. "Cisco IDS 4215 Sensor: Introduction." URL: http://www.cisco.com/en/US/products/hw/vpndevc/ps4077/ps5367/index.html

[32] Cisco Systems. "Cisco IDS 4235 Sensor: Introduction." URL: http://www.cisco.com/en/US/products/hw/vpndevc/ps4077/ps4078/index.html

[33] Cisco Systems. "Cisco IDS 4250 Sensor: Introduction." URL: http://www.cisco.com/en/US/products/hw/vpndevc/ps4077/ps4079/index.html

[34] Cisco Systems. "Cisco IDS 4250XL Sensor: Introduction." URL: http://www.cisco.com/en/US/products/hw/vpndevc/ps4077/ps5050/index.html

[35] Cisco Systems. "Data Sheet Cisco IDS Host Sensor." Nov, 2002. URL: http://www.cisco.com/warp/public/cc/pd/sqsw/sqidsz/prodlit/hid25_ds.htm

[36] Cisco Systems. "Cisco Security Agent: Data Sheet." 2003. URL: http://www.cisco.com/application/pdf/en/us/guest/products/ps5057/c1650/cdccont_0900aecd800ade37.pdf

[37] "Cisco 3700 Series Router Product Overview." Feb, 2003. URL:

http://www.stemmer.de/datenbank/pdfs/cisco_ip_gate_3700.pdf

Part 2 – Network Detects

Network Detect 1 - UPNP Service Discover Attempts

```
[**] SCAN UPNP service discover attempt [**]
01/22-09:55:53.680371 0:80:AD:73:97:6A -> 1:0:5E:7F:FF:FA type:0x800 len:0x8B
192.168.1.100:1026 -> 239.255.255.250:1900 UDP TTL:4 TOS:0x0 ID:60 IpLen:20
DgmLen:125
Len: 105
4D 2D 53 45 41 52 43 48 20 2A 20 48 54 54 50 2F  M-SEARCH * HTTP/
31 2E 31 0D 0A 48 6F 73 74 3A 32 33 39 2E 32 35  1.1..Host:239.25
35 2E 32 35 35 2E 32 35 30 3A 31 39 30 30 0D 0A  5.255.250:1900..
53 54 3A 75 70 6E 70 3A 72 6F 6F 74 64 65 76 69  ST:upnp:rootdevi
63 65 0D 0A 4D 61 6E 3A 22 73 73 64 70 3A 64 69  ce..Man:"ssdp:di
73 63 6F 76 65 72 22 0D 0A 4D 58 3A 33 0D 0A 0D  scover"..MX:3...
0A                                               .

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] SCAN UPNP service discover attempt [**]
01/22-09:55:56.683493 0:80:AD:73:97:6A -> 1:0:5E:7F:FF:FA type:0x800 len:0x8B
192.168.1.100:1026 -> 239.255.255.250:1900 UDP TTL:4 TOS:0x0 ID:61 IpLen:20
DgmLen:125
Len: 105
4D 2D 53 45 41 52 43 48 20 2A 20 48 54 54 50 2F  M-SEARCH * HTTP/
31 2E 31 0D 0A 48 6F 73 74 3A 32 33 39 2E 32 35  1.1..Host:239.25
35 2E 32 35 35 2E 32 35 30 3A 31 39 30 30 0D 0A  5.255.250:1900..
53 54 3A 75 70 6E 70 3A 72 6F 6F 74 64 65 76 69  ST:upnp:rootdevi
63 65 0D 0A 4D 61 6E 3A 22 73 73 64 70 3A 64 69  ce..Man:"ssdp:di
73 63 6F 76 65 72 22 0D 0A 4D 58 3A 33 0D 0A 0D  scover"..MX:3...
0A                                               .

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] SCAN UPNP service discover attempt [**]
01/22-09:55:59.714135 0:80:AD:73:97:6A -> 1:0:5E:7F:FF:FA type:0x800 len:0x8B
192.168.1.100:1026 -> 239.255.255.250:1900 UDP TTL:4 TOS:0x0 ID:62 IpLen:20
DgmLen:125
Len: 105
4D 2D 53 45 41 52 43 48 20 2A 20 48 54 54 50 2F  M-SEARCH * HTTP/
31 2E 31 0D 0A 48 6F 73 74 3A 32 33 39 2E 32 35  1.1..Host:239.25
35 2E 32 35 35 2E 32 35 30 3A 31 39 30 30 0D 0A  5.255.250:1900..
53 54 3A 75 70 6E 70 3A 72 6F 6F 74 64 65 76 69  ST:upnp:rootdevi
63 65 0D 0A 4D 61 6E 3A 22 73 73 64 70 3A 64 69  ce..Man:"ssdp:di
73 63 6F 76 65 72 22 0D 0A 4D 58 3A 33 0D 0A 0D  scover"..MX:3...
0A                                               .

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```


Source of Trace:

This trace came from snort running on a LAN attached to a Comcast Cable High-Speed Network. The NAT gateway performs one to many NAT and also functions as a DHCP server. The following diagram depicts the topology.

Detect Generated By:

This detect was generated by Snort NIDS version 1.9. Snort was invoked in both sniffer and IDS mode, using the standard rule set for capture and analysis. The following command was used.

snort -dev -c /etc/snort/snort.conf –l snortout
-d dump packet payloads
-e display link layer data excluding trailer
-v verbose mode
-c specifies location of snort configuration file
-l specifies directory for snort to dump alerts

The snort rule below triggered this alert.

alert udp $EXTERNAL_NET any -> $HOME_NET 1900 (msg:"MISC UPNP malformed advertisement"; content:"NOTIFY * "; nocase; classtype:misc-attack; reference:cve,CAN-2001-0876; reference:cve,CAN-2001-0877; sid:1384; rev:2;)

The log format is as follows:

192.168.1.100:1026 -> 239.255.255.250:1900 UDP TTL:4 TOS:0x0 ID:60 IpLen:20 DgmLen:125 Len: 105

[source ip]:[source port]->[destination ip]:[destination port] [protocol=UDP] [Time To Live=4] [Type of Service 0x0 = normal] [unique datagram identifier] [ip header length=20] [datagram length=125] [payload length=105]

Probability the Source Address was Spoofed:

It's hard to tell from these packet traces whether the source IP was spoofed or not. The packets look original in terms of headers sizes and fields. The TTL 4 indicates that these packets can be routed. Also, due to the nature of the udp transport, since there is no session establishment, spoofing is more likely compared with tcp. These multicasts may be from an innocent LAN host, or may be a result of spoofing from an attacker that also resides on the same LAN segment. In order for the spoofing theory to be viable, the attacker may have used an ARP cache poisoning technique. The packet headers below look normal.

192.168.1.100:1026 -> 239.255.255.250:1900 UDP TTL:4 TOS:0x0 ID:60 IpLen:20 DgmLen:125 Len: 105

192.168.1.100:1026 -> 239.255.255.250:1900 UDP TTL:4 TOS:0x0 ID:61 IpLen:20 DgmLen:125 Len: 105
192.168.1.100:1026 -> 239.255.255.250:1900 UDP TTL:4 TOS:0x0 ID:62 IpLen:20 DgmLen:125 Len: 105

Description of the Attack:

The attack alerted on is based on a weakness in Microsoft's Universal Plug and Play (UPNP) architecture. UPNP is used in peer-to-peer networking to allow Microsoft PCs to discover and utilize network devices. This specific attack is based on a documented buffer overflow vulnerability that affects Windows 98, ME, and XP. Hosts that run unpatched UPNP are vulnerable to a buffer overflow attack that allows arbitrary code to be run. This can lead to system compromise, DoS/DDoS on the victim host, or using the compromised host in a larger DoS/DDoS on other systems.

References to this vulnerability:

CVE: CAN-2001-0876
CVE: CAN-2001-0877
CERT® Advisory CA-2001-37
CERT Vulnerability VU#951555
Bugtraq ID: 3724

Attack Mechanism:

As mentioned in the previous section, hosts running unpatched UPNP are open to multiple vulnerabilities. The SSDP (Simple Service Discovery Protocol) component of UPNP makes these vulnerabilities possible. SSDP is used in UPNP to send advertisements to notify nodes of a host's existence. It also listens on multicast and broadcast addresses. The main vulnerability is a remotely exploitable buffer overflow that allows an attacker to gain system level access to a vulnerable host. From then on, the system level breach can lead to other privilege escalation techniques, which can lead to total compromise of an entire Windows network. The system level breach is feasible using a single anonymous udp/ssdp attack session. This attack session hinges on spoofing the source address to make 98/ME/XP clients connect back to the attacker IP and pass on http/https requests, which can be used in http unicode, double decode, and random GCI exploit attacks. Additionally, malformed advertisements at various speeds will cause an overwrite violation of the return pointer, allowing arbitrary code to be run. The session below shows malicious crafting to gain unauthorized access to a host running unpatched UPNP.

```
Example Session:

NOTIFY * HTTP/1.1
HOST: 239.255.255.250:1900
CACHE-CONTROL: max-age=10
LOCATION: http://IPADDRESS:PORT/.xml
NT: urn:schemas-upnp-org:device:InternetGatewayDevice:1
NTS: ssdp:alive
```

```
SERVER: EEYE/2001 UPnP/1.0 product/1.1
USN: uuid:EEYE
```

DoS/DDoS attacks are also made possible by UPNP vulnerabilities. Flood packets addressed to udp port 1900 of a Win 98/ME/XP machine can cause a DoS condition to occur on the vulnerable host. Multiple hosts can also perform a DDoS to increase the magnitude of the attack. Also, the previous attack that compromises the host can allow the attacker to install DoS agent software on the that makes the host take part in a larger DDoS attack on a chosen host or network. Many vulnerable hosts can be DoS'd at once, since SSDP announcements are sent to broadcast and multicast addresses, the damage can be inflicted at once to multiple destinations. Also note that udp is an insecure protocol and udp attacks are hard to trace.

Correlations:

Riley Hassell from eEye Digital Security originally discovered this vulnerability. The specific eEye Advisory and alert was released on December 20, 2001 referencing Multiple Remote Windows XP/ME/98 Vulnerabilities.

The advisory, AD20011220**,** can be found at the following location:
http://www.eeye.com/html/Research/Advisories/AD20011220.html

Microsoft has acknowledged this vulnerability as an Unchecked Buffer in Universal Plug and Play can lead to System Compromise.

Microsoft released a patch on December 20, 2001 and a security bulletin, MS01-059 that follows at this location:
http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-059.asp

Some common vulnerability and exposure (CVE) entries related to this incident:

CVE: CAN-2001-0876 (buffer overflow issue)
Buffer overflow in Universal Plug and Play (UPnP) on Windows 98, 98SE, ME, and XP allows remote attackers to execute arbitrary code via a NOTIFY directive with a long Location URL.
CVE: CAN-2001-0877 (denial of service issue)
Universal Plug and Play (UPnP) on Windows 98, 98SE, ME, and XP allows remote attackers to cause a denial of service via (1) a spoofed SSDP advertisement that causes the client to connect to a service on another machine that generates a large amount of traffic (e.g., chargen), or (2) via a spoofed SSDP announcement to broadcast or multicast addresses, which could cause all UPnP clients to send traffic to a single target system.

CERT Advisories related to this incident:

CA-2001-37 Buffer Overflow in UPnP Service On Microsoft Windows
http://www.cert.org/advisories/CA-2001-37.html

VU#951555 CERT Vulnerability Note: Microsoft Windows Universal Plug and Play (UPNP) vulnerable to buffer overflow via malformed advertisement packets

http://www.kb.cert.org/vuls/id/951555

BUGTRAQ references:

BUGTRAQ:20020109 UPNP Denial of Service
BUGTRAQ:20011220 Multiple Remote Windows XP/ME/98 Vulnerabilities

Evidence of Active Targeting:

There doesn't appear to be any evidence of active targeting. The udp multicast traffic seen doesn't seem to have been spoofed, nor were the advertisements sent at various speeds. They were spaced apart at three-second intervals and fairly consistent. It appears that UPNP was simply using this transmission to discover local network connected Plug and Play devices.

Severity:

Severity = (Criticality + Lethality) – (System Countermeasures + Network Countermeasures)

Criticality – 1 This computer is an end-user workstation

Lethality – 5 This attack could ultimately result in total network compromise, DoS/DDoS on the end system, or install a DDoS agent on the end system to help facilitate a large scale DDoS attack.

System Countermeasures – 1 This system has not been patched with MS01-059 safeguarding against this vulnerability

Network Countermeasures – 3 The network is protected by a NAT Gateway providing adequate protection from internet users but is totally vulnerable to an inside attack

(1 + 5) – (1 + 3) = 2

Defensive Recommendation:

Peer-to-peer networking is insecure to begin with. Turn off UPNP if possible. The first defensive recommendation is to patch the vulnerable system with patch MS01-059 which can be downloaded from Microsoft's site at the following URL: http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-059.asp
Secondly, block port 1900 traffic ingress and egress on the packet filter/NAT gateway.

Multiple Choice Test Question:

The following trace comes from a vulnerable system.

```
01/22-09:55:53.680371 0:80:AD:73:97:6A -> 1:0:5E:7F:FF:FA type:0x800 len:0x8B
192.168.1.100:1026 -> 239.255.255.250:1900 UDP TTL:4 TOS:0x0 ID:60 IpLen:20 DgmLen:125
Len: 105

4D 2D 53 45 41 52 43 48 20 2A 20 48 54 54 50 2F      M-SEARCH * HTTP/
31 2E 31 0D 0A 48 6F 73 74 3A 32 33 39 2E 32 35      1.1..Host:239.25
35 2E 32 35 35 2E 32 35 30 3A 31 39 30 30 0D 0A      5.255.250:1900..
53 54 3A 75 70 6E 70 3A 72 6F 6F 74 64 65 76 69      ST:upnp:rootdevi
63 65 0D 0A 4D 61 6E 3A 22 73 73 64 70 3A 64 69      ce..Man:"ssdp:di
73 63 6F 76 65 72 22 0D 0A 4D 58 3A 33 0D 0A 0D      scover"..MX:3...
0A
```

What is the attack that can be afflicted on this system?

(A) SYN Flood attack
(B) LAND attack
(C) Denial of Service attack
(D) Buffer Underrun to gain system level access

Answer: C. Multiple udp/ssdp packets sent to port 1900 can cause a denial of service condition on targeted systems.

References:

Roesch, Martin. Intrusion Detection Snort Style (Track 3). The SANS Institute, 2003.
"CVE-2001-0876." Mar, 2002. URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0876
"CVE-2001-0877." Mar, 2002. URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0877
"CERT Advisory CA-2001-37 Buffer Overflow in UPnP Service On Microsoft Windows." Dec 20, 2001. URL: http://www.cert.org/advisories/CA-2001-37.html
"Vulnerability Note VU#951555 – Microsoft Windows UpnP vulnerable to buffer overflow via malformed advertisement packets." Dec 20, 2001. URL: http://www.kb.cert.org/vuls/id/951555
"Microsoft Universal Plug and Play Simple Service Discovery Protocol Denail of Service Vulnerability." Dec, 2001. URL: http://online.securityfocus.com/bid/3724
eEye Digital Security. "UPNP – Multiple Remote Windows XP/ME/98 Vulnerabilities." Dec 20, 2001. URL: http://www.eeye.com/html/Research/Advisories/AD20011220.html
Microsoft TechNet. "Microsoft Security Bulletin MS01-059." Dec 20, 2001. URL: http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-059.asp

Network Detect #2 - WEB-CGI Redirect Access

```
[**] WEB-CGI redirect access [**]
[Classification: Attempted Information Leak] [Priority: 2]
01/24-22:07:42.566173 0:0:C5:E:58:7F -> 0:20:78:CF:1E:BE type:0x800 len:0x8A
192.168.1.103:1297 -> 64.12.xxx.xxx:80 TCP TTL:128 TOS:0x0 ID:3872 IpLen:20
DgmLen:124 DF
***AP*** Seq: 0x6FEE6C4C  Ack: 0xF0607E6F  Win: 0x4470  TcpLen: 20
```

```
47 45 54 20 2F 72 65 64 69 72 65 63 74 73 2F 69  GET /redirects/i
6E 63 6C 69 65 6E 74 2F 63 6F 6E 74 65 6E 74 2E  nclient/content.
61 64 70 20 48 54 54 50 2F 31 2E 31 0D 0A 75 73  adp HTTP/1.1..us
65 72 2D 61 67 65 6E 74 3A 20 41 49 4D 0D 0A 68  er-agent: AIM..h
6F 73 74 3A 20 77 77 77 2E 61 69 6D 2E 63 6F 6D  ost: www.aim.com
0D 0A 0D 0A                                      ....

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] WEB-CGI redirect access [**]
[Classification: Attempted Information Leak] [Priority: 2]
01/24-22:09:54.571702 0:0:C5:E:58:7F -> 0:20:78:CF:1E:BE type:0x800 len:0x8A
192.168.1.103:1329 -> 64.12.xxx.xxx:80 TCP TTL:128 TOS:0x0 ID:4106 IpLen:20
DgmLen:124 DF
***AP*** Seq: 0x71F885E1  Ack: 0xF0CB705C  Win: 0x4470  TcpLen: 20
47 45 54 20 2F 72 65 64 69 72 65 63 74 73 2F 69  GET /redirects/i
6E 63 6C 69 65 6E 74 2F 63 6F 6E 74 65 6E 74 2E  nclient/content.
61 64 70 20 48 54 54 50 2F 31 2E 31 0D 0A 75 73  adp HTTP/1.1..us
65 72 2D 61 67 65 6E 74 3A 20 41 49 4D 0D 0A 68  er-agent: AIM..h
6F 73 74 3A 20 77 77 77 2E 61 69 6D 2E 63 6F 6D  ost: www.aim.com
0D 0A 0D 0A                                      ....

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] WEB-CGI redirect access [**]
[Classification: Attempted Information Leak] [Priority: 2]
01/24-22:58:22.954983 0:0:C5:E:58:7F -> 0:20:78:CF:1E:BE type:0x800 len:0x8A
192.168.1.103:1512 -> 64.12.xxx.xxx:80 TCP TTL:128 TOS:0x0 ID:7478 IpLen:20
DgmLen:124 DF
***AP*** Seq: 0x9DBF9294  Ack: 0x131AE78A  Win: 0x4470  TcpLen: 20
47 45 54 20 2F 72 65 64 69 72 65 63 74 73 2F 69  GET /redirects/i
6E 63 6C 69 65 6E 74 2F 63 6F 6E 74 65 6E 74 2E  nclient/content.
61 64 70 20 48 54 54 50 2F 31 2E 31 0D 0A 75 73  adp HTTP/1.1..us
65 72 2D 61 67 65 6E 74 3A 20 41 49 4D 0D 0A 68  er-agent: AIM..h
6F 73 74 3A 20 77 77 77 2E 61 69 6D 2E 63 6F 6D  ost: www.aim.com
0D 0A 0D 0A                                      ....

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Source of Trace:

This trace came from snort running on a LAN attached to a Comcast Cable High-Speed Network. The NAT gateway performs one to many NAT and also functions as a DHCP server. The following diagram depicts the topology.

Detect Generated By:

This detect was generated by Snort NIDS version 1.9. Snort was invoked in both sniffer and IDS mode, using the standard rule set for capture and analysis. The following command was used.

snort -dev -c /etc/snort/snort.conf –l snortout2
-d dump packet payloads
-e display link layer data excluding trailer
-v verbose mode
-c specifies location of snort configuration file
-l specifies directory for snort to dump alerts

The following snort rule triggered the alert under SID 895.

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-CGI redirect access";flow:to_server,established; uricontent:"/redirect"; nocase;reference:bugtraq,1179; reference:cve,CVE-2000-0382; classtype:attempted-recon; sid:895; rev:5;)

The log format is as follows:

192.168.1.103:1329 -> 64.12.xxx.xxx:80 TCP TTL:128 TOS:0x0 ID:4106 IpLen:20 DgmLen:124 DF Seq:0x71F885E1 Ack:0xF0CB705C Win:0x4470 TcpLen:20

Source Address: 192.168.1.103
Source Port: 1329
Destination Address: 64.12.xxx.xxx
Destination Port: 80
Embedded Protocol: TCP
Time To Live (remaining lifetime of packet):128 hops
Type of Service: 0x0 is the default (used as a priority field)
Packet Identifier ID: 4106 (unique number identifies the packet)
IP Length: 20 bytes is the standard IP header
Datagram Length: 124 bytes
DF: Don't Fragment bit (means that the packet will not be fragmented)
Sequence: hex digit for communicating bytes sent - 0x71F885E1
Acknowledgement: hex digit for response to a sequence - 0xF0CB705C
Window: 0x4470 specifies the number of bytes a sender can transmit without receiving an acknowledgement
TCP Length: 20 bytes is the standard length of a TCP header

Probability the Source Address was Spoofed:

The source address is unlikely spoofed since sequence numbers and acknowledgements are present. The system is taking part in an active TCP session that appears to already have been established with a 3-way handshake. Had this been a possibility of spoofing, it would be expected to see a lone initial sequence number (ISN) or the next sequence number of an active session would need to be predicted which is extremely unlikely. However, there are acknowledgements being passed here. Spoofing is likely to occur with absence of acknowledgements and absence of completion of a 3-way handshake, like spoofing performed in SYN or ACK scans for example. These packets have the PUSH flag set indicating that data is being transmitted as part of an active session.

Description of the Attack:

This attack affects a known vulnerability in Macromedia ColdFusion ClusterCATS. ClusterCATS is a web server clustering technology that provides load balancing and failover services to ensure high availability for web servers. An attack on a vulnerable ClusterCATS web server will release confidential query string information such as usernames and passwords on a redirect, leaking valuable information that can lead to a system compromise. Here are references to this vulnerability.

Allaire Security Bulletin: ASB00-12
CVE: CVE-2000-0382
Bugtraq ID: 1179

Attack Mechanism:

ColdFusion ClusterCATS is a web server. The web server processes client requests on tcp port 80. Further, http redirections can take place. During an http redirection, the ClusterCATS can potentially leak sensitive information to the redirected site. The problem occurs when a query string, passed on in redirection to the web server, includes confidential information such as usernames and passwords, resulting in total system compromise. If the web server contains useful information in the system itself, or ODBC connection privileges to a sensitive database, an attacker would have heightened interested in exploiting a ClusterCATS web server.

Correlations:

This attack didn't turn up any responses on DShield's incident website. However, other reports confirm this vulnerability.

Allaire Knowledge Base Article (15607)
http://www.allaire.com/Handlers/index.cfm?ID=15607&Method=Full

Allaire Security Bulletin (ASB00-12)
http://packetstormsecurity.nl/advisories/allaire/asb00-12.querystring

CVE-2000-0382
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2000-0382

Bugtraq ID: 1179
http://online.securityfocus.com/bid/1179

Evidence of Active Targeting:

This trace does not appear to be active targeting. What's happening here is that an http redirect is taking place between client 192.168.1.103 on port 1297 to a ColdFusion ClusterCATS web server running at 64.12.xx.xx on TCP port 80.

The query string being passed in this redirect is:

GET /redirects/inclient/content.adp HTTP/1.1.

The query string passed here shows no indication of confidential information such as usernames and passwords being passed to this web server. Therefore, this site probably wasn't targeted and the alert is a result of a false positive.

Severity:

Severity = (Criticality + Lethality) – (System Countermeasures + Network Countermeasures)

Criticality – 5 This is a remote production web server requiring high availability that a local LAN host is communicating with

Lethality – 5 This attack would result in total system compromise and compromise of other systems if network logon usernames and passwords were passed in a query string on redirection

System Countermeasures – 3 It is completely unknown whether the remote web server has been patched, safeguarding against this

Network Countermeasures – 2 The server may be protected by a packet filter or stateful firewall. However, port 80 traffic is allowed permitting this attack.

(5 + 5) – (3 + 2) = 5

Defensive Recommendations:

Allaire has released a patch that will resolve this vulnerability issue.

See the following URL:
http://www.macromedia.com/v1/handlers/index.cfm?ID=15697&Method=Full

Download - ClusterCATS ColdFusion Stale Query String During Redirect Patch
ftp://ftp.allaire.com/outgoing/clustercats/teserver.dll

Versions of ColdFusion need to be updated to 4.5.1 SP1 or later for the patch to be successful. Version updates are available at the Allaire website at the following URL:
http://commerce.allaire.com/download/index.cfm

Update the Snort rule that triggers this alert to be more specific. If possible, configure Snort for session sniping if this rule alerts on activity that has a high probably of being a positive. Note that session sniping will cause problems if it's disconnecting legitimate sessions.

Multiple Choice Test Question:

An http redirect can leak confidential information such as usernames and passwords on which of the following web server platforms?

(A) Apache
(B) ColdFusion ClusterCATS
(C) Internet Information Server IIS
(D) Netscape Enterprise Web Server

Answer: B. ClusterCATS can leak usernames and passwords through a query string on http redirection.

References:

Roesch, Martin. Intrusion Detection Snort Style (Track 3). The SANS Institute, 2003.
"Allaire Security Bulletin (ASB00-12)." May 8, 2000. URL: http://packetstormsecurity.nl/advisories/allaire/asb00-12.querystring
"CVE-2000-0382." July, 2000. URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2000-0382
"Allaire ClusterCATS URL Redirect Vulnerability." May 8, 2000. URL: http://online.securityfocus.com/bid/1179

Network Detect #3 - Malformed IGMP Packets

```
[**] [1:527:4] BAD-TRAFFIC same SRC/DST [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
11/16-03:26:16.456507 170.129.71.37 -> 170.129.71.37
IGMP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:28
[Xref => http://www.cert.org/advisories/CA-1997-28.html]
[Xref => <http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0016>]

[**] [1:527:4] BAD-TRAFFIC same SRC/DST [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
11/16-03:26:16.456507 170.129.71.42 -> 170.129.71.42
IGMP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:28
[Xref => http://www.cert.org/advisories/CA-1997-28.html]
[Xref => <http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0016>]
```

Above are the two beginning alerts out of a total of twelve alerts that were generated. The remaining ten alerts are omitted for brevity since the only visible differences in the alert logs are the IP numbers.

Source of Trace:

This trace came from incidents.org raw log sets at http://www.incidents.org/logs/Raw/2002.10.16. This binary log is from an unknown Snort detection. The following diagram depicts the topology:

After Snort was run against this binary log file, which is described in the proceeding section, the output directory was listed to get an idea of the IP addresses that alerts got triggered on. It was determined that there were nine subnets in the 170.129.0.0 network and eight additional diverse networks with only one or two subnets each.

Next, Tcpdump was run to redirect the Tcpdump formatted binary log to a readable output file with the inclusion of MAC headers. The command used:

#tcpdump –en –r 2002.10.16 > 2002.10.16_TcpdumpOutput1
-n don't convert ip & port numbers to names
-e dump src & dst MAC addresses
-r read packets from file

Afterward, the 2002.10.16_TcpdumpOutput1 file was printed on several pages and two distinct MAC addresses were revealed, namely 00:03:e3:d9:26:c0 and 00:00:0c:04:b2:33. This tells us that Snort was logging between these two routing devices. According to Andre Cormier (GCIA 616), these MAC addresses are referenced at http://standards.ieee.org/regauth/oui/oui.txt as Cisco devices. Most likely one of these MACs is the inside interface hardware address of a Cisco border router which is labeled CISCO_DEVICE_1. The second MAC address is the outside interface of an internal Cisco router connecting the 170.129.0.0 network to the outside world. From skimming the printout, it can be seen that all destination IP numbers on the nine subnets in the 170.129.0.0 network are forwarded to the 00:00:0c:04:b2:33 MAC device or get sent from it. 170.129.0.0 must be INTERNAL_NET and sit behind MAC device 00:00:0c:04:b2:33, which we'll call CISCO_DEVICE_2. All IPs in the printout that cover the eight additional diverse networks originate from the 00:03:e3:d9:26:c0 MAC or get sent to it, because traffic is flowing in both directions. This MAC address we'll call CISCO_DEVICE_1, probably the border router. This is how the above topology was determined.

Twelve "BAD-TRAFFIC same SRC/DST" alerts were generated for this detect. All alerts were identified as IGMP group-specific query messages encapsulated in IP. Each packet was sent to a disparate node on the 170.129.71 subnet, a part of 170.129.0.0 (INTERNAL_NET).

Detect Generated By:

The "sameip" keyword within SID 527 triggered 12 alerts. Here is rule:

alert ip any any -> any any (msg:"BAD-TRAFFIC same SRC/DST"; sameip; reference:cve,CVE-1999-0016; reference:url,www.cert.org/advisories/CA-1997-28.html; classtype:bad-unknown; sid:527; rev:4;)

The alerts were generated by Snort v2.1.0, invoked in IDS mode, using the standard rule set. The tcpdump formatted binary file 2002.10.16 was used as input. Here are the commands:

#mkdir 2002.10.16_SnortOut
#snort -c /etc/snort.conf -l 2002.10.16_SnortOut -r 2002.10.16
-c specifies location of the snort configuration file
-l specifies directory for snort to dump alert logs
-r specifies the binary input file

Probability the Source Address was Spoofed:

The packet headers were consulted to determine if there was spoofing. TCPdump was invoked, piping output to grep using regular expressions.

#tcpdump -en -r 2002.10.16 | grep '\<170.129.71.*\>' > DetectOut1
-n don't convert ip & port numbers to names
-e dump src & dst MAC addresses
-r read packets from file

Note: only the first three packets headers are shown for brevity.

#cat DetectOut1
03:26:16.456507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 ip 60: 170.129.71.37 > 170.129.71.37: igmp query v2 [gaddr 240.0.2.21]
03:26:16.456507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 ip 60: 170.129.71.42 > 170.129.71.42: igmp query v2 [gaddr 240.0.2.26]
03:26:16.456507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 ip 60: 170.129.71.53 > 170.129.71.53: igmp query v2 [gaddr 240.0.2.37]

Format (using the first packet as the example):
03:26:16.456507 – timestamp
0:3:e3:d9:26:c0 – specifies MAC address that most recently forwarded the packet
0:0:c:4:b2:33 – specifies MAC address of the next hop device en route to destination
ip – specifies the IP protocol
60 – specifies the total datagram length
170.129.71.37 – specifies IP address that originated the datagram
170.129.71.37 – specifies IP address that should receive the datagram
igmp – specifies embedded protocol
query v2 – specifies that the igmp message is a membership query v2
gaddr 240.0.2.21 – specifies the group destination address within the igmp message

There are two possibilities. One, the source IPs of the packets were spoofed with the destination IPs. This would indicate that an attacker was involved. Two, the packets were not spoofed; they were instead sent to themselves because of a misconfigured IGMP setup.

It's impossible to believe that this was a misconfigured setup for the following reasons:
1. all 12 nodes on this single subnet would had to have been routers
2. all routers would have had to have been configured or elected as the designated lan querier as there should be only one per lan
3. group destination addresses (gaddr) are supposed to be Class D, not E.
4. there were 12 misconfigured group destination addresses in 12 queries sent on the same lan (could possibly be feasible if 1 group destination address was misconfigured, but all 12?)
5. all 12 queries have invalid destination MAC addresses as they should all have a multicast prefix of 01-00-5e
6. there are many additional malformations in the packets that will be described later

Also, bear in mind that the two MAC addresses associated with this traffic were identified as two separate Cisco devices. This precludes the packets from being originated and directed to the same nodes. Observing the TTL values of 46 for all 12 packets supports the idea that the packets were routed through Cisco devices 1 and 2. Tcpdump shows the TTLs as follows:

```
#tcpdump -nqvv -r 2002.10.16 | grep '\<170.129.71.*\>' > DetectOut2
-n      don't convert ip & port numbers to names
-q      quick output, omit protocol information
-vv     verbose option used to list TTL and IP ID values
#cat DetectOut2
```

(only first three packets listed and timestamps omitted for brevity)

```
170.129.71.37 > 170.129.71.37: igmp (ttl 46, id 0, len 28)
170.129.71.42 > 170.129.71.42: igmp (ttl 46, id 0, len 28)
170.129.71.53 > 170.129.71.53: igmp (ttl 46, id 0, len 28)
```

With all this supporting evidence, the packets must have been spoofed.

Description of the Attack:

This attack appears to be a random DoS attempt against IGMP that doesn't exist or hasn't been found (credit is due to Andrew Jones for this insight when he replied to the original posting for this detect on 03-04-03). There is no return address on the packets so mapping can be ruled out. The log file does not contain any additional packets than the twelve sited. It can't be discerned whether the nodes are workstations, servers, or routers. There were no

responses to these IGMP stimulus packets. There may be evidence of pseudo-random number generation in the logs. This idea will be explored in the next section.

The packets were logged on October 16, 2002 at 3:26am on the same millisecond. While it is believed that this attack was very fast, characteristic of flood packets, it is not believed that these packet deltas were within 1 millisecond of each other. This will be explained in the proceeding section as well.

Each packet was addressed to an individual IP in the 170.129.71 Class B subnet. All 12 packets contain uniform malformations except for destination IP numbers and IGMP group destination addresses, both which are unique. Let us examine these malformations starting at the bottom of the OSI model, working our way up. At Layer 2, the destination MAC address is 00:00:0c:04:b2:33 for every packet. This is an invalid MAC address for IGMP packets. According to RFC 1054, all IGMP multicast MACs should have a prefix of 01:00:5e. At Layer 3, the IP headers contain 3 faulty values. To begin with, all packets contain IP headers that are 20 bytes in length, the length of an IP header without options. IGMP packets should contain IP headers that are at least 21 bytes in length, to include an extra byte for IP option 20, the Router Alert option, which is a required option in IGMP packets. All 12 packets are void of this option. Here is a portion of a packet header without the timestamp that was captured previously.

170.129.71.37 > 170.129.71.37: igmp (ttl 46, id 0, len 28)

As you can see, the total datagram length is 28 bytes. 8 bytes are reserved for the IGMP message, leaving only 20 bytes for the IP header, which of course is void of any options. RFC 2113 describes IP option 20 in detail and how the 8 bits are to be filled. The next faulty field in the all of the IP headers of these packets is the TTL value. In the IGMPv2 RFC 2236 specification, IGMP query packets should contain a TTL value of 1 in their IP headers. All twelve packets contain an invalid TTL value of 46, which is shown above. The TTL field is 1 byte in length and specifies the hop count lifetime of the packets. It is decremented by 1 for each router a packet crosses. When the TTL hits 0, the packet is discarded. This helps prevent routing loops. The reason valid IGMP messages have a TTL value of 1 is because they're supposed to be limited to a LAN segment and are not to be routed. These packets have crafted TTL values in order to be routed. Lastly, the IP identification fields of these packets are all zero. According to IP standards, this 2-byte field is used to uniquely identify each datagram and is incremented by 1 for each datagram sent from a specific host. It is also used for matching IP packet fragments during reassembly. These IP IDs should be non-zero unique numbers, however, all 12 packets contain a zero value for this field. It's known that some Linux IP stacks use a zero value for the IP ID on initial connections, but it's more likely a script crafted the IP ID values along with many other fields.

Moving on to the embedded IGMP protocol message, it can be seen that the group destination addresses (GDAs) in these IGMP group-specific query packets are incorrect. Group-Specific membership queries, according to RFC 2236, should contain GDAs equal to the multicast group addresses being queried and fall within Class D address space (224.0.0.0 - 239.255.255.255). In these twelve examples, each packet has a unique Class E address (240.0.0.0/5) within the GDA field of its IGMP message. Redirecting the log again through TCPdump shows this.

```
#tcpdump -n -r 2002.10.16 | grep '\<170.129.71.*\>' > DetectOut2
-n      don't convert ip & port numbers to names
-r      read packets from file

(timestamps omitted)

170.129.71.37 > 170.129.71.37: igmp query v2 [gaddr 240.0.2.21]
170.129.71.42 > 170.129.71.42: igmp query v2 [gaddr 240.0.2.26]
170.129.71.53 > 170.129.71.53: igmp query v2 [gaddr 240.0.2.37]
170.129.71.47 > 170.129.71.47: igmp query v2 [gaddr 240.0.2.31]
170.129.71.69 > 170.129.71.69: igmp query v2 [gaddr 240.0.2.53]
170.129.71.74 > 170.129.71.74: igmp query v2 [gaddr 240.0.2.58]
170.129.71.63 > 170.129.71.63: igmp query v2 [gaddr 240.0.2.47]
170.129.71.20 > 170.129.71.20: igmp query v2 [gaddr 240.0.2.4]
170.129.71.7   > 170.129.71.7:   igmp query v2 [gaddr 240.0.1.247]
170.129.71.26 > 170.129.71.26: igmp query v2 [gaddr 240.0.2.10]
170.129.71.31 > 170.129.71.31: igmp query v2 [gaddr 240.0.2.15]
170.129.71.58 > 170.129.71.58: igmp query v2 [gaddr 240.0.2.42]
```

The only reference worth sighting is the LAND attack.

References:

CERT Advisory: CA-1997-28
CVE: CVE-1999-0016

Attack Mechanism:

It's difficult to deduce what caused these malformed IGMP packets. A faulty implementation has pretty much been rules out. There's no hard evidence that points to a deliberate attack since there's no evidence of any mapping or reconnaissance activity. In addition, the malformations seen in these twelve packets have not been documented publicly as causing any overflows or exploitations to any IGMP implementations. All that is known is that the malformations are not partial to any real IGMP implementations. As stated earlier, this is probably the result of somebody testing out a custom script. These packets were logged at 3:26am in the morning, possibly an ideal for someone to be fiddling around with a script. No packet crafting tools on the Internet were found that were capable of crafting these kinds of packets. If it were possible

that an attacker was running a script to cause a DoS on IGMP that hasn't been documented, then there should be some evidence of reconnaissance activity to show that the targeted hosts were running IGMP. If mapping was performed, it had of have been performed on a previous day. There was no information in the log file that indicates mapping was performed. No conclusions can be except that these packets are unlikely the result of a faulty implementation.

It's more likely that a homegrown packet-crafting tool created and transmitted these packets. All of the anomalies in these packets could have been crafted by a utility. The speed at which they were sent is indicative that a special tool was used. The high TTL values indicate that the packets were routed from afar, approximately 20 hops away is a good guess presumably. This is because the tool most likely used a TTL value of 64, since that's what most Unix/Linux type platforms use for their packets, and there wouldn't be much of a reason for the script to alter this value, unless it was known that the targets were farther away than 64 hops. The GCIA IDS Signatures and Analysis book states that most sites and servers are about 30-40 hops away and that a site 102 hops away has not been sited.

Considering a custom utility transmitted these packets it is then possible that, since there were no indications of reconnaissance activity performed, a pseudo-random number generator (PRNG) algorithm was used to generate some of the parameters since so many appear to be arbitrary. PRNGs are commonly used by worms for propagation. The parameters that will be considered are the destination IP numbers, indicating that this might have been a blind attempt, in addition to the group destination addresses. If a PRNG algorithm was added to the script, it's possible that poor quality random number generation was used if the script was homegrown. Poor quality random number generation means that the numbers generated may turn up a pattern, especially if the numbers were generated from a particular uniform or fixed distribution.

In the destination IP numbers, there appears to be some uniformity in the arithmetic differences among the numbers. Here is the list:

170.129.71.37	
170.129.71.42	5 more than the last
170.129.71.53	11 more than the last
170.129.71.47	6 less than the last
170.129.71.69	16 more than the last
170.129.71.74	5 more than the last
170.129.71.63	11 less than the last
170.129.71.20	43 less than the last
170.129.71.7	13 less than the last
170.129.71.26	19 more than the last
170.129.71.31	5 more than the last
170.129.71.58	29 more than the last

It looks like increases and decreases of 5 and 11 appear to be common. For all others, the one's digits match - 6 and 16, 43 and 13, 19 and 29. Is this just a coincidence or were the numbers contrived? Did a person choose the numbers or did a script generate them? If a person chose these numbers, on what basis were they chosen and why does there seem to be a pattern?

Also, there's some coherence between the destination IP numbers and the group destination addresses in the packets. Here is the destination IP number list with its corresponding GDA contained in the packet:

Packet#	DestIP#	GDA#
[01]	170.129.71.37	[gaddr 240.0.2.21]
[02]	170.129.71.42	[gaddr 240.0.2.26]
[03]	170.129.71.53	[gaddr 240.0.2.37]
[04]	170.129.71.47	[gaddr 240.0.2.31]
[05]	170.129.71.69	[gaddr 240.0.2.53]
[06]	170.129.71.74	[gaddr 240.0.2.58]
[07]	170.129.71.63	[gaddr 240.0.2.47]
[08]	170.129.71.20	[gaddr 240.0.2.4]
[09]	170.129.71.7	[gaddr 240.0.1.247]
[10]	170.129.71.26	[gaddr 240.0.2.10]
[11]	170.129.71.31	[gaddr 240.0.2.15]
[12]	170.129.71.58	[gaddr 240.0.2.42]

Seven out of the twelve packets have a destination IP host portion (4th octet) that matches the 4th octet of a GDA from a dissimilar packet. Each octet has 255 possible number choices and a little more than half of these packets have matching fields. This doesn't prove the theory that a PRNG was used to generate these values, but it doesn't rule out the possibility that they were randomly generated by a PRNG.

The identical timestamp phenomenon for these twelve packets can most likely be attributed to a bug in the resolution timer of the OS that recorded the packet logs. After researching the issue, it was found that multiple Linux kernel versions including 2.0.30, 2.1.126, and others provide useless sub millisecond timing resolution and should have their kernels patched or upgraded. Make note that every timestamp in the log file has the same exact last four digits, namely 6507. This means that the logger is definitely not providing accurate millisecond and sub ms timing resolution. The most accurate time delta found within the log file was between frames 221 and 222, with a 10ms delta:

Frame 221 03:26:58.196507
Frame 222 03:26:58.206507

It's unlikely that these 12 packets arrived within 1 millisecond of each other. It's

fair to say though, since the smallest non-zero time delta detected in the log file was 10ms, and it's backed by research that Linux kernel time resolution bugs limited resolution to 10ms, that the packets in this example arrived in 10ms for all twelve. This indicates that there was on average a 0.83ms time delta between each packet, still a very fast transmission stream.

Correlations:

Multiple GCIA students chose IGMP examples for their detects. As for publicly documented malformed IGMP group-specific query vulnerabilities, none were found. There are many documented IGMP vulnerabilities, but none of this nature. The LAND attack really isn't a correlation. The only reason it triggered the rule that generated the detect was because of the "sameip" keyword. Other than that, there is no correlation. The LAND attack requires tcp/udp ports, which this example is void of. Matching source and destination IP numbers are only a fraction of the anomalies in these packets.

GCIA Students:
Ashley Thomas, Vance Victorino, and Guru Cumarasamy analyzed alike traces from different log files. Brent Deterding and Buddy Smith analyzed different IGMP traces.

Evidence of Active Targeting:

There really isn't any evidence of active targeting. It is thought that the packets are more likely a result of a blind run of a script. Twelve destinations were chosen by user input or a number generator employed by the script. Since there was no indication of any mapping or reconnaissance activity, it can most likely be concluded that there was no active targeting involved in this process, unless we could prove that there was reconnaissance activity performed on a prior day, which we can't.

Severity:

Severity = (Criticality + Lethality) - (System Countermeasures + Network Countermeasures)

Criticality = 2. The level of criticality of the end nodes is impossible to determine from the log file. It is unknown whether they are workstations, servers, or routers. Without this information, a low criticality of 2 is assigned.

Lethality = 1. If the packets reached the end nodes, which there's a significant chance they didn't, there isn't any published vulnerability that states that IGMP implementations are vulnerable to any of the malformations seen in these packets. The only sure way to test the lethality of this attack is to write a tool that

crafts these exact packets and test them against various IGMP implementations to see if a DoS occurs.

System Countermeasures = 4. It's impossible to tell whether the destinations nodes were configured for IGMP. If they weren't, then the packets would be dropped on arrival. If the nodes were configured for IGMP, there still aren't any known vulnerabilities for these malformations. RFC 2236 states that a group-specific query will only be processed if it has a valid multicast group address, which these examples do not have. Most likely Class E group addresses will not be added to any multicast tables.

Network Countermeasures = 2. The perimeter of the network obviously did not block these packets since they reached the logger. That means any firewall or border routers didn't filter IP packets with IP protocol 2, the identification value for IGMP. Whatever CISCO_DEVICE_1 (00:03:e3:d9:26:c0) is, it didn't filter these packets. We don't know whether CISCO_DEVICE_2 (00:00:0c:04:b2:33) or any other internal firewalls/routers filtered these packets because the logger didn't have this field of vision. It's probably safe to assign a value of 2.

Severity = (2 + 1) - (4 + 2) = -3 (very low)

Defensive Recommendations:

One recommendation is to block IP/IGMP packets from being routed since they should be restricted to the LAN. This network employs Cisco hardware as do many networks. Considering Cisco Device 1, facing the external side, apply the following ingress filter to block IP packets with a protocol field value of 2 in the 9th byte offset in the IP header:

```
ciscodev-1#access-list 101 deny igmp any any log
ciscodev-1#conf t
ciscodev-1(config)#interface {external interface}
ciscodev-1(int-config)#ip access-group 101 in
ciscodev-1(int-config)#end
```

After the access list is entered, it is applied with the "access-group {list number} in" command to the interface facing the external side. The trailing "log" argument should be appended to the access list to log the attempt to a Syslog server since the packet will no longer pass through Cisco Device 1 for Snort to capture the event. Networks should have a variety of logging mechanisms in place to reach the highest level of auditing capability and additionally for log correlation purposes.

The second recommendation is to configure an anti-spoof filter to block source ip numbers that equal internal network numbers from entering the internal network. This will effectively block matching source and destination ip numbers. Here, an

access list will effectively provide spoofing protection for this example. Again, include the keyword “log” at the end of the statement to log offending packets to Syslog.

```
ciscodev-1#access-list 101 deny ip 170.129.71.0 0.0.0.0 170.129.71.0 0.0.0.0 log
ciscodev-1#conf t
ciscodev-1(config)#interface {external interface}
ciscodev-1(int-config)#ip access-group 101 in
ciscodev-1(int-config)#end
```

Either of these sets of commands, or both will effectively block these malformed IGMP packets. Note that the access-group statement needs to only be applied once to the interface in configuration.

The above sets of commands will block these packets at a network device. However, we do not know whether an actual vulnerability exists for any IGMP implementations. Most modern networks and systems are largely invulnerable to LAND-like attacks with matching source and destination IP numbers. The only way to tell for sure would be to write a packet craft tool that could regenerate these packets and test them against various OSs with diverse IGMP implementations to see if a DoS actually occurs. If there is a certain IGMP configuration that is vulnerable, a patch will need to be developed.

Multiple Choice Test Question:

An IGMPv2 host membership query carries a TTL of what hop count value?

a. 46
b. 44
c. 1
d. 0

Answer: C. This is because an IGMP host or router strictly operates on a LAN segment and uses a TTL of 1 (1 hop) to reach its destination. A higher TTL would indicate that the packet is to be routed.

Feedback & Responses from Intrusions List (intrusions@incidents.org)
Date of posting: January 22, 2004 2:46 PM

My original posting received no responses, but Donald Smith (GCIA) replied to my second posting. Donald’s responses are in blue. My thoughts and replies are in gray.

Donald's 1st Response (Feedback 1)
Subject: RE: LOGS: GIAC GCIA Version 3.4 Practical Detect Michael Bernstein
Date: Mon, 2 Feb 2004 12:48:41 -0700
From: "Smith, Donald" <Donald.Smith@qwest.com>
To: "Michael Bernstein" <mb_jobs@yahoo.com>, intrusions@incidents.org

My comments will be marked with djs
Djsdjs did you do any dumps with -d (data)?
I would like to see the igmp type and code.

(included marked comments only)

```
Djsdjs recommend you break these into one packet per line.
170.129.71.37 > 170.129.71.37: igmp (ttl 46, id 0, len 28)
170.129.71.42
> 170.129.71.42: igmp (ttl 46, id 0, len 28) 170.129.71.53 >
170.129.71.53: igmp (ttl 46, id 0, len 28) 170.129.71.47 >
170.129.71.47: igmp (ttl 46, id 0, len 28) 170.129.71.69 >
170.129.71.69: igmp (ttl 46, id 0, len 28) 170.129.71.74 >
170.129.71.74: igmp (ttl 46, id 0, len 28) 170.129.71.63 >
170.129.71.63: igmp (ttl 46, id 0, len 28) 170.129.71.20 >
170.129.71.20: igmp (ttl 46, id 0, len 28) 170.129.71.7 >
170.129.71.7:
igmp (ttl 46, id 0, len 28) 170.129.71.26 > 170.129.71.26: igmp (ttl
46,
id 0, len 28) 170.129.71.31 > 170.129.71.31: igmp (ttl 46, id 0, len
28)
170.129.71.58 > 170.129.71.58: igmp (ttl 46, id 0, len 28)
```

MB: these packets are actually 1 per line. They were just formatted wrong during the posting. See below:

```
170.129.71.37 > 170.129.71.37: igmp (ttl 46, id 0, len 28)
170.129.71.42 > 170.129.71.42: igmp (ttl 46, id 0, len 28)
170.129.71.53 > 170.129.71.53: igmp (ttl 46, id 0, len 28)
170.129.71.47 > 170.129.71.47: igmp (ttl 46, id 0, len 28)
etc..
```

Multiple Choice Test Question:

An IGMPv2 host membership query carries a TTL of what millisecond value?
Djsdjs millisecond value or hop count?
a. 46
b. 44
c. 1
d. 0

MB: My mistake. TTL is hop count, not millisecond.

My 1st Reply (Feedback 1)
Subject: RE: LOGS: GIAC GCIA Version 3.4 Practical Detect Michael Bernstein
Date: Tue, 3 Feb 2004 09:39:29 -0800 (PST)
From: "Smith, Donald" <Donald.Smith@qwest.com>
To: "Michael Bernstein" <mb_jobs@yahoo.com>, intrusions@incidents.org

Donald - I'll get you the app layer data for these packets. I believe they're set as membership queries 0x11 as the type but not sure about the code. Thanks for taking an interest in the detect. -Mike

from rfc2236.....
2.1. Type
There are three types of IGMP messages of concern to the host-router interaction:

0x11 = Membership Query
There are two sub-types of Membership Query messages:
- General Query, used to learn which groups have members on an attached network.
- Group-Specific Query, used to learn if a particular group has any members on an attached network.

These two messages are differentiated by the Group Address, as described in section 1.4 . Membership Query messages are referred to simply as "Query" messages.

My 2nd Reply (Feedback 1)
Subject: RE: LOGS: GIAC GCIA Version 3.4 Practical Detect Michael Bernstein
Date: Wednesday, February 04, 2004 12:11 AM
From: Michael Bernstein [mailto:mb_jobs@yahoo.com]
To: Smith, Donald
CC: intrusions@incidents.org

Interesting Donald... I am curious - I thought igmp just has types, not codes. What does code 31 indicate?

From rfc 2236##############################
The Max Response Time field is meaningful only in Membership Query messages, and specifies the maximum allowed time before sending a responding report in units of 1/10 second. In all other messages, it is set to zero by the sender and ignored by receivers.

Varying this setting allows IGMPv2 routers to tune the "leave latency" (the time between the moment the last host leaves a group and when the routing protocol is notified that there are no more members), as discussed in section 7.8. It also allows tuning of the burstiness of IGMP traffic on a subnet, as discussed in section 7.3.
##

Here's the dump you were asking for:

Frame 208 (60 bytes on wire, 60 bytes captured)
Arrival Time: Nov 16, 2002 03:26:16.456507000
Time delta from previous packet: 1551.890000000 seconds
Time relative to first packet: 28768.850000000 seconds
Frame Number: 208
Packet Length: 60 bytes
Capture Length: 60 bytes
Ethernet II, Src: 00:03:e3:d9:26:c0, Dst: 00:00:0c:04:b2:33
Destination: 00:00:0c:04:b2:33 (Cisco_04:b2:33)
Source: 00:03:e3:d9:26:c0 (Cisco_d9:26:c0)
Type: IP (0x0800)
Trailer: 00000000000000000000000000000000...
Internet Protocol, Src Addr: 170.129.71.37

(170.129.71.37), Dst Addr: 170.129.71.37
(170.129.71.37)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00:
Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint:
Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        ....0 = ECN-CE: 0
    Total Length: 28
    Identification: 0x0000
    Flags: 0x00
        .0.. = Don't fragment: Not set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 46
    Protocol: IGMP (0x02)
    Header checksum: 0xa993 (correct)
    Source: 170.129.71.37 (170.129.71.37)
    Destination: 170.129.71.37 (170.129.71.37)

Djsdjs ttl = 46 that implies this packet is crafted.

Internet Group Management Protocol
    IGMP Version: 2
    Type: Membership Query (0x11)
    Max Response Time: 10.0 sec (0x64)
    Header checksum: 0xfc85 (correct)
    Multicast Address: 240.0.2.21 (240.0.2.21)

0000 00 00 0c 04 b2 33 00 03 e3 d9 26 c0 08 00 45 00 .....3....&...E.
0010 00 1c 00 00 00 00 2e 02 a9 93 aa 81 47 25 aa 81 ....G%..
0020 47 25 11 64 fc 85 f0 00 02 15 00 00 00 00 00 00 G%.d............
0030 00 00 00 00 00 00 00 00 00 00 00 00   ....

Donald's 2nd Response (Feedback 1)
Subject: RE: LOGS: GIAC GCIA Version 3.4 Practical Detect Michael Bernstein
Date: Wednesday, February 04, 10:10:45 – 0700
From: "Smith, Donald" Donald.Smith@qwest.com
To: "Michael Bernstein" mb_jobs@yahoo.com
CC: intrusions@incidents.org

Take a quick look at http://www.securityfocus.com/bid/5020/discussion/
I dont think it applies but is very interesting related to igmp.

Your correct there is no "code" in igmp. The "code" is the max response
time field.

(Feedback 2)
Subject: RE: LOGS: GIAC GCIA Version 3.4 Practical Detect Michael Bernstein
Date: Tue, 3 Feb 2004 14:40:40 -0700
From: "Smith, Donald" <Donald.Smith@qwest.com>
To: "Michael Bernstein" <mb_jobs@yahoo.com>
CC: intrusions@incidents.org

This looks similar to a tool I know. Upscan.c an igmp scanner that was

used as part of synscan1.6. It used type 2 code 31.

Donald.Smith@qwest.com GCIA
http://pgp.mit.edu:11371/pks/lookup?op=get
<http://pgp.mit.edu:11371/pks/lookup?op=get&search=0xAF00EDCC>
&search=0xAF00EDCC
h8Hz

References:

"Snort Signature Database." 2004. URL: http://www.snort.org/snort-db/sid.html?sid=527
"GCIA Practical." http://www.giac.org/practical/GCIA/Andre_Cormier_GCIA.pdf
"GCIA Practical." http://www.giac.org/practical/GCIA/Ashley_Thomas_GCIA.pdf
"GCIA Practical." http://www.giac.org/practical/GCIA/Brent_Deterding_GCIA.pdf
"GCIA Practical." http://www.giac.org/practical/GCIA/Buddy_Smith_GCIA.pdf
"Intrusions@incidents.org position."
http://cert.uni-stuttgart.de/archive/intrusions/2003/07/msg00211.html
"RFC2236 – Internet Group Management Protocol, Version 2." Nov, 1997. URL: http://www.faqs.org/rfcs/rfc2236.html
"IP Option Numbers." June, 2001. URL: http://www.iana.org/assignments/ip-parameters
"Security Advisory: TCP Loopback DoS Attack (land.c) and Cisco Devices." Dec, 1997. URL:
http://www.cisco.com/en/US/tech/tk828/tk363/technologies_security_advisory09186a00800b1693.shtml
"IP option 20, Router Alert." URL:
http://www.networksorcery.com/enp/protocol/ip/option020.htm
Novak, Judy et al. TCP/IP for Intrusion Detection (Track 3). The SANS Institute, 2002.
Northcutt, Stephen. IDS Signatures and Analysis, Parts 1 and 2 (Track 3). The SANS Institute, 2002.
Roesch, Martin. Intrusion Detection Snort Style (Track 3). The SANS Institute, 2003.

Part 3 - Analyze This

Executive Summary:

This part of the assignment is based on the analysis of 15 log files that were captured during a five day period. The log files were obtained from an unknown university campus network whose topology is undetermined. The chosen date range of the log files is from 01/29/04 to 02/02/04. However, the dates of the actual log data are one year behind. "Analyze This" is broken down into a few sections. First, analysis of each of the Alert log data is performed. A link graph follows detailing the activity of an external host with multiple MY.NET hosts. Following, the Scan and OOS logs are analyzed. Registration information is provided for six external hosts whose activity is considered jeopardous to the university network. Lastly, general recommendations are made for the university's network.

FILES ANALYZED:

Table 1 provides a list of the files used for the proceeding analysis. Please note that the timestamps of the packets within the log files are one year lagged compared to these file dates.

Alert Logs	Scan Logs	OOS Logs
alert.040129	scans.040129	oos_report_040129
alert.040130	scans.040130	oos_report_040130
alert.040131	scans.040131	oos_report_040131
alert.040201	scans.040201	oos_report_040201
alert.040202	scans.040202	oos_report_040202

Note that a few alerts are not included in the analysis. They were removed due to malformations that precluded processing. The number counts in the proceeding analysis are not 100% exact, but do accurately represent the state of the university network.

Alert Analysis:

SUMMARY:

Snort triggered 50 unique alerts from the five alert log files. The proceeding table displays the 50 alerts and number of occurrences for each day with totals for the five-day period.

Alert Name	Jan29	Jan30	Jan31	Feb01	Feb02	Total
MY.NET.30.4 Activity	2579	781	4709	8175	4134	20378
MY.NET.30.3 Activity	3295	1948	2527	2287	464	10521
Incomplete Packet Fragments Discarded	1448	298	1292	2057	2831	7926

EXPLOIT x86 NOOP	628	1666	182	696	282	3454
SMB Name Wildcard	914	506	269	377	355	2421
High port 65535 tcp - possible Red Worm	291	510	1354	162	89	2406
High port 65535 udp - possible Red Worm	148	103	705	1090	139	2185
Possible trojan server activity	229	55	599	698	211	1792
Null scan!	278	330	311	270	249	1438
SUNRPC highport access!	25	46	13	922	129	1135
NMAP TCP ping!	180	168	161	143	155	807
External RPC call	145	0	0	0	189	334
TCP SRC and DST outside network	78	25	40	48	44	235
Tiny Fragments - Possible Hostile Activity	40	20	55	68	22	205
[UMBC NIDS] External MiMail alert	22	6	92	59	13	192
FTP passwd attempt	20	60	17	13	14	124
[UMBC NIDS IRC Alert] IRC user /kill detected, possible trojan	42	24	14	23	21	124
Traffic from port 53 to port 123	93	0	0	0	1	94
SMB C access	34	12	5	7	20	78
ICMP SRC and DST outside network	8	12	5	16	9	50
EXPLOIT x86 setuid 0	4	12	8	2	9	35
EXPLOIT x86 setgid 0	10	10	5	4	3	32
TFTP - Internal UDP connection to external tftp server	2	26	0	1	1	30
EXPLOIT x86 stealth noop	5	2	4	4	3	18
IRC evil - running XDCC	0	0	3	0	14	17
connect to 515 from inside	0	0	0	16	1	17
Probable NMAP fingerprint attempt	7	0	0	4	3	14
[UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected	0	0	1	0	8	9
TCP SMTP Source Port traffic	0	0	0	9	0	9
SYN-FIN scan!	2	0	0	1	4	7
DDOS shaft client to handler	0	2	1	1	2	6
TFTP - Internal TCP connection to external tftp server	0	3	0	0	2	5
RFB - Possible WinVNC - 010708-1	2	3	0	0	0	5
External FTP to HelpDesk MY.NET.53.29	1	1	0	1	0	3
External FTP to HelpDesk MY.NET.70.49	1	0	0	1	2	3
[UMBC NIDS IRC Alert] K\:line'd user detected, possible trojan	1	1	0	1	0	3
[UMBC NIDS IRC Alert] User joining XDCC channel detected. Possible XDCC bot	0	0	2	0	1	3
NIMDA - Attempt to execute cmd from campus host	0	0	2	1	0	3
Attempted Sun RPC high port access	0	1	0	1	0	2
External FTP to HelpDesk MY.NET.70.50	1	0	0	1	1	2
Fragmentation Overflow Attack	2	0	0	0	0	2
EXPLOIT NTPDX buffer overflow	0	0	1	0	1	2
FTP DoS ftpd globbing	1	0	0	1	0	2
TFTP - External UDP connection to internal tftp server	2	0	0	0	0	2
NETBIOS NT NULL session	2	0	0	0	0	2
FTP .forward	0	0	1	0	0	1
EXPLOIT identd overflow	0	0	1	0	0	1
TFTP - External TCP connection to internal tftp server	0	0	1	0	0	1
Happy 99 Virus	0	1	0	0	0	1
Fragmentation Overflow Attack	2	0	0	1	0	3

The alert analysis begins by analyzing the top 10 alerts that were the most frequently triggered following the two highest triggered alerts, MY.NET.30.4 and MY.NET.30.3 activity. Greg Bassett (GCIA 675) pointed out in his practical that these two MY.NET alerts are false positives so they will be omitted. The Top 10 Alert Analysis includes general descriptions of each of the 10 alerts, relevant traces from the alert logs, top talkers or top offenders, and suggestions based on how to handle the alerts. Relevant trace data is representative of attacks, compromises, false positives, and any meaningful data.

TOP 10 ALERT ANALYSIS:

Incomplete Packet Fragments Discarded 7,926 alerts

This alert was generated by the deprecated Snort defrag preprocessor (spp_defrag) which was superseded in Snort release 1.8 by the frag2 preprocessor (spp_frag2). As Glenn Larratt (GCIA 486) pointed out in his practical, the defrag preprocessor discards fragmented packets that are not at least half full when the last fragment arrives. This alert is attributed to false positives that are resultant of transmission problems, broken stacks, misbehaving applications, or it can alert positively to fragmentation attacks. In the log files, two examples are sighted that are indicative of two separate attacks from two different attacking hosts to two separate internal destination hosts. Refer to the following two tables:

Signature	Timestamp	SRC addr	DST addr
Incomplete Packet Fragments Discarded	2003-01-29 01:20:14	63.199.242.82:0	MY.NET.97.215:0
Incomplete Packet Fragments Discarded	2003-01-29 01:30:11	63.199.242.82:0	MY.NET.97.215:0
Fragmentation Overflow Attack	2003-01-29 01:22:48	63.199.242.82:0	MY.NET.97.215:0
Incomplete Packet Fragments Discarded	2003-01-29 01:22:48	63.199.242.82:0	MY.NET.97.215:0
Incomplete Packet Fragments Discarded	2003-01-29 01:31:09	63.199.242.82:0	MY.NET.97.215:0
Incomplete Packet Fragments Discarded	2003-01-29 01:23:56	63.199.242.82:0	MY.NET.97.215:0
Incomplete Packet Fragments Discarded	2003-01-29 01:32:14	63.199.242.82:0	MY.NET.97.215:0
Incomplete Packet Fragments Discarded	2003-01-29 01:24:38	63.199.242.82:0	MY.NET.97.215:0
Incomplete Packet Fragments Discarded	2003-01-29 01:33:38	63.199.242.82:0	MY.NET.97.215:0
Incomplete Packet Fragments Discarded	2003-01-29 01:26:02	63.199.242.82:0	MY.NET.97.215:0
Incomplete Packet Fragments Discarded	2003-01-29 01:34:40	63.199.242.82:0	MY.NET.97.215:0
Incomplete Packet Fragments Discarded	2003-01-29 01:27:01	63.199.242.82:0	MY.NET.97.215:0
Incomplete Packet Fragments Discarded	2003-01-29 01:27:43	63.199.242.82:0	MY.NET.97.215:0
Incomplete Packet Fragments Discarded	2003-01-29 01:28:45	63.199.242.82:0	MY.NET.97.215:0
Incomplete Packet Fragments Discarded	2003-01-29 01:46:04	63.199.242.82:0	MY.NET.97.215:0
Fragmentation Overflow Attack	2003-01-29 01:37:42	63.199.242.82:0	MY.NET.97.215:0
Null scan!	2003-01-29 01:37:42	63.199.242.82:0	MY.NET.97.215:0
Incomplete Packet Fragments Discarded	2003-01-29 01:47:13	63.199.242.82:0	MY.NET.97.215:0

This table above shows two Fragmentation Overflow Attacks interleaved with the Incomplete Packet Fragments Discarded alerts in the January 29th log file. The Fragment Overflow Attack alerts are supportive that these Incomplete Packet Fragments Discarded alerts are part of a larger fragmentation attack. During a Fragmentation Overflow Attack, the recipient's IP reassembly memory buffer is overwritten beyond its bounds and a DoS occurs. The function of the Null scan following the second Fragmentation Overflow Attack is probably a check to see whether the victim host is still responding or not. It also supports that the fragmentation is crafted and not naturally occurring since the null scan is a result of packet craft. This implies that source host 63.199.242.82 is malicious and isn't suffering from any misconfigurations or kernel bugs.

Signature	Timestamp	SRC addr	DST addr
Null scan!	2003-02-01 09:21:06	69.3.87.209:0	MY.NET.12.2:0
Incomplete Packet Fragments Discarded	2003-02-01 09:23:17	69.3.87.209:0	MY.NET.12.2:0
Incomplete Packet Fragments Discarded	2003-02-01 09:28:54	69.3.87.209:0	MY.NET.12.2:0
Null scan!	2003-02-01 09:30:12	69.3.87.209:0	MY.NET.12.2:0

Null scan!	2003-02-01 09:34:13	69.3.87.209:0	MY.NET.12.2:0
Incomplete Packet Fragments Discarded	2003-02-01 09:38:21	69.3.87.209:0	MY.NET.12.2:0
Null scan!	2003-02-01 09:41:26	69.3.87.209:0	MY.NET.12.2:0
Incomplete Packet Fragments Discarded	2003-02-01 09:42:43	69.3.87.209:0	MY.NET.12.2:0
Null scan!	2003-02-01 09:44:30	69.3.87.209:0	MY.NET.12.2:0
Incomplete Packet Fragments Discarded	2003-02-01 09:45:50	69.3.87.209:0	MY.NET.12.2:0
Incomplete Packet Fragments Discarded	2003-02-01 09:54:04	69.3.87.209:0	MY.NET.12.2:0
Null scan!	2003-02-01 10:00:30	69.3.87.209:0	MY.NET.12.2:0
Null scan!	2003-02-01 10:06:56	69.3.87.209:0	MY.NET.12.2:0
Null scan!	2003-02-01 10:30:27	69.3.87.209:39	MY.NET.12.2:61730
Null scan!	2003-02-01 10:25:19	69.3.87.209:0	MY.NET.12.2:0
Null scan!	2003-02-01 10:27:27	69.3.87.209:0	MY.NET.12.2:0
Null scan!	2003-02-01 10:41:34	69.3.87.209:0	MY.NET.12.2:0
Null scan!	2003-02-01 10:48:50	69.3.87.209:0	MY.NET.12.2:0
Null scan!	2003-02-01 10:50:07	69.3.87.209:0	MY.NET.12.2:0
Null scan!	2003-02-01 10:42:51	69.3.87.209:0	MY.NET.12.2:0
Null scan!	2003-02-01 10:43:42	69.3.87.209:0	MY.NET.12.2:0
Null scan!	2003-02-01 10:56:06	69.3.87.209:53	MY.NET.12.2:41446
Null scan!	2003-02-01 10:58:14	69.3.87.209:0	MY.NET.12.2:0
Fragmentation Overflow Attack	2003-02-01 11:01:13	69.3.87.209:0	MY.NET.12.2:0

The table above shows similar activity to the one above it except for the fact that the alerts came from a totally different source host from a disparate network three days later. The main difference is that there are many more Null scans. The intent appears to be the same – cause a DoS on the destination host. Packet craft is evident here as well. The reason there are more Null scans embedded in this alert trace is that the attacker is sending Null scans and fragmented packets at the same time. Null scans to a closed port will elicit a RST/ACK. If the attacker is sending the end host a fairly constant stream of fragmented packets that are malformed and Null scans to a closed port eliciting a response, when the host's IP defragmentation buffer is overflowed, the Null scans will timeout. This way the attacker will know whether or not his attack was successful. Spoofing is unlikely here since the attacker requires replies from the Null scan. Snort may be logging the fragmented packets as Incomplete Packet Fragments Discarded and then interpreting them as a Fragmentation Overflow Attack once a certain threshold is met that is indicative of problems with the fragments. One unusual item sighted in both the two examples above is that there are alerts that have timestamps that are out of order. It's not certain what caused this. It may be possible that the Snort logger is dropping packets or there is a bug. It should also be mentioned that in the second example two of the Null scans are using high numbered destination ports and unique source ports. The reason for this is unknown as neither destination port has any services registered with it or CVEs according to dshield.org. The attacker may be trying to determine if these ports are filtered by an intermediary device.

Notice that all the top three offenders are internal to the university network. Each generated a steady number of alerts for each day of the five day period. However, there weren't any correlations of these Incomplete Packet Fragments Discarded with scans of any type or Fragmentation Overflow Attacks which leads to the conclusion that these three hosts have transmission problems or broken stacks relating to kernel bugs or configuration issues. The alerts from these three top talkers are likely false positives.

TOP TALKERS

IP Address	# alerts
MY.NET.21.67	2,960
MY.NET.21.68	2,436
MY.NET.21.69	2,355

Recommendations:
Block the two offending IP addresses from the examples above, 63.199.242.82 and 69.3.87.209, which have likely performed Fragmentation Overflow Attacks. Investigate further external source addresses that have performed Fragmentation Overflow Attacks correlating with Incomplete Packet Fragments Discarded alerts. Additionally, the three top talkers should be investigated to see what is the cause of them generating consistent Incomplete Packet Fragments Discarded alerts.

Ensure that victims MY.NET.12.2 and MY.NET.97.215 have the latest networking patches in addition to all MY.NET hosts. These patches should include preventative overflow and bounds checking for IP defragmentation/reassembly so that their stacks stop accepting data when the buffer is full during packet reassembly.

EXPLOIT x86 NOOP 3,454 alerts

The meaning of this alert is when a series of NOP (no operation) instructions called a NOP sled is detected (e.g: 0x90 0x90 0x90). The x86 specifies NOP bytes 0x90 specific to Intel x86 architectures including 386,486,586,686, etc. This is used in malicious code when the exploited routine's address is hard to determine. The NOP bytes allow an attacker to pad the address space of the memory buffer so that the offset doesn't need to be precise. When the return address is located during the NOP sled execution, the buffer is overflowed and the attacker's exploit code is run. A series of NOPs may occur naturally inside executable files for alignment and optimization purposes. This traffic is also known to trigger false positives since innocuous binary traffic such as ftp and http transfers can trigger this alert. One host triggered 18 consecutive alerts for this signature on port 80. The timestamp chronology and ephemeral source port look genuine, and additionally, since this source host didn't trigger any other distinct alerts, it can be concluded that this activity is an http binary transfer representative of false positive alerting. Below is a sample of this activity.

Signature	Timestamp	SRC addr	DST addr
EXPLOIT x86 NOOP	2003-01-29 15:12:26	213.118.157.107:3343	MY.NET.32.167:80
EXPLOIT x86 NOOP	2003-01-29 15:12:26	213.118.157.107:3343	MY.NET.32.167:80
EXPLOIT x86 NOOP	2003-01-29 15:12:27	213.118.157.107:3343	MY.NET.32.167:80
EXPLOIT x86 NOOP	2003-01-29 15:12:27	213.118.157.107:3343	MY.NET.32.167:80
EXPLOIT x86 NOOP	2003-01-29 15:12:27	213.118.157.107:3343	MY.NET.32.167:80
EXPLOIT x86 NOOP	2003-01-29 15:12:27	213.118.157.107:3343	MY.NET.32.167:80
EXPLOIT x86 NOOP	2003-01-29 15:12:27	213.118.157.107:3343	MY.NET.32.167:80
EXPLOIT x86 NOOP	2003-01-29 15:12:27	213.118.157.107:3343	MY.NET.32.167:80
EXPLOIT x86 NOOP	2003-01-29 15:12:28	213.118.157.107:3343	MY.NET.32.167:80
EXPLOIT x86 NOOP	2003-01-29 15:12:28	213.118.157.107:3343	MY.NET.32.167:80

The table below is an example of NOP sleds used in a real exploit attack. Port 119, Network News Transfer Protocol (nntp), could be innocuous binary traffic from news server communication, but it's more likely that the news server has been compromised. Lone EXPLOIT x86 NOOP alerts may indicate false positives, but coupled with a EXPLOIT x86 setuid 0 alert indicates that a compromise has been made on the new servers. Setuid 0 indicates that the attacker has gained root privileges, since root has the user ID (UID) of 0.

Signature	Timestamp	SRC addr	DST addr
EXPLOIT x86 NOOP	2003-01-29 04:23:10	131.118.254.130:1072	MY.NET.24.8:119
EXPLOIT x86 NOOP	2003-01-29 04:23:10	131.118.254.130:1072	MY.NET.24.8:119
EXPLOIT x86 NOOP	2003-01-29 04:23:10	131.118.254.130:1072	MY.NET.24.8:119
EXPLOIT x86 NOOP	2003-01-29 04:23:10	131.118.254.130:1072	MY.NET.24.8:119
EXPLOIT x86 setuid 0	2003-01-29 06:54:51	131.118.254.130:1096	MY.NET.24.8:119
EXPLOIT x86 NOOP	2003-01-29 09:07:59	131.118.254.130:1150	MY.NET.24.8:119
EXPLOIT x86 NOOP	2003-01-29 09:07:59	131.118.254.130:1150	MY.NET.24.8:119
EXPLOIT x86 NOOP	2003-01-29 09:07:59	131.118.254.130:1150	MY.NET.24.8:119
EXPLOIT x86 NOOP	2003-01-29 09:07:59	131.118.254.130:1150	MY.NET.24.8:119

The top offenders are all from external sources as can be seen by the IP numbers below.

TOP OFFENDERS (External)

IP Address	FQDN	# alerts
65.93.189.44	sherbrooke-HSE-ppp3611661.sympatico.ca	509
67.33.199.81	adsl-33-199-81.lft.bellsouth.net	104
81.166.219.254	dyn-81-166-219-254.ppp.tiscali.fr	147
218.110.243.34	p6ef322.tokyte00.ap.so-net.ne.jp	111
219.95.165.76	Unable to resolve address	116
193.174.151.221	dhcp21.fh-bielefeld.de	631

Recommendations:

Since MY.NET.24.8 appears to have been compromised, this host should be taken offline and investigated for infection. Both external source addresses in the two examples above should be denied access, especially 131.118.254.130. The top offenders should be blocked at the perimeter unless they are trusted hosts. It would be a good idea to find out if these alerts were coupled with either the setuid 0 or setgid 0 alerts which would indicate active exploits. Source hosts that show correlating EXPLOIT x86 NOOP and setuid 0 or setgid 0 alerts should be immediately filtered. Lone EXPLOIT x86 NOOP alerts may represent innocuous binary transfers and generate false positives. Hosts generating these should be questioned whether they are trusted or untrusted hosts and to deny or allow them access.

SMB Name Wildcard 2,421 alerts

SMB (Server Message Block) is an intrinsic part of NetBIOS. SMB provides the services of the application, presentation, and session layers while NetBIOS handles the functions of the transport and network layers. Clients send SMB commands to servers to access shares, open files, and perform print operations.

Pre Win2k, TCP/IP clients required NBT (NetBIOS over TCP/IP) to run SMB. Win2k clients run SMB directly over TCP port 445 without the intervening NBT layer. This is the Snort signature for SMB Name Wildcard:

alert udp any any -> $HOME_NET 137 (msg:"SMB Name Wildcard"; content:"CKAAAAAAAAAAAAAAAAAAAAAAAAAAAA|0000|";)

This alert is triggered when Snort detects a wildcard "*" search for a host's NetBIOS table. It's used to discern all the resource names and types a host knows of. The content string in the above signature represents NetBIOS encoding and translates to the wildcard "*". SMB Name Wildcard scanning is usually followed by an attack or TCP connection on port 139 if there are open SMB shares on the scanned host.

Here is an SMB Name Wildcard reconnaissance probe. This scan is very fast as the timestamps indicate, being identical. It's certain that these are crafted packets due to the consecutive hosts being queried at a very fast rate.

Signature	Timestamp	SRC addr	DST addr	L4 Proto
SMB Name Wildcard	2003-01-29 14:44:00	MY.NET.80.197:1041	192.168.1.1:137	UDP
SMB Name Wildcard	2003-01-29 14:44:00	MY.NET.80.197:1041	192.168.1.2:137	UDP
SMB Name Wildcard	2003-01-29 14:44:00	MY.NET.80.197:1041	192.168.1.3:137	UDP
SMB Name Wildcard	2003-01-29 14:44:00	MY.NET.80.197:1041	192.168.1.4:137	UDP
SMB Name Wildcard	2003-01-29 14:44:00	MY.NET.80.197:1041	192.168.1.5:137	UDP
SMB Name Wildcard	2003-01-29 14:44:00	MY.NET.80.197:1041	192.168.1.6:137	UDP
SMB Name Wildcard	2003-01-29 14:44:00	MY.NET.80.197:1041	192.168.1.7:137	UDP

The table below appears to be an example of natural SMB Name Wildcard probes used for Windows host discovery. This is because a single host was chosen and the timestamps are well enough spaced.

Signature	Timestamp	SRC addr	DST addr	L4 Proto
SMB Name Wildcard	2003-01-29 15:06:58	MY.NET.153.94:137	216.145.5.196:137	UDP
SMB Name Wildcard	2003-01-29 15:07:04	MY.NET.153.94:137	216.145.5.196:137	UDP
SMB Name Wildcard	2003-01-29 15:08:36	MY.NET.153.94:137	216.145.5.196:137	UDP
SMB Name Wildcard	2003-01-29 15:09:37	MY.NET.153.94:137	216.145.5.196:137	UDP

The top talkers are all internal hosts. There is a strong possibility that all of these hosts are infected with worms that are trying to propagate via SMB Name Wildcard scans. It seems unlikely that such a high number of SMB Name Wildcard alerts could be attributed to natural Windows host discovery. The top talkers will need to be examined to prove this.

TOP TALKERS

IP Address	# Alerts
MY.NET.80.197	699
MY.NET.75.13	358
MY.NET.11.4	288
MY.NET.150.198	277
MY.NET.150.44	272

Recommendations:
Take the Top Talker hosts offline and investigate for possible infection. Block NetBIOS ports udp 137 (name service), udp 138 (datagram service), and tcp 139 (session service) from egressing and ingressing at the border router and firewall. This traffic should be restricted as LAN only traffic. If it must be allowed to and from external networks, monitor the traffic closely. In addition, prohibit tcp 445 (SMB over TCP/IP) at the perimeter as well.

High Port 65535 tcp/udp combined (Possible Red Worm) 4,591 alerts

This alert triggers when a source or destination host port number is equal to TCP or UDP 65,535 during connections. This is the highest possible port number that exists within the pool of dynamic port numbers for both TCP and UDP connections. 65,535 is a valid ephemeral port number used in normal TCP and UDP connections. After this port number is used, the port numbers are wrapped and begin again at the beginning of available ephemeral ports greater than 1024. However, it is also a sign of someone connecting to an infected and compromised host through a backdoor installed by the Adore worm. Adore is the name given to the Red worm. Adore is a variant of the Ramen worm that installs a backdoor by exploiting one of more of the following vulnerabilities mainly found in Red Hat systems, but also other Linux systems. The Adore worm is a self-propagating multi-exploit. Here are the exploits with links containing more specific information.

BIND remote exploit –
http://www.redhat.com/support/errata/RHSA-2001-007.html
LPRng exploit - http://www.redhat.com/support/errata/RHSA-2000-065-06.html
wuftpd remote exploit - http://www.redhat.com/support/errata/RHSA-2000-039-02.html
Rpc statd exploit - http://www.redhat.com/support/errata/RHSA-2000-043-03.html

Note that the RC1 Trojan and Sins Trojan also use TCP port 65,535. However, this rule was written with the Adore worm in mind and there is no way to determine this unless the payloads were available.

In a nutshell, once a vulnerable system is found, the “PS” binary is replaced with a trojaned version, email is sent to multiple addresses containing information about the compromised system, a cron job is added that runs daily to remove traces of Adore’s existence, and a special icmp package is ran that looks for an icmp packet of a certain length. When it sees this icmp packet, a rootshell backdoor is opened on port 65,535.

William Stearns from Darthmouth ISTS has contributed a tool that scans for systems and files that have been infected with the Adore Worm. Here is the link: http://www.ists.dartmouth.edu/IRIA/knowledge_base/tools/adorefind.htm

The following internal hosts may likely be infected with the Adore (Red) worm and should be investigated.

Internal hosts that should be investigated

IP Address	# Alerts	Unique DST IPs	L4-Proto
MY.NET.163.76	1,014	91	UDP
MY.NET.84.164	844	1	TCP
MY.NET.111.34	70	11	UDP
MY.NET.153.153	58	1	TCP
MY.NET.152.251	22	3	UDP
MY.NET.25.72	19	3	TCP
MY.NET.12.6	18	2	TCP
MY.NET.25.70	14	2	TCP

Correlations:
http://www.giac.org/practical/Michael_Reiter_GCIH.zip
http://www.pestpatrol.com/Whitepapers/PortsAndTrojans.asp
http://www.sans.org/y2k/adore.htm
http://www.cert.org/advisories/CA-2001-02.html
http://www.kb.cert.org/vuls/id/196945
http://www.linuxsecurity.com/advisories/turbolinux_advisory-1374.html

Recommendations:
Have the university campus network administrators download William Stearns Adore scanner and scan the MY.NET hosts listed above. Secondly, scan remaining MY.NET hosts and remove the worm. The latest anti-virus softwares should also detect this worm and perform removal.

Possible Trojan Server Activity 1,792 alerts

This alert triggers when either a source or destination host port number is equal to TCP 27374. This service port is well-known to the SubSeven Trojan as its default listening port. Hackers scan for listening TCP 27374 ports so they can connect to trojaned hosts. It's possible that this alert triggers false positives since 27374 is an ephemeral port in the pool of approximately 64k ports used as ephemeral ports. The presence of this alert is more likely tied to this well-known Trojan than harmless TCP connections. Here's an example of a MY.NET host controlling an external host that was compromised by this Trojan.

Signature	Timestamp	SRC addr	DST addr
Possible trojan server activity	2003-01-30 16:26:28	MY.NET.24.74:443	24.89.26.94:27374
Possible trojan server activity	2003-01-30 16:26:28	24.89.26.94:27374	MY.NET.24.74:443
Possible trojan server activity	2003-01-30 16:26:28	MY.NET.24.74:443	24.89.26.94:27374
Possible trojan server activity	2003-01-30 16:26:28	24.89.26.94:27374	MY.NET.24.74:443
Possible trojan server activity	2003-01-30 16:26:28	MY.NET.24.74:443	24.89.26.94:27374
Possible trojan server activity	2003-01-30 16:26:28	24.89.26.94:27374	MY.NET.24.74:443

The MY.NET attacking internal host 65.40.24.74 opened up a connection on port 443 to the infected SubSeven trojaned host 24.89.26.94 on port 27374. Six packets of communication are displayed above.

The following hosts in the internal network should be investigated immediately.

Either they are infected with the Trojan or are being directed by hackers for scanning for and controlling remote Subseven infected hosts.

MY.NET.5.20	MY.NET.24.34	MY.NET.190.1
MY.NET.6.15	MY.NET.24.44	MY.NET.190.95
MY.NET.6.16	MY.NET.24.74	MY.NET.190.97
MY.NET.12.2	MY.NET.29.3	MY.NET.190.102
MY.NET.12.4	MY.NET.60.17	MY.NET.190.202
MY.NET.12.6	MY.NET.75.13	MY.NET.190.203
MY.NET.24.33	MY.NET.153.221	

TOP OFFENDERS

IP Address	FQDN	# alerts
217.122.72.254	cp306825-a.gelen1.lb.home.nl	199
24.128.135.233	h0000e88e831e.ne.client2.attbi.com	136
68.112.209.79	cable-68-112-209-79.sli.la.charter.com	151
24.24.37.75	roc-24-24-37-75.rochester.rr.com	193
81.80.39.155	Unable to resolve address	186
64.109.212.223	adsl-64-109-212-223.dsl.lgnnmi.ameritech.net	135

Recommendations:
Ingress filter out the top offenders. In addition, it might be a good idea to egress filter connections to destinations on TCP port 27374 since it's possible that MY.NET hosts could be hackers trying to control these if any are infected with the SubSeven Trojan.

Correlations:
http://www.giac.org/practical/Simon_Tung_GCIA.doc

Null Scan 1,438 alerts

Null scanning is used for reconnaissance purposes. It is intended to be a stealth scanning technique to avoid detection. It's possible that this type of scanning activity can evade stateless routers or firewalls because none of the flags are set within the TCP header. It is also means of performing OS fingerprinting since different host architectures may respond differently. In theory, this scanning method employs inverse mapping in the sense that open ports won't respond to these packets, but closed ports will respond with packets that have the RST and ACK flags set in the TCP header.

TOP OFFENDERS

IP Address	FQDN	# alerts
63.251.52.75	www.shockwave.com	338
68.122.128.1	adsl-68-122-128-1.dsl.sndg02.pacbell.net	128
195.208.34.220	center.chph.ras.ru	85
80.213.65.73	ti500720a080-0329.bb.online.no	74
61.171.204.91	Unable to resolve address	71
81.6.217.192	81-6-217-192.gotadsl.co.uk	62

Recommendations:
Add these IP addresses to the ban list and block at the border router or firewall.

SUNRPC Highport Access 1,135 alerts

This alert is specifically designed to detect access to tcp and udp port 32771 as either a source or destination. Solaris hosts running RPC services typically listen in the range 32,771 – 34,000. Windows RPC services are restricted to tcp 135 and are not covered by this signature. The following Sun Solaris services are known to run on port 32771:

rpc.bind (udp 32771 – ghost portmapper, standard portmapper is 111)
rpc.nisd (known buffer overflow in NIS+)
rpc.ttdbserverd (known buffer overflow in Tooltalk DB server)

This port can be used to gain information about RPC services running on the destination host by using the following command:

rpcinfo -p host This command calls the DUMP RPC on the portmapper to obtain a list of all the registered RPC programs the host offers. This command can be scripted by attackers using common, non-standard, non-ephemeral source ports that firewalls allow into their networks in order to glean information about RPC services offered by destination hosts for the purpose of exploitation. That's exactly what's happening here. HTTP,SSL, and SMTP are usually allowed by firewalls inbound and outbound. It looks like three hosts were targeted on MY.NET. Normal RPC connections should be made from ephemeral source ports above 1023.

SRC Port 80

Signature	Timestamp	SRC addr	DST addr
SUNRPC highport access!	2003-01-30 09:57:53	206.98.174.20:80	MY.NET.67.26:32771
SUNRPC highport access!	2003-01-30 09:57:53	206.98.174.20:80	MY.NET.67.26:32771
SUNRPC highport access!	2003-01-30 09:57:53	206.98.174.20:80	MY.NET.67.26:32771
SUNRPC highport access!	2003-01-30 09:57:53	206.98.174.20:80	MY.NET.67.26:32771

SRC Port 443

Signature	Timestamp	SRC addr	DST addr
SUNRPC highport access!	2003-01-30 17:22:50	66.187.232.101:443	MY.NET.70.56:32771
SUNRPC highport access!	2003-01-30 17:22:5	66.187.232.101:443	MY.NET.70.56:32771

SRC Port 25

Signature	Timestamp	SRC addr	DST addr
SUNRPC highport access!	2003-01-30 11:03:41	144.126.75.19:25	MY.NET.25.66:32771
SUNRPC highport access!	2003-01-30 11:03:41	144.126.75.19:25	MY.NET.25.66:32771
SUNRPC highport access!	2003-01-30 11:03:41	144.126.75.19:25	MY.NET.25.66:32771
SUNRPC highport access!	2003-01-30 11:03:41	144.126.75.19:25	MY.NET.25.66:32771

The source IP address is probably not spoofed since a reply back is necessary to discern available RPC services. NIS+ and Tooltalk DB also run on this port and are susceptible to buffer overflows that can be exploited by attackers to run arbitrary code with root privileges. There are many RPC exploits available.

TOP OFFENDERS

IP Address	FQDN	# alerts
64.12.30.204	Unable to resolve address	509
128.183.16.124	halo.gsfc.nasa.gov	370

209.249.182.79	hmotteler.dsl.patriot.net	81
206.98.174.20	raba-020.raba.com	34
64.12.25.210	Unable to resolve address	14
12.167.138.42	www.aacc.edu	13

Recommendations:

This signature is narrow in scope and only detects access to this one port number while there are a plethora of RPC vulnerabilities extending beyond the use of tcp/udp port 32771. Ensure that Snort includes many of the other RPC signatures [SIDs: 569-600,612,937,1262 1299,1732,1733,1746,1747, 1890,1891,1905-1916,1922-1926,1931,1932,1949-1965]. It may be infeasible to ingress/egress filter required service ports like 80, 443, and 25. However, it makes sense to ingress filter access to all the SUNRPC high ports (32,771-34,000) unless absolutely required. Blocking 32,771 will effectively prevent connections like these regardless of source port. Investigate MY.NET Solaris and Unix hosts to determine whether they are vulnerable to any RPC services. If so, patch and disable this signature since it can generate a lot of false positives.

Correlations:

http://www.giac.org/practical/David_Singer_GCIA.doc
http://www.bekkoame.ne.jp/~s_ita/port/port30000-39999.html
http://www.snort.org/cgi-bin/sigs-search.cgi?sid=rpc
http://cgi.nessus.org/plugins/dump.php3?id=11111
http://www.lurhq.com/idsindepth.html
http://www.dshield.org
CVE-1999-0003 RPC Tooltalk
CVE-1999-0189 RPC BIND

NMAP TCP Ping 807 alerts

This alert is like the Null scan in the sense that it's an inverse mapping technique with the goal of evading stateless filtering devices and also devices that block ICMP types. NMAP was identified as the tool used to craft packets Snort alerted on because of the NMAP signature of a TCP ACK with the ACK=0 value. The packets sent have a lone ACK bit set in their IP headers. Live hosts should respond with a RST/ACK, tearing down the connection. Non-existing destinations shouldn't respond to this activity and that's what makes this technique a type of inverse mapping. It's likely that other tools beside NMAP transmitted these packets, since other tools can perform ACK scans and set the acknowledgement numbers to 0.

In the log files, two IP addresses had significantly higher occurrences of triggering this alert than any other IP addresses. Here are the two top talkers.

TOP TALKERS

IP Address	FQDN	# Alerts
63.211.17.228	proximitycheck1.allmusic.com	223
64.152.70.68	proximitycheck2.allmusic.com	213

It appears that the two top talkers above are being controlled by the same entity. Consulting the logs, both addresses appear to be crafting packets in the same fashion as can be seen below.

63.211.17.228

Signature	Timestamp	SRC addr	DST addr
NMAP TCP ping!	2003-01-30 01:28:11	63.211.17.228:80	MY.NET.1.3:53
NMAP TCP ping!	2003-01-30 01:28:11	63.211.17.228:80	MY.NET.1.3:53
NMAP TCP ping!	2003-01-30 01:37:09	63.211.17.228:53	MY.NET.1.3:53
NMAP TCP ping!	2003-01-30 01:37:09	63.211.17.228:80	MY.NET.1.3:53
NMAP TCP ping!	2003-01-30 02:09:21	63.211.17.228:53	MY.NET.1.3:53
NMAP TCP ping!	2003-01-30 02:09:21	63.211.17.228:80	MY.NET.1.3:53
NMAP TCP ping!	2003-01-30 07:45:51	63.211.17.228:53	MY.NET.80.133:2232
NMAP TCP ping!	2003-01-30 08:01:04	63.211.17.228:80	MY.NET.80.133:2534
NMAP TCP ping!	2003-01-30 12:50:17	63.211.17.228:80	MY.NET.1.3:53
NMAP TCP ping!	2003-01-30 12:50:17	63.211.17.228:80	MY.NET.1.3:53
NMAP TCP ping!	2003-01-30 14:08:10	63.211.17.228:53	MY.NET.84.157:49276
NMAP TCP ping!	2003-01-30 14:11:48	63.211.17.228:53	MY.NET.1.3:53
NMAP TCP ping!	2003-01-30 14:11:48	63.211.17.228:80	MY.NET.1.3:53

64.152.70.68

Signature	Timestamp	SRC addr	DST addr
NMAP TCP ping!	2003-01-30 22:08:12	64.152.70.68:80	MY.NET.1.3:53
NMAP TCP ping!	2003-01-30 22:08:12	64.152.70.68:53	MY.NET.1.3:53
NMAP TCP ping!	2003-01-30 22:33:27	64.152.70.68:80	MY.NET.1.3:53
NMAP TCP ping!	2003-01-30 22:33:27	64.152.70.68:53	MY.NET.1.3:53
NMAP TCP ping!	2003-01-30 22:45:03	64.152.70.68:80	MY.NET.1.3:53
NMAP TCP ping!	2003-01-30 22:45:03	64.152.70.68:53	MY.NET.1.3:53
NMAP TCP ping!	2003-01-30 23:22:39	64.152.70.68:80	MY.NET.1.3:53
NMAP TCP ping!	2003-01-30 23:22:39	64.152.70.68:53	MY.NET.1.3:53
NMAP TCP ping!	2003-01-30 23:35:12	64.152.70.68:53	MY.NET.1.4:53
NMAP TCP ping!	2003-01-30 23:35:12	64.152.70.68:80	MY.NET.1.4:53
NMAP TCP ping!	2003-01-30 23:35:12	64.152.70.68:80	MY.NET.84.242:3480
NMAP TCP ping!	2003-01-30 23:35:17	64.152.70.68:80	MY.NET.84.242:3480

TCP packets with the only the ACK flag set persisted from these two source IP addresses for destination IP addresses 65.40.1.2 through 65.40.1.4 for many more instances than the two tables above. The bulk of the traffic is made up of two packets sent at a given time, destined to port 53 and sourced from ports 53 and 80, ports that are normally left open by firewalls. A few attempts are made to disparate IP addresses with ephemeral ports. These packets are interleaved with the packets that are sent to port 53, a heavily trafficked port, in order to help to possibly camouflage them. They are spread out enough in the logs that hopefully they don't stand out too much in the logs. It appears that the attacker is trying to possibly identify the firewall secretly by embedding the discovery packets within harmless DNS traffic.

Recommendations:

Install a stateful firewall that will block ACK scans and explicitly ban IP addresses 63.211.17.228 and 64.152.70.68. Contact allmusic.com and inquire about two of their hosts scanning the campus network.

Correlations:

http://www.giac.org/practical/Mike_Bell_GCIA.doc

TCP SRC and DST outside network 235 alerts

The presence of this alert can be related to a few causes. This alert is triggered when neither source nor destination IP numbers are equal to MY.NET as defined by Snort. MY.NET equals 65.40. which is supposed to be HOME_NET. However, this alert is triggering on packets that match MY.NET which it shouldn't. The example below displays this.

Signature	Timestamp	SRC addr	DST addr
TCP SRC and DST outside network	2003-01-30 21:53:08	169.254.183.95:1141	MY.NET.240.194:2796
TCP SRC and DST outside network	2003-01-30 21:53:11	169.254.183.95:1141	MY.NET.240.194:2796
TCP SRC and DST outside network	2003-01-30 21:53:17	169.254.183.95:1141	MY.NET.240.194:2796
TCP SRC and DST outside network	2003-01-30 21:53:29	169.254.183.95:1141	MY.NET.240.194:2796

The destination address is an example of MY.NET indicating that MY.NET wasn't configured for all included networks, as in this example. This rule is most likely a false positive. Note that the timestamps of these packets indicate that the connection was unsuccessful since there is evidence of retries in the back-off timer. The second connection is three seconds apart from the first; the third connection is six seconds apart from the second, and the fourth connection is twelve seconds apart from the third. This also works as evidence that the source address wasn't spoofed because a valid TCP/IP stack is going to employ a TCP retransmit timer and back-off algorithm like we see in this example whereas a packet craft tool isn't going to wait between connection attempts.

On the contrary, this alert was also triggered by a spoofed loopback address. It is strange that this traffic was spoofed with the generic loopback address and yet still employs a TCP back-off algorithm and retransmit timer for every three packet attempts.

Signature	Timestamp	SRC addr	DST addr
TCP SRC and DST outside network	2003-01-30 14:15:24	127.0.0.1:5000	MY.NET.177.97:64407
TCP SRC and DST outside network	2003-01-30 14:15:27	127.0.0.1:5000	MY.NET.177.97:64407
TCP SRC and DST outside network	2003-01-30 14:15:39	127.0.0.1:5000	MY.NET.177.97:64407
TCP SRC and DST outside network	2003-01-30 16:19:07	127.0.0.1:5000	MY.NET.174.100:1089
TCP SRC and DST outside network	2003-01-30 16:19:11	127.0.0.1:5000	MY.NET.174.100:1089
TCP SRC and DST outside network	2003-01-30 16:19:13	127.0.0.1:5000	MY.NET.174.100:1089

Loic Juillard (GCIA 667) noted in his practical that many hosts generating this alert are coming from AOL dial-up accounts on a segment local to the campus network and that bad dialup or VPN configurations, possibly with a bad route, could trigger this alert. In the log files used here, large amounts of AOL traffic were seen generating these alerts, more so than any other. A very small number of RFC 1918 private addresses were sighted in these alerts as well.

Top Offenders

IP Address	FQDN	# alerts
127.0.0.1	localhost.localdomain	49
192.168.1.102	Unable to resolve address	37
172.130.19.189	AC8213BD.ipt.aol.com	14

Recommendations:
Make sure the INTERNAL_NET or MY.NET is complete with all subnets that are internal including the one in the example (65.40.240). Add anti-spoofing filters and ingress filter RFC 1918 addresses plus the loopback address from entering the network. It might be a good idea to anti-spoofing egress filters so the campus network isn't used as an attack springboard.

TOP 10 TALKERS (ALERTS)

The chart below shows the ten IP addresses that triggered the highest number of alerts in the alert logs. These are all internal hosts.

IP Address	FQDN	# Total Alerts
MY.NET.1.3	user3.net316.fl.sprint-hsd.net	211,010
MY.NET.162.92	user92.net477.nc.sprint-hsd.net	155,122
MY.NET.111.34	user34.net426.nc.sprint-hsd.net	102,193
MY.NET.1.4	user4.net316.fl.sprint-hsd.net	79,000
MY.NET.84.164	user164.net399.nc.sprint-hsd.net	65,384
MY.NET.163.107	user107.net478.nc.sprint-hsd.net	63,731
MY.NET.81.39	user39.net396.nc.sprint-hsd.net	62,040
MY.NET.153.37	user37.net468.lv.sprint-hsd.net	46,642
MY.NET.80.243	user243.net395.nc.sprint-hsd.net	20,838
MY.NET.34.14	user14.net349.fl.sprint-hsd.net	20,438

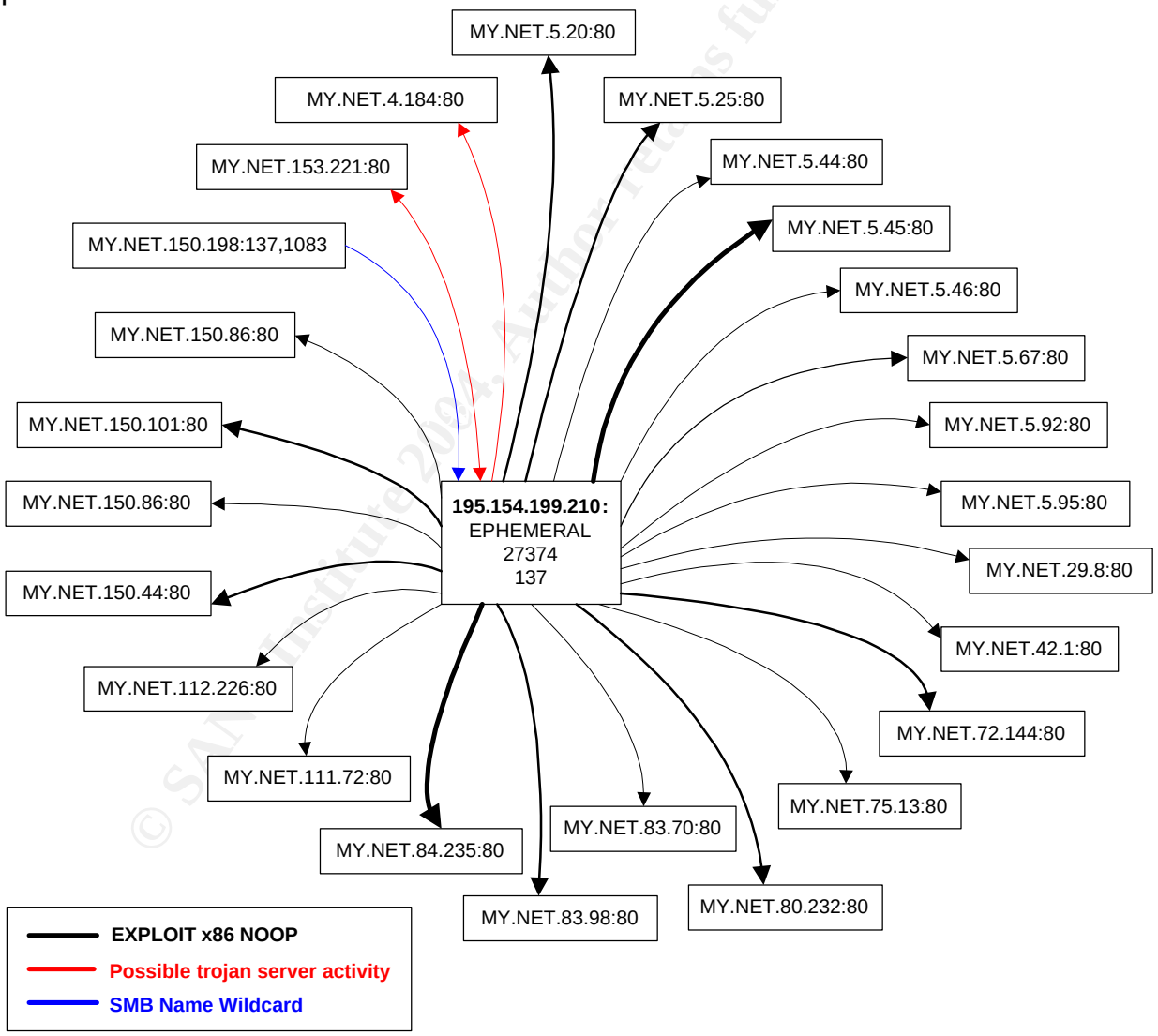
Link Graph

The link graph below shows the relationship between external host 195.154.199.210 and MY.NET hosts on February 2nd. There were a total of 207 EXPLOIT x86 NOOP alerts, 11 Possible Trojan server activity alerts, and 5 SMB Name Wildcard alerts. All connections attempts to MY.NET hosts were on tcp port 80 and sourced from an ephemeral port except for traffic to MY.NET.4.184 and MY.NET.153.221, which was sourced from port 27374 instead of an ephemeral port. The Possible Trojan server activity began at 4:58:09 and ended at 4:59:02. SMB Name Wildcard alerts immediately followed beginning at 4:59:01 and ending at 4:59:20. The EXPLOIT x86 NOOP alerts occurred approximately 10 hours 45 minutes after these two alerts. It is hard to say exactly what is going on here without full packet dumps, but there seems to be evidence of Ramen worm activity. Any host sighted with traffic to or from port 27374 should be considered suspect. It appears that 195.154.199.210 is infected with the Ramen worm and it also seems that the worm spread to MY.NET.153.221. MY.NET.153.221 seems to have carried out the last transaction of Ramen worm infection by connecting back to host 195.154.199.210 on port 27374 to download ramen.tgz to infect other hosts. The call that would be made is below. This link can be identified as the only bi-directional link in the link graph below.

```
GET / HTTP/1.0
Host: 195.154.199.210:27374
```

```
Accept: text/html, text/plain, audio/mod, image/*, video/*, video/mpeg,
application/pgp, application/pgp, application/pdf, message/partial,
message/external-body, application/postscript, x-be2, application/andrew-inset,
text/richtext, text/enriched
Accept: x-sun-attachment, audio-file, postscript-file, default, mail-file, sun-
desksetmessage, application/x-metamail-patch, text/sgml, */*;q=0.01
Accept-Encoding: gzip, compress
Accept-Language: en
User-Agent: Lynx/2.8.3dev.18 libwww-FM/2.14
```

It's possible that SubSeven or a SubSeven variant is active here, but there is no way to be certain. The interesting part is all the port 80 scans of MY.NET that occur later in the day. A SubSeven type Trojan isn't going to scan for port 80, but the Ramen worm could. The Ramen worm employs Synscan to scan for vulnerable hosts, and has been sighted to scan for port 80, but usually scans for port 21.



Correlations:

http://ciac.llnl.gov/ciac/bulletins/l-040.shtml

http://www.symantec.com/avcenter/venc/data/linux.ramen.worm.html
http://www.whitehats.com/library/worms/ramen/
http://www.dshield.org/port_report.php?port=80

Recommendations:
Block host 195.154.199.210 at the border and investigate MY.NET.153.221 and MY.NET.4.184 for possible Trojan or worm infection. Also, investigate MY.NET.150.198 to see why this host is scanning 195.154.199.210 for NetBIOS resources.

Scan Analysis:

This section describes the scanning activity that was present during the five-day period based on the scan logs. Some of this traffic, recorded as scan activity, should be ignored for it is considered general network traffic. This includes widely used public application protocols such as DNS, SMTP, HTTP, and FTP. These protocols should be ignored and the rest attended to.

Port	Service	# Attempts
53	*** dns** (name resolution & zone xfers)	3549184
135	* **RPC** services (Windows)	2998081
6129	Dameware Exploit (trojan)	477278
41170	Blubster, Piolet (both file sharing)	377003
25	*** smtp**	172418
80	*** http** (many documented vulnerabilities)	145391
4899	RADMIN (remote control/administration)	122147
6257	WinMX (file sharing)	101399
4000	Command and Conquer (game)	74050
1257	Shockwave2	69681
20168	Worm [controlled from IRC server]	67260
1214	Kazaa, Morpheous, Grokster (file sharing)	39729
6346	Gnutella (file sharing)	35872
21	*** ftp** (many documented vulnerabilities)	35267
9100	HP JetDirect (printing port)	20724
22321	Wnn6 (Korean input) tcp	19499

*** many documented vulnerabilities**

The remaining service ports that don't fall under the category of general network traffic should be blocked at the perimeter of the campus network. If given trusted hosts require use of these service ports, monitor them closely for malicious traffic. The most concerning ports are the file sharing/p2p ports and Trojan type remote administration ports that allow attacks, worms, and viruses to enter networks. RPC is the second highest used service port. Note that many known vulnerabilities and exploits exist for RPC services.

The table below shows the Top Talkers. These source IP addresses generated the highest number of scans. Note that they all originate from the same Class B network address space.

Source IP	# Attempted Scans
130.85.1.3	3083985

130.85.162.92	1464879
130.85.111.34	1260232
130.85.84.164	734902
130.85.163.107	610992
130.85.81.39	577009
130.85.1.4	479300
130.85.153.37	476850
130.85.80.243	230733
130.85.34.14	160616
130.85.53.225	138385
130.85.72.155	120739
130.85.97.12	106067
130.85.97.32	100716
130.85.163.76	98479
130.85.163.234	74089

The table below shows the destination IP addresses that were the highest scanned. To limit the scans seen by the campus network, it may make sense to place filters, such as ACLs, based on source network address or destination host address. This method would effectively reduce the number of bandwidth wasting scans.

Destination IP	# Attempted Scans
192.26.92.30	68647
192.5.6.30	41928
203.20.52.5	39961
69.6.33.10	39227
192.55.83.30	38969
69.6.33.11	34359
131.118.254.34	31956
192.48.79.30	31439
216.109.116.17	30906
209.92.188.201	30831
165.230.209.227	29714
131.118.254.33	28900
64.136.109.242	26440
69.20.36.152	26415
69.20.36.154	26185
128.194.254.5	24436
128.194.254.4	24076

OOS Analysis:

Packets of this nature are all logged with the same Snort signature, OOS (Out of Spec) for various reasons. Malicious packet crafting used in probes and attacks may set off this alert as well as non-malicious transient packet corruption caused by broken TCP/IP stacks, misbehaving applications, and malfunctioning or misconfigured hardware. Routers are a good example of a hardware device that can cause non-malicious packet corruption.

Here is a summary of the OOS alerts captured for the five-day period.

# OOS alerts	# Unique SRC IPs	# Unique DST IPs
4469	427	76

The table below shows the top 15 talkers that triggered the highest number of OOS alerts. Note that the top OOS talkers are all external addresses.

SRC IP address	FQDN	# OOS alerts	# Unique DST IPs
68.54.84.49	pcp01741335pcs.howard01.md.comcast.net	1166	1
207.138.63.21	Unable to resolve address	784	3
207.138.63.20	Unable to resolve address	166	3
66.225.198.20	unknown.servercentral.net	93	1
67.114.19.186	adsl-67-114-19-186.dsl.pltn13.pacbell.net	88	1
35.8.2.252	mdlv2.h-net.msu.edu	85	1
217.64.169.230	bulk05.india.192.com	81	1
217.88.221.218	pD958DDDA.dip.t-dialin.net	79	1
68.101.191.71	ip68-101-191-71.sd.sd.cox.net	58	2
68.122.128.1	adsl-68-122-128-1.dsl.sndg02.pacbell.net	57	1
207.228.236.26	Unable to resolve address	47	1
141.152.34.202	Unable to resolve address	47	2
62.58.92.114	users.linvision.com	46	1
62.210.155.58	Unable to resolve address	45	6
213.193.231.167	hope.webware.be	38	2

Something that immediately came to mind when examining the OOS logs was that the recorded timestamps are out of the range of the dates of the OOS logs. Not only are they one year lagged, but the timestamps begin on 02-01 instead of 01-29, and continue for the duration of six days instead of five.

Destination Port (tcp/110-pop3)

Source IP	Destination IP	# alerts
68.54.84.49	MY.NET.6.7	1166

This host is the top talker in the OOS logs. A total of 1,166 packets were sent to MY.NET.6.7 over six consecutive days. All packets were directed to the POP3 port, tcp 110. One packet was sent about every minute, too slow to be a SYN flood. Here a snippit of this activity.

Signature	Timestamp	SRC addr	DST addr
OOS	2003-02-02 04:05:39	68.54.84.49:33668	65.40.6.7:110
OOS	2003-02-02 04:06:42	68.54.84.49:33669	65.40.6.7:110
OOS	2003-02-02 04:07:45	68.54.84.49:33670	65.40.6.7:110
OOS	2003-02-02 04:08:52	68.54.84.49:33671	65.40.6.7:110

Note that the timestamps are approximately one minute and three seconds apart. The source port numbers steadily increment by one. This indicates that this was the only network activity taking place on the source host. In the logs, there was an interesting gap in source port numbers and timestamps. See the following table.

Signature	Timestamp	SRC addr	DST addr
OOS	2003-02-01 23:54:12	68.54.84.49:33433	65.40.6.7:110
OOS	2003-02-03 00:05:09	68.54.84.49:34870	65.40.6.7:110
OOS	2003-02-03 00:06:13	68.54.84.49:34871	65.40.6.7:110

This gap shows that the source host was busy with other network processing that consumed six ephemeral port numbers (33,434 - 34,869), and that this activity took about six minutes to complete. Afterward, the host resumed its activity of sending packet after packet to MY.NET.6.7 on tcp/110. A total of 5,371 ephemeral ports, from 33,444 to 38,815, were used by the source host.

It makes sense to believe that MY.NET.6.7 is a mail server and source host 68.54.84.49 is a pop3 client trying to communicate with it. From examining the packets, all have the SYN flag set as well as ECN/CWR flags, which indicate that there was congestion. All these packets appear to be connection attempts.

Recommendations:
Investigate MY.NET.6.7 to see if there is a configuration problem with this POP3 mail server. Check with the email administrator to ensure configuration is correct. Also, check the routing path and see why these packets are experiencing congestion, and get the issued resolved so the mail client can efficiently communicate with the mail server.

Source IP	Destination IP	# alerts
68.122.128.1	MY.NET.12.4	57

Host 68.122.128.1 is similar to the above example, however this host performed a fraction of the attempts that host 68.54.84.49 performed to tcp/110. Also, a different MY.NET host was targeted in this example. One important item to mention is that the no flags were set in these 57 OOS packets. And because there are so little, 57 spread out over a course of 5 days, this seems like we're dealing with scanning activity here. The packets were definitely crafted since none of the flags were set.

Recommendations:
Investigate this external host and implement a blocking firewall rule if necessary.

Destination Port (tcp/25-smtp)

Source IP	MY.NET.12.6	MY.NET.34.14	MY.NET.110.150	MY.NET.151.88
207.138.63.21	759	24	1	
207.138.63.20	161	4	1	
68.225.198.20	93			
35.8.2.252	85			
207.228.236.26	47			
141.152.34.202	46			
213.193.231.167				36
217.64.169.230				81

All of the packets sourced from 207.138.63.21 seem to be valid. The reason OOS alerts were triggered is because the ECN/CWR flags were set. The only other flag set in these packets is the SYN flag. It appears that network congestion was experienced when this source host was trying to make

connections to three different mail servers. Supporting evidence that indicates that these are mail servers is that other IP addresses were sighted in the logs trying to make connections on port tcp/25 to these three mail server addresses. Here are some IP addresses that tried connecting to MY.NET.110.150 on tcp/25: 195.140.185.22, 66.232.32.231.153, 66.232.231.208, 207.138.63.20, and 65.118.187.178.

Alike traffic originates from 207.138.63.20, but only a lesser amount. Source IP addresses 68.225.198.20, 35.8.2.252, 207.228.236.26, 141.152.34.202, 213.193.231.167, and 217.64.169.230 all fall under this category with the SYN and ECN/CWR flags set and no other. MY.NET.12.6, MY.NET.34.14, MY.NET.110.150, and MY.NET.151.88 are all legitimate mail servers.

Recommendations:
Do nothing. All four MY.NET hosts are mail servers. The reason OOS alerts were triggered is because of network congestion. Try and find out what is causing the congestion and if there is any way to remedy it.

Source IP	MY.NET.34.5
141.152.34.202	1

The single connection attempt to MY.NET.34.5 may be a mistake and MY.NET.34.5 might not be a mail server. There's no way to tell for sure since there were no other packets addressed to MY.NET.34.5 and no other packets sent from it.

Source IP	MY.NET.24.20
213.193.231.167	2

MY.NET.24.20 is probably a mail server. The following additional external addresses tried making connections to MY.NET.24.20: 209.152.161.131, 143.229.1.38, 213.193.231.167, 143.229.1.37, 193.231.18.20, 143.229.1.30, 143.229.1.40.

Recommendations:
Monitor traffic to MY.NET.34.5 and MY.NET.24.20 and determine whether the connections are legitimate. Also, find out whether these two MY.NET hosts are mail servers.

Destination Port (tcp/80-http)

Two external addresses from totally disparate networks were seen in the logs throughout the time period between 02-02-03 and 02-06-03 trying to connection on port 80 to the same host. All initial connection attempts were sourced from valid ephemeral ports and had the SYN, ECN-echo, and CWR flag bits set. The table below shows a summary of this activity.

Internal IP	67.114.19.186	62.58.92.114
MY.NET.24.44	88 alerts	46 alerts

The main difference between these two external hosts is that 67.114.19.186 made many more connection attempts. See table below.

Signature	Timestamp	SRC addr	DST addr
OOS	2003-02-02 01:30:02	67.114.19.186:36887	MY.NET.24.44:80
OOS	2003-02-02 01:30:03	67.114.19.186:36896	MY.NET.24.44:80
OOS	2003-02-02 01:30:03	67.114.19.186:36904	MY.NET.24.44:80
OOS	2003-02-02 01:30:03	67.114.19.186:36908	MY.NET.24.44:80
OOS	2003-02-02 02:30:02	67.114.19.186:38698	MY.NET.24.44:80
OOS	2003-02-02 02:30:03	67.114.19.186:38712	MY.NET.24.44:80
OOS	2003-02-02 02:30:03	67.114.19.186:38722	MY.NET.24.44:80
OOS	2003-02-02 02:30:03	67.114.19.186:38733	MY.NET.24.44:80
OOS	2003-02-02 03:30:03	67.114.19.186:40299	MY.NET.24.44:80
OOS	2003-02-02 03:30:03	67.114.19.186:40310	MY.NET.24.44:80
OOS	2003-02-02 03:30:03	67.114.19.186:40314	MY.NET.24.44:80
OOS	2003-02-02 03:30:04	67.114.19.186:40319	MY.NET.24.44:80

This table shows initial connection attempts by 67.114.19.186. They occurred approximately 30 minutes into each hour from hours 12am through 4:30am. This activity persisted for the five-day period.

Initial connection attempts from 62.58.92.114 to web server MY.NET.24.44 persisted for this five-day period as well. The difference is that the connection attempts never fell 30 minutes into each hour. The connection attempts were more random, not occurring at predictable intervals like the prior IP address. Additionally, packets from the prior example were sent in the same amount on each day of the five-day period whereas the packets from IP address 62.58.92.114 were mostly seen on 02-02 and lesser amounts of traffic on the following days. What is the same when compared with the prior example is that the connection attempts also occurred in groupings of four packets at a time. Below shows a table of some of the activity of this host.

Signature	Timestamp	SRC addr	DST addr
OOS	2003-02-02 02:45:54	62.58.92.114:46852	MY.NET.24.44:80
OOS	2003-02-02 02:45:54	62.58.92.114:46853	MY.NET.24.44:80
OOS	2003-02-02 02:45:54	62.58.92.114:46866	MY.NET.24.44:80
OOS	2003-02-02 02:45:54	62.58.92.114:46867	MY.NET.24.44:80
OOS	2003-02-02 02:53:41	62.58.92.114:47093	MY.NET.24.44:80
OOS	2003-02-02 02:53:41	62.58.92.114:47095	MY.NET.24.44:80
OOS	2003-02-02 02:53:41	62.58.92.114:47107	MY.NET.24.44:80
OOS	2003-02-02 02:53:41	62.58.92.114:47108	MY.NET.24.44:80
OOS	2003-02-02 03:16:21	62.58.92.114:47971	MY.NET.24.44:80
OOS	2003-02-02 03:16:21	62.58.92.114:47973	MY.NET.24.44:80
OOS	2003-02-02 03:16:21	62.58.92.114:47983	MY.NET.24.44:80
OOS	2003-02-02 03:16:21	62.58.92.114:47984	MY.NET.24.44:80

What is interesting about both these tables is that the groupings of each set of four packets was logged on nearly the same second in the first example and on the same second in the second example. Source port numbers in both examples show gaps between each packet so it is wise to assume that there must be other

network processes running on both source hosts that allocated these port numbers.

There is one remaining external source IP among the top 15 OOS talkers that made initial connection attempts to port 80. This host is different from the two above in that targeted five additional unique MY.NET destinations, including a single connection attempt to MY.NET.24.44. The table below summarizes host 62.210.155.58's port 80 connection attempts.

Internal IP	62.210.155.58
MY.NET.6.7	30 attempts
MY.NET.34.11	8 attempts
MY.NET.60.14	3 attempts
MY.NET.162.235	2 attempts
MY.NET.24.34	1 attempt
MY.NET.24.44	1 attempt

The SYN, ECN echo, and CWR bits were set in all of these packets. They were mostly likely logged by the OOS alerting mechanism because the ECN bits were set.

Recommendations:
Investigate all six of these MY.NET hosts and see whether they're legitimate web servers. If they aren't, there's a possibility that external source addresses trying to make connections to these servers are hostile and should be monitored and blocked if necessary. Also, try and find out why these connections are experiencing congestion. If congestion can't be verified, it's possible that packet crafting turned on the ECN flag bits.

Destination Port (tcp/6885)

All packets (79 total), sourced from this host 217.88.221.218, targeted MY.NET.82.8 on tcp/6885. The SYN, ECN echo and CWR flags are set. All this activity began at 12:08pm on 02-04-2003 and ended at 04:54pm on the same day. The ending sequence # for the last packet is 64,877 so it appears that the sequence number turnover period is near.

Recommendations:
Validate whether there is any application or service running on MY.NET.82.8 on tcp/6885. No registered applications turned up for this port on dshield or a google search. Investigate whether the source address is a trusted host find out why many connections would be made to this arbitrary port. Filter source IP as necessary.

Multiple Destination Ports (tcp)

A single IP, 68.101.191.71, scanned multiple ports on two MY.NET destinations residing on a common subnet. A total of 58 tcp datagrams were transmitted to various ports, with varying numbers. The following two tables represent this activity for each specific host. The source port numbers were static in reference to destination ports.

SRC Port #	DST Port #	MY.NET.42.2
1322	5316	16 attempts
2027	5314	2 attempts
2592	5616	6 attempts
2942	5384	5 attempts
2947	5689	12 attempts

SRC Port #	DST Port #	MY.NET.42.6
3017	5962	1 attempt
3467	5096	1 attempt
3642	5277	2 attempts
4049	5044	2 attempts
4149	5529	1 attempt
4151	5544	9 attempts

All packets transmitted to these two destinations have severe anomalies. Some are “christmas tree” packets with all the tcp flags set. Other packets are indicative of evasive scan packets, with the SYN+FIN flags set, to bypass stateless filtering devices. Many abnormal flag combinations within these packets were set that would trigger the OOS signature. None are specific to destination port number. For example the “christmas tree” type packets were addressed to various destination port numbers.

Recommendations:
Block source IP 68.101.191.71 at the border since this host is likely engaging in malicious scanning activity.

Registration Information

The ARIN Whois database was consulted at http://ww1.arin.net/whois/ to obtain the registration information for six external hosts that appeared throughout the analysis to be hostile.

131.118.254.130
This external host likely compromised MY.NET.24.8, a news server, since an EXPLOIT x86 setuid 0 alert was seen with consecutive NOP sleds for intel architectures.

OrgName: University of Maryland
OrgID: UNIVER-270
Address: System Administration
Address: 3300 Metzerott Road

City: Adelphi
StateProv: MD
PostalCode: 20783
Country: US

NetRange: 131.118.0.0 - 131.118.255.255
CIDR: 131.118.0.0/16
NetName: MINCNET
NetHandle: NET-131-118-0-0-1
Parent: NET-131-0-0-0-0
NetType: Direct Assignment
NameServer: NS.USMD.EDU
NameServer: UMCPNOC.UMS.EDU
NameServer: NOC.USMD.EDU
NameServer: TRANTOR.UMD.EDU
Comment:
RegDate: 1988-11-15
Updated: 1998-11-24

TechHandle: NM162-ARIN
TechName: Malmberg, Norwin
TechPhone: +1-301-445-2758
TechEmail: malmberg@usmh.usmd.edu

OrgTechHandle: NM162-ARIN
OrgTechName: Malmberg, Norwin
OrgTechPhone: +1-301-445-2758
OrgTechEmail: malmberg@usmh.usmd.edu

63.199.242.82
This host was selected because of initiating Fragmentation Overflow Attacks interleaved with Incomplete Packet Fragments Discarded alerts against MY.NET.97.215

Pac Bell Internet Services PBI-NET-7 (NET-63-192-0-0-1)
63.192.0.0 - 63.207.255.255
SNDG02 Rback4 PPPoX Pool SBCIS-000202-1405
(NET-63-199-240-0-1) 63.199.240.0 - 63.199.247.255

68.122.128.1
This host is also part of PacBell's network, most likely a DSL user. This host performed 128 NULL Scans, possibly looking to enumerate available services to target in a later attack.

Pac Bell Internet Services PBI-NET-10 (NET-68-120-0-0-1)
68.120.0.0 - 68.127.255.255
PPPoX Pool - Rback3 SNDG02 SBC068122128000030714

(NET-68-122-128-0-1) 68.122.128.0 - 68.122.129.255

217.122.72.254
This host is very likely hostile coming from the Netherlands. This host was the top talker for the Possible Trojan Server Activity alerts and is probably infected with the SubSeven Trojan or is a malicious user attempting to control computers that are infected with it.

OrgName: RIPE Network Coordination Centre
OrgID: RIPE
Address: Singel 258
Address: 1016 AB
City: Amsterdam
StateProv:
PostalCode:
Country: NL

ReferralServer: whois://whois.ripe.net

NetRange: 217.0.0.0 - 217.255.255.255
CIDR: 217.0.0.0/8
NetName: 217-RIPE
NetHandle: NET-217-0-0-0-1
Parent:
NetType: Allocated to RIPE NCC
NameServer: NS.RIPE.NET
NameServer: NS3.NIC.FR
NameServer: SUNIC.SUNET.SE
NameServer: AUTH00.NS.UU.NET
NameServer: SEC1.APNIC.NET
NameServer: SEC3.APNIC.NET
NameServer: TINNIE.ARIN.NET
Comment: These addresses have been further assigned to users in the RIPE NCC region. Contact information can be found in the RIPE database at http://www.ripe.net/whois
RegDate: 2000-06-05
Updated: 2003-09-19

OrgTechHandle: RIPE-NCC-ARIN
OrgTechName: RIPE NCC Hostmaster
OrgTechPhone: +31 20 535 4444
OrgTechEmail: search-ripe-ncc-not-arin@ripe.net

68.101.191.71
This source IP, from the OOS log data, scanned two MY.NET destinations with packets that had an array of crafted TCP flag combinations.

Cox Communications Inc. SD-RDC-68-101-128-0
(NET-68-101-128-0-1) 68.101.128.0 - 68.101.255.255
Cox Communications Inc. COX-ATLANTA-2
(NET-68-96-0-0-1) 68.96.0.0 - 68.111.255.255

65.93.189.44
This host is the top talker for the EXPLOIT x86 NOOP alerts. This host triggered approximately five times the number of EXPLOIT x86 NOOP alerts as the second top talker.

Bell Canada BELLNEXXIA-10
(NET-65-92-0-0-1) 65.92.0.0 - 65.95.255.255
Bell Nexxia (High Speed) HSSHER-CA
(NET-65-93-160-0-1) 65.93.160.0 - 65.93.191.255

General Recommendations

The security analysis of the university campus network shows that there are many changes that should be made to improve the state of network security. To begin with, Snort should have its fragmentation reassembly preprocessor, spp_defrag, upgraded to spp_frag2. This will improve the handling of fragmented packets and reduce false positives related to fragmentation. Next, follow all recommendations throughout this assignment to block offending external addresses at the network perimeter. Ensure that all MY.NET hosts, whether they be Windows or Unix systems, have the latest service packs, bug fixes, and patches – especially for running RPC services. It is also best practice to filter known Windows ports at the perimeter including 135 tcp/udp, 138 udp, 139 tcp/udp, and 445 tcp/udp. Traffic to these ports should be restricted to internal use only. As for Unix hosts, be on the lookout for EXPLOIT x86 setuid 0 and setgid 0 alerts, since their presence highly suggests Unix root compromise. Additionally, turn off unneeded RPC services. Investigate the following internal hosts for SubSeven infection: MY.NET.5.20, MY.NET.6.15, MY.NET.6.16, MY.NET.12.2, MY.NET.12.4, MY.NET.12.6, MY.NET.24.33, MY.NET.24.34, MY.NET.24.44, MY.NET.24.74, MY.NET.29.3, MY.NET.60.17, MY.NET.75.13, MY.NET.153.221, MY.NET.190.1, MY.NET.190.95, MY.NET.190.97, MY.NET.190.102, MY.NET.190.202, and MY.NET.190.203. There's a strong possible that many of the university computers are running unnecessary web servers and that they should be turned off it they're not supposed to be serving web pages, since http/80 is a highly exploited application layer protocol. Validate the following MY.NET hosts taken from the link graph for running web servers: MY.NET.4.184, MY.NET.5.20, MY.NET.5.25, MY.NET.5.44, MY.NET.5.45, MY.NET.5.46, MY.NET.5.67, MY.NET.5.92, MY.NET.5.95, MY.NET.29.8, MY.NET.42.1, MY.NET.72.144, MY.NET.75.13, MY.NET.80.232, MY.NET.83.70, MY.NET.83.98, MY.NET.84.235, MY.NET.111.72, MY.NET.112.226, MY.NET.150.44, MY.NET.150.86, MY.NET.150.101, MY.NET.150.86, and

MY.NET.153.221. Also, validate POP3 and SMTP mail servers running on the internal network since there were a lot of these seen. For file sharing programs, define a policy that allows or disallows their usage. Some, if not all of the following file sharing programs were active on the network: Kazaa, Morpheous, Grokster, Gnutella, WinMX, Blubster, and Piolet. Lastly, disable ICMP unreachable errors on routers to prevent reconnaissance information from being gathered.

Process for Part3

Joe Bowling (GCIA 674) was kind enough to provide me with perl scripts written by Ryan Johnson to help with Part 3 of this assignment. These perl scripts include a script that clears the ACID database (acid_flushall.pl), alert.pl, snortOOSclean.pl, destCount.pl, srcCountpl, and portCount.pl. I used the UNIX 'cat' command to 'cat' the OOS logs and Scan logs together, but had to analyze the alert logs one by one. When I was finished my queries in ACID, I used the acid_flushall.pl script to clear out the MySQL database. I followed Patrick S. Harper's document for installing Snort with Apache, PHP, MySQL, ACID on Redhat 9.0. The installation went smooth, but I had some issues with ACID and had to do a complete reinstall once, and partial reinstalls of MySQL and ACID to get everything working again a few times. I learned that while running these scripts to import log data into MySQL that the Ethernet cable should be unplugged from the NIC card since ACID was freezing up trying to sniff LAN traffic and deal with the log data from the logs at the same time. My computer that housed this setup was a Pentium 700 Mhz with 512 MB ram. Once everything loaded into ACID, I did a search and set the variable to "Fast Mode" to process searches for the Alert logs. The OOS logs I didn't have to do that since the signature was the same for all the logs. Instead I chose the top 15 source addresses and performed my queries from there. For the Scan logs I cat'd the 5 days together and ran them against the three scripts: destCount.pl, srcCount.pl, and portCount.pl. The output files I ftp'd to my other computer and queried through an MS Access database or grep'd directly on the Redhat computer for the information I was looking for. One important thing to mention that I had to do to get these perl scripts working was downgrade Perl from 5.8.1, which comes bundled with the Redhat 9 installation, to Perl 5.6.1. This was mandatory to support the execution of these scripts. I downloaded the following rpm (redhat package manager) and performed the following command to downgrade:
rpm –Uvh –nodeps –force perl-5.6.1-0rh71.i386.rpm

References

Snort.org Signature Database. "SHELLCODE x86 NOOP." 2004. URL: http://www.snort.org/snort-db/sid.html?sid=648

Snort.org Signature Database. "SCAN nmap TCP." 2004. URL:

http://www.snort.org/snort-db/sid.html?sid=628

Snort.org Signature Search. “Search for Snort RPC Signatures.” 2004. URL: http://www.snort.org/cgi-bin/sigs-search.cgi?sid=rpc

Security Focus Mailing List. “SHELLCODE x86 NOOP.” Apr, 2002. URL: http://www.derkeiler.com/Mailing-Lists/securityfocus/focus-ids/2002-04/0035.html

Byte.com. “Unix and Windows Dance the Samba.” Mar, 1997. URL: http://www.byte.com/art/9703/sec5/art2.htm

Frisch, Aeleen. “Sharing File Systems, Part 2.” Jun, 1998. URL: http://swexpert.com/C5/SE.C5.JUN.98.pdf

Snort Faq. “Q: SMB Name Wildcard alerts.” Mar, 2002. URL: http://www.snort.org/docs/faq.html#4.15

Wray, Steve. “Steve Wray’s mIRC Pages.” 1998. URL: http://www.yippyskippy.com/servers.html

Red Hat Linux Manuals. “Red Hat Linux 6.2: The Official Red Hat Linux Getting Started Guide.” URL: http://www.redhat.com/docs/manuals/linux/RHL-6.2-Manual/getting-started-guide/ch-glossary.html

Hawley, Bart E. “Port 554/tc discussion.” Oct, 2003. URL: http://lists.virus.org/dshield-0310/msg00012.html

LockDown Corp. “SubSeven Trojan Demo.” 2002. URL: http://lockdowncorp.com/trojandemo.html

Key Focus. “xfSubSeven honeypot emulation.” URL: http://www.keyfocus.net/kfsensor/extras/

AusCERT. “Potential Increase in “Code Red” Worm Activity.” Jul, 2001. URL: http://www.auscert.org.au/render.html?it=106

F-Secure. “F-Secure Virus Descriptions: CodeRed.” Aug, 2001. URL: http://www.f-secure.com/v-descs/bady.shtml

Sourcefile. “Snort ChangeLog.” Nov, 2001. URL: http://www.jheiss.com/res/usr/share/doc/snort-1.8.2/ChangeLog

“Port 30000 – 39999.” Feb, 2004. URL: http://www.bekkoame.ne.jp/~s_ita/port/port30000-39999.html

Arbio, Michel. “Nessus.org port mapper search.” 2002. URL:

http://cgi.nessus.org/plugins/dump.php3?id=11111

LURHQ Threat Intelligence Group. "Intrusion Detection: In-Depth Analysis." 2003. URL: http://www.lurhq.com/idsindepth.html

Dshield.org. "Port Report." Mar, 2004. http://www.dshield.org

Sans.org. "Adore Worm." Apr, 2001. URL: http://www.sans.org/y2k/adore.htm

Cert.org. "CERT Advisory CA-2001-02 Multiple Vulnerabilities in BIND." Aug, 2001. URL: http://www.cert.org/advisories/CA-2001-02.html

US Computer Emergency Readiness Team. "ISC BIND 8 contains buffer overflow in transaction signature (TSIG) handling code." May, 2002. URL: http://www.kb.cert.org/vuls/id/196945

Guardian Digital. "Turbolinux Security Announcement: Adore Worm." 2000. URL: http://www.linuxsecurity.com/advisories/turbolinux_advisory-1374.html

ARIN Whois Database. URL: http://ww1.arin.net/whois/

Bowling, Joe. "GCIA Practical Assignment." URL: http://www.giac.org/practical/GCIA/Joe_Bowling_GCIA.pdf

Larratt, Glenn. "GCIA Practical Assignment." URL: http://www.giac.org/practical/Glenn_Larratt_GCIA.zip

Bassett, Greg. "GCIA Practical Assignment." URL: http://www.giac.org/practical/GCIA/Greg_Bassett_GCIA.pdf

Bell, Mike. "GCIA Practical Assignment." URL: http://www.giac.org/practical/Mike_Bell_GCIA.doc

Singer, David. "GCIA Practical Assignment." URL: http://www.giac.org/practical/David_Singer_GCIA.doc

Tung, Simon. "GCIA Practical Assignment." URL: http://www.giac.org/practical/GCIA/Simon_Tung_GCIA.pdf

Reiter, Michael. "GCIA Practical Assignment." URL: http://www.giac.org/practical/Michael_Reiter_GCIH.zip