# Global Information Assurance Certification Paper

# Putting Eyes on the Wire

## Deploying a Sensor Network at a Large University

GSEC Practical Assignment
Version 1.4b
Option 2: Case Studey

By Don Murdoch, CISSP



Submission Date:
Tuesday, December 14, 2004

In conjunction with Mary Washington College / James Monroe Center Graduate Certificate in Information Security

# Table of Contents



## List of Figures

## Abstract

This paper discusses the deployment of a network of recycled PC class systems that are used as an internal sensor network - a network designed to find PC systems that are compromised from various viruses or otherwise misusing network resources. Many techniques are borrowed from honeyport and honeynet methodologies as articulated by the HoneyNet project. Historically a major virus outbreak occurs every few months on a University network – and detecting that outbreak is a main goal of the sensor network. Viri often run rampant on a University network - one that follows an "Allow by Default" perimeter security stance, bearing in support of the principle of academic freedom. The sensor network is designed to look for network intrusions and to inform the security / network teams about anomalies, Events of interest, and suspicious system behaviors so that they can identify misbehaving systems as early as they can and begin the process of remediation.

## Project Roles

Several people participated in this project:

- Information Systems Security Officer (ISSO) (paper author): architecture, solution design, tool selection, script development, project management, Snort and LeBrea2 configuration.
- Network Engineer: network management, switch/router configuration, and locating systems on campus.
- Unix System Admin: initial system build and system maintenance.

## Before the Sensor Net

Several times through the years (usually clustered around a semester boundary) two general classes of security issues are encountered on the network - viruses spreading like wildfire and general network misuse in violation of University policy. Throughout the rest of a given semester there is often remediation of infected clients (almost always Microsoft Windows based) and other issues relating to general misuse - accidental and intentional. In order to deal with these problems, the network team and the security team needed to move a single Linux based PC from location to location around the campus in order to assess the network. During assessment and data capture (w/ tcpdump and Snort). The author usually wrote customized scripts to analyze subsets of data in order to deal with security issues.

Two recent examples include viruses that attacked unprotected Microsoft Windows systems. The MS Blaster[1] worm impacted the network last summer,

[1] Symantec has a write up on the virus posted on their website. See: http://securityresponse.symantec.com/avcenter/venc/data/w32.blaster.worm.html

with remediation during Q3 2003 taking between 12 and 13 person months[2]. This virus had a few specific signatures that Snort could easily identify - the Cyberkit 2.2 ping, NetBIOS packets of a certain size, and tftp activity being the three main examples. Another recent example was the Sasser[3] worm, which involves traffic on three ports: 5554, 9996, and 445. These two worms highlight the risk that the network faces - without corporate grade controls on the network (such as "Deny by Default" firewalls) the network and its users face high risk of the loss of productivity when a worm propagates.

Virus outbreaks impact the network - and thus, the people using the network - posing a serious threat to the health of the LAN and to productivity. Without having a more comprehensive means of finding infected computers it is very difficult to combat these threats to the network. For example, on a network with an average of 7,000 systems with only 70% of the systems coming under centralized management effective remediation of a system not under centralized control can take days to get to and then hours to actually clean the infection. For centrally managed systems an antivirus update and then a scan can be forced - with the end user being impacted. More specifically, running McAffee's "stinger" virus removal tool and updating the system with the most recent patches and updates can take 1 to 1.5 hours. Performing those tasks several times per year would keep a few system administrators gainfully employed doing nothing but hunting down viruses!

The same is true for other security issues - policy violations, excessive bandwidth utilization, internal network scanning, game servers, and systems that are internally compromised with a backdoor Trojan.

By having a specially configured PC that can be plugged into a switch the network team and security team could analyze traffic on the LAN - manually, with always a delay in moving a system around the network. Not to mention that wire closets are not the most comfortable working space. This approach has long outlived its usefulness as it is primarily reactive and makes no effort to bring any proactive stance to network security. A single PC also significantly limits analysis capability as it can only see part of the network at one time.

After dealing with numerous virus outbreaks and policy violations the network team decided to say "No More". The security group sought budget approval and labor approval in order to deploy an internal sensor network - systems configured to monitor the network with a variety of open source tools designed to "catch the bad guy".

[2] Actual remediation was estimated to take between 20 and 25 people working a solid 2 to 3 weeks dealing with this particular virus.
[3] Symantec has a write up on the virus posted on their website. See: http://securityresponse.symantec.com/avcenter/venc/data/w32.sasser.b.worm.html

## Sensor Net Goal

Deploying a sensor network has a few goals in its implementation. First, putting the network and security teams in the position of being able to monitor, collect, and analyze data on the network. Second, having a suite of tools in place that support automating the analysis and collection of network data - Events of Interest, as they are usually called in the Intrusion Analysis game. Third, there is the business goal of producing *accurate actionable intelligence* on systems that are posing a threat to the heath of the network in order to help guide a limited number of technicians so that their efforts can be directed to where they are needed most.

## Sensor Net Software Components - Gen One

There are a number of components for the first generation of the Sensor Net (discussed in this paper, and initially deployed). First, an installation of Snort 2.1.3 with a rule set that is designed to monitor the local network (not the Internet). Second, SAMBA is installed with a network share that is designed to be attractive to viruses. Third, Tom Listens' LaBrea TarPit is used and it is configured to trap on and report a variety of connection attempts. Fourth, these components are configured to syslog to a central syslog server. That system is configured with a log watcher utility that reports if a system on the network connects to a sensor system.

Complimenting these "on the wire" tools a Cisco 350 Aeronet card was added to the mix, as it is the best tool to support Kismet - a totally passive W-LAN sniffing tool. By incorporating a wireless LAN card capable of functioning in R-MON mode, the network team could make use of the sensor network to monitor the air space around the campus.

This sensor network is described as low interaction – if functions more passively and actively. A high interaction system is planned and described next.

## Sensor Net Software Components - Gen Two

Once the security group has a good handle on how these packages can improve network security, the plan is to move forward with high interaction sensor network based on honeyd (from www.honeyd.org). Honeyd is designed to mimic a real system - and is an ideal way to capture network based worms and actual attackers on the network. However, for the first deployment, it was decided to wait on using honeyd until the teams had a good handle on using Snort, LeBrea, and mining Events of Interest on the syslog server. Further, experience taught the team that many information sources need to be correlated in order to effectively combat viri.

## Gen One Cost Components

Ever mindful of cost, the basic sensor system is a recycled LAB computer. These systems were augmented with more memory, an additional gigabit

Ethernet card, a new high speed and higher capacity hard drive, and a W-LAN card.

| Item | Description | Cost |
|---|---|---|
| 1 | 256 MB memory additional per PC | 600.00 |
| 2 | 80 GB 7200 RPM HD w/ 8 MB buffer | 750.00 |
| 3 | Cicso Aero 350 W-LAN card | 1950.00 |
| 4 | Intel gigabit Ethernet card | 530.00 |
| 5 | Shelves for wiring closets and racks | 540.00 |
| 6 | Legal RedHat ES 3 O.S. (Education price) | 500.00 |
| 6 | Sensor PC - Gateway<br>• P 3 933 MHz<br>• 512 MB RAM (total)<br>• 100 MB NIC (mgmt)<br>• CD and Floppy | FREE |
| | | 4870.00 |

Note that the only component in the configuration that is not reusable is the 256 MB memory module - as more modern computers don't use SD-RAM.

There was also an additional benefit in upgrading hard drives. The security group had several 20 GB hard drives that could be used to replace a hard drive that was gathered up when a PC on campus became compromised - a real benefit, as there was no longer a need to quickly find a PC for the user. These drives were sanitized to DoD standards and a base University standard image was loaded on the drive, ready for use as needed.

## *Sensor Placement*

The network follows the Cisco Core-Distribution-Edge model. At the highest level are redundant high end Cisco routers in a fail over / fault tolerant configuration for on campus routing. There is a single OC3 connection to the Internet.

Systems that are part of the sensor network are placed inside the network. Sensor placement is illustrated in the next figure.

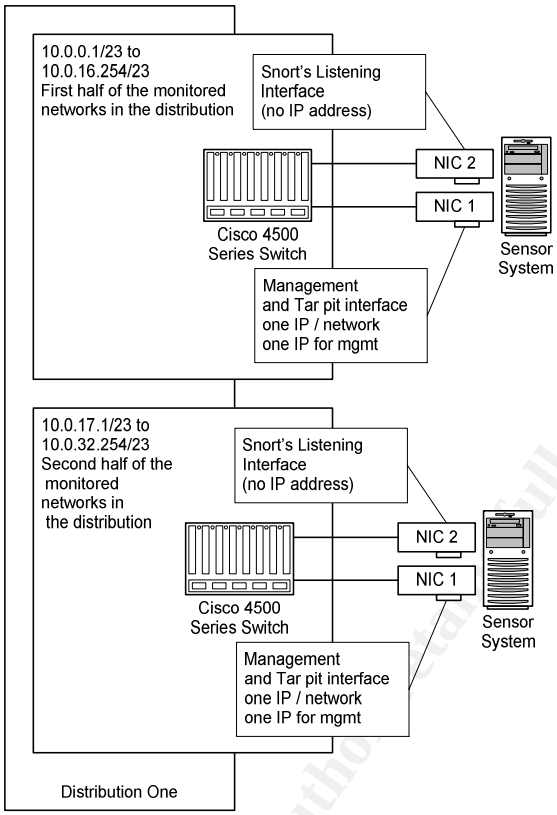


**Figure 1: Sensor Placement**

There are five distribution layers on campus - each serving either a different geographic area or a different business area of the site (illustrated in the next figure). Two sensor systems are placed in four of the distributions. One distribution is for the residence halls - and is outside of the scope for initial deployment. For placing the sensor systems, eight computers placed in strategically chosen network closets and putting two sensors per distribution, spread out enough that almost the entire campus had W-LAN coverage was idea.

About 70% of the network address space is utilized. The network uses a 23 bit subnet mask, so there are 128 addressable networks on campus with 510 possible hosts per subnet[4]. A 70% occupancy rate yields (on average) 18 networks per distribution. With two sensors per distribution, each one is configured to monitors 9 networks.

[4] A 23 bit subnet mask for a Class B address space means that the network portion of the third octet in an IP address has 7 bits for the network portion and one bit that is added to the fourth octet for nine total bits for host addresses. This means that a Class B address space has 128 networks with 510 hosts per network. The reaming two addresses are the network number and the broad cast address.

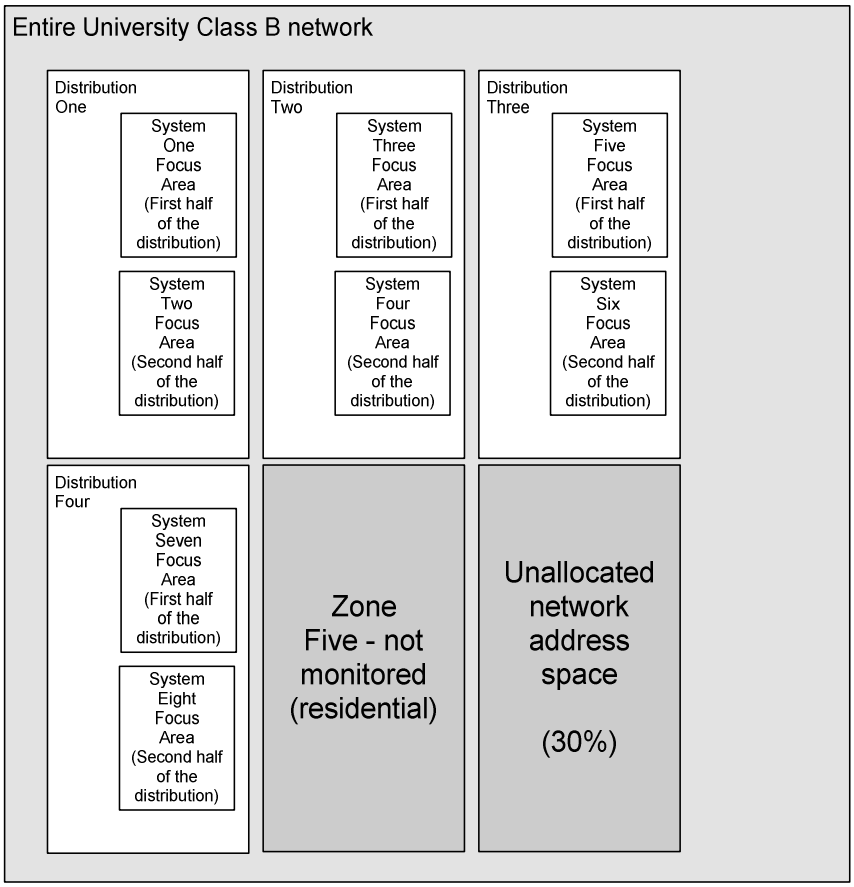


**Figure 2: Sensor Layout**

Example IP Address Distribution and Addressing

- First network starting IP: 10.0.0.1 / 23
  - First network ending IP: 10.0.1.254 / 23
  - First network broadcast address: 10.0.1.255 / 23
- Second network starting IP: 10.0.2.1 / 23
- Third network starting IP: 10.0.4.1 / 23
- Fourth network starting IP: 10.0.6.1 / 23
- Fifth network starting IP: 10.0.8.1 / 23
- Sixth network starting IP: 10.0.10.1 / 23
- Seventh network starting IP: 10.0.12.1 / 23
- Eight network starting IP: 10.0.14.1 / 23
- Ninth network starting IP: 10.0.16.1 / 23
  - Ninth network ending IP: 10.0.17.254 / 23
  - Ninth network broadcast address: 10.0.17.255 / 23

## *Reporting Server*

The reporting server is a system running a centralized syslog server. This system collects log data from the various sensors and runs a log analysis tool which monitors for interesting events.

This server is also an exercise in recycling. It was formerly a high end PC and in order to provide necessary disk space and redundancy four Serial ATA drives and a S-ATA controller were purchased. The drives were configured as dual mirror sets, so that if one drive failed the system could keep on running. Total cost was about 700.00 - and the team had 300 GB of online storage for current and historical data.

## *Sensor Net Tool Configuration*

This section discusses details on the specifics of configuring the various tools used on the sensor network.

### IPTables

In order for the system to be properly managed remotely some rules need to be configured for the local IPTables firewall. Briefly, the rules allow inbound management for SSH from the section of the network where the security group and the network group reside and also allow inbound NetBIOS / SMB traffic. All other inbound traffic is blocked.

Basically, the rules are:

- Default policy for the INPUT chain = DROP
- Default policy for the OUTPUT chain = ACCEPT
- Default policy for the FORWARD chain = ACCEPT
- Allow “related, established” packets in
- Allow SSH traffic from the management network in
- Allow inbound NetBIOS traffic – 137, 138, 139, and 445 traffic in
- Drop all other traffic

During the testing and development phase, a script similar to the one below was used to configure iptables.

```
#!/bin/sh

# stop the service - mainly to make sure that rules are
# being applied as cleanly as possible - you would not
# want to do this on a production system.
/etc/init.d/iptables stop
/etc/init.d/iptables start
IPT=/sbin/iptables
$IPT --flush
$IPT --delete-chain

# setup the default policy
$IPT -P INPUT DROP
$IPT -P OUTPUT ACCEPT
$IPT -P FORWARD ACCEPT

# allow loopback traffic to "talk to me"
```

```
$IPT -A INPUT -i lo -j ACCEPT
$IPT -A OUTPUT -o lo -j ACCEPT

# allow related traffic in
$IPT -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT

# allow inbound SSH from the management network - 10.0.123.0
$IPT -A INPUT -s 10.0.123.0/24 -i eth0 -p tcp -m state --state
NEW --dport 22 -j ACCEPT

# rules to catch DCOM and NetBIOS traffic w/ labrea
# for network 10.0.17.0 - our "testing" network
$IPT -A INPUT -s 10.0.17.0/24 -i eth0 -p tcp -m state --state NEW
--dport 135:139 -j ACCEPT
$IPT -A INPUT -s 10.0.17.0/24 -i eth0 -p tcp -m state --state NEW
--dport 445 -j ACCEPT
$IPT -A INPUT -s 10.0.17.0/24 -p icmp -j ACCEPT

# getting DNS responses is really useful!
$IPT -A INPUT -s 10.0.17.0/24 -i eth0 -p udp --sport 53 -j ACCEPT

$IPT -i eth0 -p tcp -j REJECT --reject-with tcp-reset
$IPT -i eth0 -p udp -j REJECT --reject-with icmp-port-unreachable

$IPT -A INPUT -s 0.0.0.0 -j DROP
# show the output. Note that in real life, we need to do an
# iptables-save - as this is a test/development script
$IPT --list -n
```

## Packet Analysis Tools

A variety of tools were installed for packet analysis - Ethereal, EtherAPE, tcpshow, tcpflow, and tcptrace are but a few examples. These tools can be used to analyze captured data in libpacp format. These tools are discussed in Chapter 5 of "Network Troubleshooting Tools" by Joseph Sloan[5].

Another particularly useful tool from the Honeynet project is privmsg.pl. The perl program shows a list of messages to and from an IRC server in human readable format. This program needed to be modified so it could read IRC traffic on ports other an the default port, 6667.

## Snort 2.1.3 Intrusion Detection System

Snort [6] is a light weight intrusion detection system. Snort is configured to monitor for signatures of known attacks and activity on the network. Each Snort rule has two parts - the first part defines the scope of the rule and the second part defines the specific rule characteristics. For example, the default rule for the CyberKit 2.2 signature (one of the signatures used to detect the Nachi worm) is shown below.

[5] Network Troubleshooting Tools is published by O'Reilly Media.
[6] Snort's website is www.snort.org.

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP PING
CyberKit 2.2 Windows"; itype:8; content:"|AA AA AA AA AA AA AA AA
AA AA AA AA AA AA AA AA|"; depth:32; reference:arachnids,154;
classtype:misc-activity; sid:483; rev:5;)
```

In order to properly configure the sensor network the `$EXTERNAL_NET` and the `$HOME_NET` variables need to be configured properly to limit the scope for the rule base. For the first system on the first distribution on the network these two variables are configured in the `/etc/Snort /Snort .conf` file as follows.

```
var EXTERNAL_NET [10.0.0.0/16]
var HOME_NET [10.0.0.0/23.,10.0.2.0/23,10.0.4.0/24,10.0.6.0/23,
10.0.8.0/23,10.0.10.0/23,10.0.12.0/24,10.0.14.0/23,10.0.16.0/23]
```

As can be seen here (and in the figure below) the external network is configured to be all of the University network - the entire class B address space. The home network is configured to be a *subset* of the entire network space. Also notice the order - Snort will generate an alert if any attack comes from anywhere on the entire network to a group of subnets on the network, *but not the Internet*. This means that Snort will alarm if a system on the University network attacks others on the same network. Why is this important? Since the sensor network is configured to be monitoring for internal intrusions, and since there is a perimeter IDS system in place, we would not want to see alerts twice - nor do we want to consume precious CPU cycles analyzing out of scope traffic. And more importantly the sensor network is designed to monitor for internal intrusions.

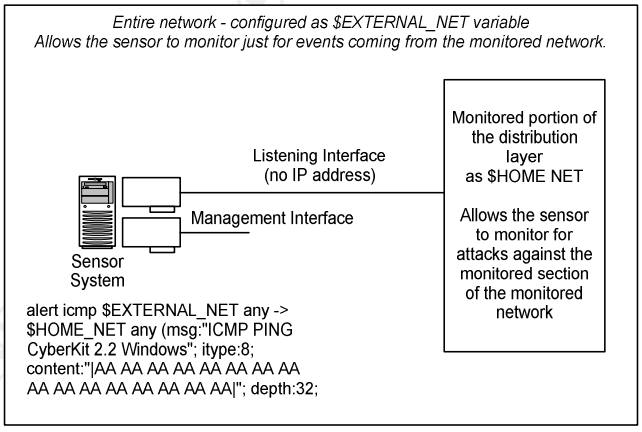


**Figure 3: Snort IDS Configuration w/ Example Rule**

## Analysis Console for Intrusion Databases

The ACID system is a great way to store and visualize data from the Snort IDS[7]. In this environment data for ACID is only kept for the most recent 24 hours. Overnight a program is executed against the ACID database and a report that summarizes traffic is written. After the report finishes "yesterday's" data is purged from the database.

[7] Configuring ACID is well documented at its website: http://acidlab.sourceforge.net/

## LaBrea Tarpit

Quoting from the LaBrea home page at SourceForge:

> "LaBrea takes over unused IP addresses, and creates virtual servers that are attractive to worms, hackers, and other denizens of the Internet. The program answers connection attempts in such a way that the machine at the other end gets "stuck", sometimes for a very long time.".[8]

In this environment with the great importance placed on not interfering with normal network operations additional precautions were used to make sure that LaBrea was configured to only monitor and respond to specific addresses[9]. Specifically, entries were added to the `/usr/local/etc/labrea.conf` configuration file to meet the following requirements:

- LaBrea ignored IP addresses outside of its scope - meaning that some network IP address ranges were excluded from its attention.
- Monitor the NetBIOS ports

```
# exclude the server network segments
10.0.170.0/23 IPI
10.0.174.0/23 IPI
# always tarpit the NetBIOS ports - commonly used by network
worms
135 PMN
137-139 PMN
445 PMN
```

LaBrea uses a command line similar to the one below as its startup:

```
labrea -z -s -b -p 10000 -i eth1 -m 10.0.17.0 --mask
255.255.254.0 --bpf-file /usr/local/etc/lebrea.monitored --log-
to-syslog -v
```

Command line options mean:

- `-z`: no nag (disable the startup warning).
- `-s:` Tells LaBrea to work properly in a switch environment by sending out an ARP request of its own for the same IP.
- `-b`: Log bandwidth (need to monitor bandwidth usage).
- `-p 10000:` The maximum rate for tarpitting connections (10Kbps)
- `-i eth1`: the Ethernet interface to listen on.
- `-n 10.0.17.0`: Specifies that LaBrea is monitoring this specific subnet on the overall network.

[8] See: http://labrea.sourceforge.net/labrea-info.html
[9] The LaBrea is documented more fully at its website: http://labrea.sourceforge.net/labrea-info.html

- `--mask 255.255.254.0`: Specifies the 23 bit subnet mask in dotted decimal format (as opposed to CIDR).
- `--bpf-file /usr/local/etc/lebrea.monitored`: LaBrea uses the IP's in this file are the ones that it is monitoring. The file is in BPF format[10].
- `--log-to-syslog`: generate events to the syslog service.
- `-v`: log with more verbosity (useful when starting to use LaBrea)

Note that in order to use the bpf filter file to function the IPTables firewall must be configured to "DROP" traffic that is inbound to the interface.

## *Defense In Depth Considerations*

The overall architecture has several DiD aspects to it. First are some protections at the perimeter. Second, there are several local access control lists on the sensors themselves. Third, the reporting server only accepts connections from the sensor network and from the management stations. Fourth, the systems are not listed in DNS. Lastly, SSH is used to keep the systems current.

### Perimeter Protection

At the perimeter - the Internet Router - there is a set of Cisco Access Control Lists which *deny inbound connections* to the sensors and the management station. By stopping inbound traffic at the Internet connection an Internet based attacker cannot easily determine that there are systems. Secondly, there is a set of access control lists which prevent outbound traffic coming from the sensor net systems passing through to the Internet. This layer of protection helps in several ways. First, if the software on the system should misbehave it will be stopped at the perimeter. Second, if an attacker on the network should find and compromise the system they cannot use these systems for outbound attacks. Third, the security and network groups cannot connect to the Internet from these systems - an unlikely occurrence, but every bit of protection helps. Fourth - and most importantly - LaBrea can be seen as interrupting service on the network. By protecting other networks off of the campus network from being able to communicate with sensors the University, the sole network provider at this site, is protected from LaBrea interfering with other networks.

### No Domain Name Service (DNS) Listing

The sensor network is not listed in the site's DNS systems, nor do they broadcast any sort of "advertisement" traffic (such as NetBIOS traffic). By not listing these systems in DNS they cannot normally be queried for and found on the network. Also, by not broadcasting advertisements systems would not normally discover them. Therefore if a query is recorded against these systems it is almost certain that the connection attempt is for one of a few reasons: a) accidental (fat fingering), b) network scanning (by a person or an agent), c) an attempt at

[10] One good source of information on BPF syntax is: http://winpcap.polito.it/docs/man/html/group__language.html

compromise by a malicious agent, or d) a legal connection from a few management IP addresses.

## After the Sensor Net

By having these systems in place the security group was successful in dealing with recent worms and system compromises. Below are several examples of how the sensor network is used to combat viruses on the University LAN. Unfortunately, older viruses and their variants keep appearing on the LAN. Not only that, several viruses have similar signatures - exploring shares, rapidly pinging the network, starting IRC agents, making tftp connections, and such. So even though some of these viruses are a year old they are examples of attacks that keep reappearing on the network.

### *Example 1: Fighting MS Blaster Variants*

In order to deal with successive generations' of this particular worm the Snort IDS was configured with extra signatures - lessons learned from handling the Q3 2003 outbreak. These signatures were added to Snort's "`local.rules`" file, and their scope was limited to network under protection. Symantec has an excellent article[11] on detecting this worm and its variants (that keep appearing), which helped to guide the traffic analyses.

- Alert for traffic to port 135 using the flow:to_server,established; keywords. This rule catches basic connection attempts.
- Alert for traffic to port 445 using the flow:to_server,established; keywords. This rule catches basic connection attempts.
- Alert if TFTP traffic is seen (port 69).
- Alert if port 4444 traffic is seen (common backdoor port)

Every hour, a script was run against the Snort alert file. This program was designed to perform traffic analysis and to build up a confidence score for the likelihood that a client was infected. The pseudo code is listed below. Remember that the sensor is inside the network - not at the Internet border - and it's monitoring about 1/10 of the usable network address space. With that in mind the network traffic that the sensor sees is higher resolution than traffic seen at the Internet border.

```
Open log file
     Read alert and collect these characteristics
          source ip and port
          destination ip and port
          signature ID
     If alert is for destination port 135 traffic AND packet
size is 72 or 244 bytes
          increment PORT_135 count for this IP
          add IP to IPLIST
     If alert is for destination port 445
```

[11] Symantec Website: http://securityresponse.symantec.com/avcenter/venc/data/detecting.traffic.due.to.rpc.worms.html

```
            increment PORT_445 count for this IP
            add IP to IPLIST
      If alert is for destination port 135 traffic and DEST IP
OFF NETWORK
            increment PORT_135_OFFNET count for this IP
            add IP to IPLIST
      If alert is for destination port 445 traffic and DEST IP
OFF NETWORK
            increment PORT_445_OFFNET count for this IP
            add IP to IPLIST
      If alert is to or from port 69 and for the UDP protocol
            increment PORT_69 count for this IP
            add IP to IPLIST
      # next - look for traffic to / from the backdoor this is
not web traffic
      If alert is to or from port 4444 and other port is not 80
            increment PORT_4444 count for this IP
            add IP to IPLIST
Close log file
For each IP in IPLIST
      If IP in PORT_69 or PORT_4444
            print "high confidence - TFTP or BackDoor port"
      If IP in PORT_135 and > 60
            print "Port 135 traffic " + count of packets
      If IP in PORT_135_OFFNET entries exist
            print "Port 135 traffic " + count of packets
      If IP in PORT_445 and > 60
            print "Port 445 traffic: Off network" + count of
packets
      If IP in PORT_445_OFFNET entries exist
            print "Port 445 traffic: Off network " + count of
packets
      Perform nmap check - make sure its Windows
      Perform DNS lookup for IP and print results
      Perform NetBIOS lookup - print out information received
      Lookup Support contact information based on network address
- print
End for loop
```

With the data produced from this simple analysis a support person can look through the report. The first clue is the host name. When a system has an "H########" host name it is using an IP address from DHCP. The second clue is any traffic to and from a TFTP server or the backdoor port from non-HTTP (web browser) traffic. If these signatures are found the confidence is high that the system is infected. Third, statistics on port 135 and port 445 traffic can be checked. If there is off network traffic on these ports then confidence is high that the system is infected - the worm attempts to randomly scan hosts on the Internet as part of its propagation method. Lastly, NetBIOS lookup information lets the support person know who is using the computer and if it is part of a domain or workgroup. Coupling this with the support contact information from a locally maintained table and the analyst has a very clear picture of traffic coming from a host.

Below are current Snort signatures that can detect this worm.

- 2192: NETBIOS DCERPC ISystemActivator bind attempt
- 2193: NETBIOS SMB DCERPC ISystemActivator bind attempt
- 2351: NETBIOS DCERPC ISystemActivator path overflow attempt little endian
- 2352: NETBIOS DCERPC ISystemActivator path overflow attempt big endian

## *Example 2: Nachi*

Similar to the Blaster worm discussed above, searching for the Nachi worm used similar techniques with a few changes. The Snort IDS already had a signature - indirectly - for the Nachi worm. The rule "ICMP PING CyberKit 2.2 Windows" identifies the worm correctly in the sense that this alert is triggered from the worm. When an analysis of alerts coupled with other port 135/445/69 traffic as above for the Blaster worm there is a high confidence that a system is infected. The pseudo code is essentially the same with additional counters for the Cyberkit signature and deletion of the port 4444 traffic.

## *Example 3: Sasser (and it's variants)*

The Sasser[12] worm attempts to exploit the vulnerability that is described in Microsoft's bulletin MS04-011[13]. This worm has two telltale signatures - rapid connections in rapid succession from a target to port 445 and, if compromised, a backdoor opens up a remote shell on port 9996.

In order to deal with this worm a different approach was used. The individual distributions were *sampled* - traffic was analyzed to see what the statistical breakdown was for the sample period (arithmetic mean and mode). Next, traffic was sampled again, and for systems which are a) significantly above the mean and b) not destined for known servers prompted attention. Secondly, any traffic to/from port 9996 also figured into the analysis.

Commands used:

Capture traffic destined to port 445:

```
tcpdump -n -w DistSample "dst port 445 and net 128.82.0.0 mask
255.255.0.0"
```

Reduce the data by getting initial TCP SYN packets:

```
tcpdump -n -r DistSample -w DistSampleReduced "tcp[13] =0x02"
```

Pseudo Code for analysis process:

```
for each row of reduced data
     collect unique source IP addresses into a hash
```

[12] Symantec Website: http://securityresponse.symantec.com/avcenter/venc/data/w32.sasser.worm.html
[13] http://www.microsoft.com/technet/security/bulletin/MS04-011.mspx

```
      if source port not listed in hash then add to hash
      if src port list
           search the data portion of the hash
           if src port not found then add to list
end loop
for each IP in the hash
      print the IP
      print the list of source ports
      query for the NetBIOS information
      query for the network contact information from static list
      print details
end loop
```

Output of this program shows a nice list of IP's that are generating huge amounts of traffic. These systems were high on the priority list for remediation and personal attention, which consisted of running a variety of virus removal tools and updating the system with current patches.

After a day or two a Nessus plug-in[14] was available to scan for systems which where vulnerable to the Microsoft vulnerability that Sasser (and its variants) exploited. The same analysis process was repeated - sample, reduce, digest and report. The systems that had a high score were analyzed using Nessus and these plugins.

## *Example 4: Virus-like Compromise - Controlled over IRC*

Occasionally the security staff or the network staff will detect a tremendous amount of unexplained traffic (often using NetFlow). In this case, gigabits of traffic were detected over a very short interval, which usually indicates systems may be doing something that may be out of bounds. A tremendous amount of ICMP traffic was detected. Initial analysis indicated that systems in a newly configured lab (with new PC's that hadn't had service packs applied) where apparently spreading viruses - hundreds of connections were visible on a few systems by running "`netstat -an`". Further analysis revealed that systems inside the network were attacking other systems, and that they were being remote controlled in an automated fashion over IRC. Inbound IRC was stopped on the outer router and outbound IRC was stopped on the outer router - as illustrated in the next figure.

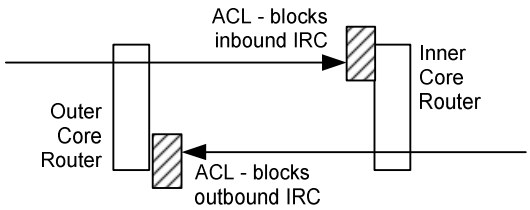


**Figure 4: Example 4: Perimeter Security**

[14] Nessus is a remote vulnerability assessment tool, which does a very good job at finding issues with operating systems. See: http://cgi.nessus.org/plugins/dump.php3?id=12209 and http://cgi.nessus.org/plugins/dump.php3?id=12205

The sensor network played an important role here in detecting what was going on, as the security team decided to disable IRC traffic (port 6667) on and off the network. Data was captured on the various distributions and then analyzed. As data was analyzed, compromised systems were found to be attacking each other in an automated fashion - connecting to shares, exploring NetBIOS information, sending ICMP traffic, and attempting to make connections all over campus. Analysis was similar to what was presented above in example 1 and 3. Even through the controllers were cut off there were still systems "following orders" for a period of time - without having the sensor network it would be difficult to see who was attacking whom.

## Future Directions

Once the first generation sensors were deployed several gaps were identified. These gaps are the basis for future directions with the sensor network. Gaps include DNS query correlation, perimeter Snort event correlation, DHCP address assignment, NetBIOS information, and determining physical location of the machine.

Some solutions are outlined below, and when manpower becomes available then customized scripts can be written to better understand how the environment is being used.

### *DNS Queries*

Often malware will want to look for an IRC server. This has been seen time and time again from analysis at the honeynet project. Under development is a monitoring script that performs these tasks:

- Every minute (out of `cron`) check for the following conditions:
  - Has the named log file rolled over within the last 3 minutes?
  - Has the named file grown to 75% of its max size in the last 3 minutes?
- If the above conditions are met, then:
  - Digest the named query log file, and determine frequently targeted sites.
- Produce a “top 15” report of all sites in descending order by query count for non .com, .edu, .mil, .net, and .gov sites. Frequently an IRC based BOT army is being remote controlled from or through a foreign host. By eliminating sites that are likely to be legitimate, focus is drawn in on sites from foreign countries.
- Produce a “top 100” report of all queried sites.

### *DHCP Addresses*

One of the other problems encountered while dealing with viruses is the ability to know how recently a host came on the network – is it a traveling user with a virus

or a standard desktop PC that has become infected. Along with the DNS queries software will be developed to allow for a quick query of the DHCP logs to determine the machines' MAC address and lease time.

## *NetBIOS Information*

One of the useful things that network and system administrators really need is some idea of where computers "are" on the campus. By taking the address list of top hosts querying DNS and writing out a script file that can be used to do a NetBIOS lookup (with either SaMBa's nmblookup or Windows nbtstat program), various administrators can cut and paste a script from an email and run it on their section of the network to determine valuable NetBIOS information. Example NetBIOS information is below. This information shows us that the computer at IP addres 192.168.1.101 is in the "workgroup" workgroup on a computer named "Bach", the logged on user is "Don", and the machines MAC address. By using workgroup names that relate to physical location on campus the systems can be more easily identified by location. MAC address correlation is usually very important in a dynamic environment where laptop users connect and disconnect frequently. In one instance during the August 2003 virus outbreak mentioned above, the security staff tracked a user who connected to the network 26 different times and was assigned 23 different addresses as they moved all around the campus. The NetBIOS information allowed the machine to be identified.

```
Local Area Connection 2:
Node IpAddress: [192.168.1.101] Scope Id: []

           NetBIOS Remote Machine Name Table

       Name               Type         Status
    ---------------------------------------------
    BACH            <00>  UNIQUE      Registered
    WORKGROUP       <00>  GROUP       Registered
    BACH            <20>  UNIQUE      Registered
    BACH            <03>  UNIQUE      Registered
    WORKGROUP       <1E>  GROUP       Registered
    WORKGROUP       <1D>  UNIQUE      Registered
    ..__MSBROWSE__.<01>  GROUP       Registered
    DON             <03>  UNIQUE      Registered

    MAC Address = 00-0F-EA-32-E3-69
```

# Appendix One: Sensor System Configuration

O.S.: RedHat Enterprise Linux 3.0 - (Academic Price)

Install base operating system with the following guidance.

1. Disk Configuration w/ mount points
    a. /boot - 256 MB (multiple kernels)
    b. / (root)- 2.0 GB
    c. swap - 768 MB (2x main memory)
    d. Remaining disk is allocated to /var, as that's the principle purpose of the system.
2. Set a boot loader password on GRUB
3. Set the firewall to "high", as a different IPtables setup will be used later on.
4. Set the time zone to Eastern time zone
5. Don't install NIS, LDAP, or SMB support here - will install a more current SMB later on, downloaded from the SaMBa website
6. Desktops - X11 & Gnome . Default configuration w/ init Level 3 - we want the system to run in "text mode", not GUI mode.
7. Editors - Install
8. Engineering / Scientific - nothing is selected here
9. Graphical Internet - install - browsers are required occasionally in the field and in the computer room
10. Text Internet – Install ncftp - other utils not realy needed.
11. Office Productivity - nothing
12. Sound / Video - nothing
13. Graphics - nothing
14. Games - nothing (violation of state policy)
15. Server tools - install
16. Web server - nothing – custom apache install later on
17. Mail Server – install – will provide mc/cf files later in support of a "smart relay" option. The system will be sending mail out to network/security staff at regular email addresses and to pagers
18. Windows file server - nothing
19. DNS, news and SQL Database – nothing
20. Network Server – install (nfs support)
21. Development tools – nothing - if possible do **not** install compilers - we will compile software/ make RPM's from a "head" node and distribute to sensor nodes
22. Kernel Development – nothing
23. X Software Development – install
24. Gnome Software Development – install
25. Administration tools – install
26. System tools - install
27. Printing support - install

## Appendix One: Sensor System Configuration

Post Operating System Installation

1. Get current updates from redhat.com
2. Kernel update to get it current including kernel source
3. install all rpm updates for files installed by using the freshen "rpm -Fvh"
4. Install apps to support IDS functions - see http://www.Snort .org/docs/Snort _acid_rh9.pdf - Current Snort , Mysql, apache, zlib, ACID, etc.
    a. Install Apache - this needs to be hardened. For instance, only accept connections from the "management" section of the network (stations where the network and security team sit).
    b. Install MySQL - make sure that the root passwords for the DB server are set and permissions are secured on the data base server.
5. Additional software includes:
    a. p0f
    b. nmap
    c. nessus - need to determine how to schedule or automatically get updates by ... ?
6. TCP/IP specific analysis tools - primarily discussed in "Network Troubleshooting Tools" by Joseph Sloan[15].
    a. tcpdump
    b. tcpshow
    c. tcpflow
    d. tcp-reduce
    e. tcpshow
    f. tcpslice
    g. tcptrace
    h. sanitize script set
7. Also needed - the "privmesg.pl" IRC extractor script on the honeynet.org site[16].
8. Ethereal - update it to patch some vulnerabilities
9. Etherape - need it
    h. ssh - harden configuration - only accept SSH 2 connections, install a legal support banner, prevent root logins, and log to "secure".
10. Other misc support
    a. cvs client
    c. what ever it takes to SCP stuff to the boxes and update Snort rules
    d. perl - current (stable)
    e. iptables – Security Group will configure Iptables logging to syslog
    g. LaBrea 2
    h. zip / unzip, rar, unrar support
    j. SaMBa - we need to setup a "ghost" share that perhaps points to

[15] Available from O'Reilly Publishing. http://www.oreilly.com/catalog/nettroubletools/
[16] Available from: http://www.honeynet.org/tools/index.html

/dev/null so that we can allow a virus to "attempt" to connect and THEN syslog the data
l. Tripwire - a nightly analysis ... in a cron job

11. System hardening - check out the various GCUX practicals, including: http://www.giac.org/practical/GCUX/Don_Murdoch_GCUX.pdf

## References

Honeynet Project. “Know Your Enemy: Learning about Security Threats (2nd Edition)”. Pearson Education, 2004. Entire text.

Lorgor, "LaBrea: "Sticky" Honeypot and IDS" URL: http://labrea.sourceforge.net/labrea-info.html (Aug 11, 2004)

Microsoft Corp. "What You Should Know About the Blaster Worm". Jan 22, 2004. URL: http://www.microsoft.com/security/incident/blast.mspx (Aug 7, 2004)

Microsoft Corp. "Microsoft Security Bulletin MS04-011 Security Update for Microsoft Windows (835732)" Aug 10, 2004. URL: http://www.microsoft.com/technet/security/bulletin/MS04-011.mspx (Aug 23, 2004).

Sloan, Joseph. "Network Troubleshooting Tools". O'Reilly Media, Sebastopol, CA, 2001. Ch 5, 10.

Symantec Corp. "W32.Blaster.Worm". Aug 13, 2003. URL: http://securityresponse.symantec.com/avcenter/venc/data/w32.blaster.worm.html (Aug 8, 2004)

Symantec Corp. "W32.Sasser.B.Worm". Jul 27, 2004. URL: http://securityresponse.symantec.com/avcenter/venc/data/w32.sasser.b.worm.html4 (Aug 8, 2004)

Symantec Corp. "Detecting network traffic that may be due to RPC worms". Sep 12, 2003. URL: http://securityresponse.symantec.com/avcenter/venc/data/detecting.traffic.due.to.rpc.worms.html (Aug 8, 2004)

Tenable Network Security, " Microsoft Hotfix for KB835732 (SMB check)", May 6, 2004. URL: http://cgi.nessus.org/plugins/dump.php3?id=12209 (Aug 23, 2004)

Torino, Politecnico. "Filtering expression syntax" URL: http://winpcap.polito.it/docs/man/html/group__language.html (Aug 11, 2004)