GIAC Intrusion Detection

Level II Practical Assignment

SANS Parliament Hill

August 21-24, 2000

Curtis L. Blais CCNA

Table of Contents

Introduction

This document is to meet the practical requirements for the level two GIAC Intrusion Detection track from the August 2000 Parliament Hill SANS conference.

The log files herein contained require some explanation. They have been acquired from a Department of National Defense (DND) Contractor's network. The site was chosen for two specific reasons:

1) Because they are a DND contractor they would be prone to probes and attacks
2) Because they were willing to share log files and information with me as I worked on this practical. Special thanks to Kurt S. for all his help and work getting me log files and answering all my questions as to what he had set up, version numbers and the like while still doing his regular work - much appreciated.

The log files are of the IPChains format from a RedHat V6.0 machine that is well maintained (described in greater detail in the next section). As I have come to realize while performing this analysis, IPChains appears to be somewhat limited in its granularity for logging. This may limit the kind of detects one can perform with these logs, however it has shown a number of interesting incidents from some rather interesting locations.

The log files have been compiled for nearly two months, from July 1,2000 to August 31,2000 inclusive with the exceptions of August 16th and August 22nd when there were issues with the backup procedure for the log files.

I have made one small addition to the outline given for this practical which includes multiple logs for the same kind of detect. Each of these logs is labeled as a separate incident in the case the I need to refer to one of the incidents specifically.

Also, I would specifically like to mention that before attending the SANS conference in Ottawa, I had not been actively involved in collecting or analyzing Intrusion data or logs; and further have limited experience in the system administration side of Unix systems or Intrusion Detection Systems (IDS) for that matter. The reason I include this is for those of you who will read this and are in the same situation I was in. It has taken a tremendous amount of work and study from the material presented at the SANS conference for me to become somewhat familiar with the IDS terms and the attacks that seem to be common for a majority of the people in attendance at the conference. Don't give up - you can get through this stuff.

Although in-depth knowledge of Linux/Unix is required to make you credible in the IDS field for the design and implementation of sensors and understanding many of the hacker tools, it is not (assuming this practical passes) required for you to understand the concepts and methodologies for Intrusion Detection. My numerous years of looking at Sniffer traces were a great help - not to mention the Cisco ACL experience that I have. Use the Internet for your research, study the material, and use tools you are familiar with (i.e. - Microsoft Access was great for re-sorting log files in different ways).

All of the suspect IP addresses have been looked up using the following HTML WHOIS resources:

North America ->	http://www.arin.net/whois/index.html
Europe ->	http://www.ripe.net/cgi-bin/whois
Asia ->	http://www.apnic.net/

The names listed with the incidents are the blocks of addresses that are registered with the appropriate NIC.

IPCHAINS Described

The following description of IP chains was copied from the listed URL which describes, rather well, what IPCHAINS are and how they work. Please check out this URL for more details on the workings of IPCHAINS (http://www.bb-zone.com/FWHowTo/chapter1.html).

> A firewall chain is nothing more than a set of rules which are used to determine the course of action for a packet as it is matched against each rule. The rules fall in a certain order. When a rule matches a packet, the target of the rule determines what happens next. If the packet doesn't match a rule, the next rule in the chain will be followed. If the end of the chain is reached, the default policy or the default target is taken in order to process the packet.
>
> By default there are three types of predefined chains:
>
> **Input**
> Rules in this chain regulate the acceptance of incoming IP packets. All packets entering via one of the local network interfaces is checked against the input rules. When no matching rule is found, the default policy for the input chain is used.
>
> **Output**
> These rules define the permissions for sending IP packets. All packets that are ready to be sent via one of the local network interfaces are checked against the rules of the output chain. When no matching rule is found, the default policy for the output firewall is used.
>
> **Forward**
> These rules define the permissions for forwarding IP packets. All packets sent by a remote host with another remote host as the destination are checked against the forwarding chain. Again the filter defers to a default policy when no matching rule is found.

IPCHAINS Syntax

A description of how the filters are constructed and what the options mean will also be helpful here. I found this description at http://amazon.oreilly.com/catalog/linag2/ch09.html#9.7 . It does a great job of describing the IPCHAINS filters.

> **9.7.2 ipchains Command Syntax**
>
> The ipchains command syntax is straightforward. We'll now look at the most important of those. The general syntax of most ipchains commands is:
>
> **ipchains command rule-specification options**
>
> **9.7.2.1 Commands**
>
> There are a number of ways we can manipulate rules and rulesets with the ipchains command. Those relevant to IP firewalling are:

-A chain

Append one or more rules to the end of the nominated chain. If a hostname is supplied as either source or destination and it resolves to more than one IP address, a rule will be added for each address.

-I chain rulenum

Insert one or more rules to the start of the nominated chain. Again, if a hostname is supplied in the rule specification, a rule will be added for each of the addresses it resolves to.

-D chain

Delete one or more rules from the specified chain that matches the rule specification.

-D chain rulenum

Delete the rule residing at position rulenum in the specified chain. Rule positions start at one for the first rule in the chain.

-R chain rulenum

Replace the rule residing at position rulenum in the specific chain with the supplied rule specification.

-C chain

Check the datagram described by the rule specification against the specific chain. This command will return a message describing how the datagram was processed by the chain. This is very useful for testing your firewall configuration and we look at it in detail a little later.

-L [chain]

List the rules of the specified chain, or for all chains if no chain is specified.

-F [chain]

Flush the rules of the specified chain, or for all chains if no chain is specified.

-Z [chain]

Zero the datagram and byte counters for all rules of the specified chain, or for all chains if no chain is specified.

-N chain

Create a new chain with the specified name. A chain of the same name must not already exist. This is how user-defined chains are created.

-X [chain]

Delete the specified user-defined chain, or all user-defined chains if no chain is specified. For this command to be successful, there must be no references to the specified chain from any other rules chain.

-P chain policy

Set the default policy of the specified chain to the specified policy. Valid firewalling policies are ACCEPT, DENY, REJECT, REDIR, or RETURN. ACCEPT, DENY and REJECT have the same meanings as for the tradition IP firewall implementation. REDIR specifies that the datagram should be transparently redirected to a port on the firewall host. The RETURN target causes the IP firewall code to return to the Firewall Chain that called the one containing this rule, and continue starting at the rule after the calling rule.

9.7.2.2 Rule specification parameters

A number of ipchains parameters create a rule specification by determining what types of packets match. If any of these parameters is omitted from a rule specification, its default is assumed.

-p [!]protocol

Specifies the protocol of the datagram that will match this rule. Valid protocol names are tcp, udp, icmp, or all. You may also specify a protocol number here to match other protocols. For example, you might use 4 to match the ipip encapsulation protocol. If the ! is supplied, the rule is negated and the datagram will match any protocol other than the protocol specified. If this parameter isn't supplied, it will default to all.

-s [!]address[/mask] [!] [port]

Specifies the source address and port of the datagram that will match this rule. The address may be supplied as a hostname, a network name, or an IP address. The optional mask is the netmask to use and may be supplied either in the traditional form (e.g., /255.255.255.0) or the in the modern form (e.g., /24). The optional port specifies the TCP or UDP port, or the ICMP datagram type that will match. You may supply a port specification only if you've supplied the -p parameter with one of the tcp, udp or icmp protocols. Ports may be specified as a range by specifying the upper and lower limits of the range with a colon as a delimiter. For example, 20:25 described all of the ports numbered from 20 up to and including 25. Again the ! character may be used to negate the values.

-d [!]address[/mask] [!] [port]

Specifies the destination address and port of the datagram that will match this rule. The coding of this parameter is the same as that of the -s parameter.

-j target

Specifies the action to take when this rule matches. You can think of this parameter as meaning "jump to." Valid targets are ACCEPT, DENY, REJECT, REDIR, and RETURN. We described the meanings of each of these earlier. However, you may also specify the name of a user-defined chain where processing will continue. If this parameter is omitted, no action is taken on matching rule datagrams at all, other than to update the datagram and byte counters.

-i [!]interface-name

Specifies the interface on which the datagram was received, or is to be transmitted. Again the ! inverts the result of the match. If the interface name ends with "+" then any interface that begins with the supplied string will match. For example, -i ppp+ would match any PPP network device and -i ! eth+ would match all interfaces except Ethernet devices.

[!] -f

Specifies that this rule applies to everything but the first fragment of a fragmented datagram.

9.7.2.3 Options

The following ipchains options are more general in nature. Some of them control rather esoteric features of the IP chains software.

-b

Causes the command to generate two rules. One rule matches the parameters supplied and the other rule added matches the corresponding parameters in the reverse direction.

-v

Causes ipchains to be verbose in its output. It will supply more information.

-n

Causes ipchains to display IP address and ports as numbers without attempting to resolve them to their corresponding names.

-l

Enables kernel logging of matching datagrams. Any datagram that matches the rule will be logged by the kernel using its printk() function, which is usually handled by the sysklogd program and written to a log file. This is useful for making unusual datagrams visible.

-o[maxsize]

Causes the IP chains software to copy any datagrams matching the rule to the userspace "netlink" device. The maxsize argument limits the number of bytes from each datagram that are passed to the netlink device. This option is of most use to software developers, but may be exploited by software packages in the future.

-m markvalue

Causes matching datagrams to be marked with a value. Mark values are unsigned 32-bit numbers. In existing implementations this does nothing, but at some point in the future it may determine how the datagram is handled by other software such as the routing code. If a markvalue begins with a + or -, the value is added or substracted from the existing markvalue.

-t andmask xormask

Enables you to manipulate the "type of service" bits in the IP header of any datagram that matches this rule. The type of service bits are used by intelligent routers to prioritize datagrams before forwarding them. The Linux routing software is capable of this sort prioritization. The andmask and xormask represent bit masks that will be logically ANDed and ORed with the type of service bits of the datagram respectively. This is an advanced feature that is discussed in more detail in the IPCHAINS-HOWTO.

-x

Causes any numbers in the ipchains output to be expanded to their exact values with no rounding.

-y

Causes the rule to match any TCP datagram with the SYN bit set and the ACK and FIN bits clear. This is used to filter TCP connection requests.

Production Filters

Here are the filters used on the Network where the Detects have been recorded

```
IPCH="/sbin/ipchains"
IPMASQADM=/usr/sbin/ipmasqadm

ipfilter_firewall_cfg () {

# $IPCH is a variable that is set at the top to /sbin/ipchains
# this is so that it is easy to globaly change
# the program that the rules use.
# The same goes for $IPMASQADM it calls /usr/sbin/ipmasqadm
# ipmasqadm is the program that takes care of port forwarding

# Flush the input, forward and output chains
$IPCH -F input
$IPCH -F output
$IPCH -F forward
```

```
# Set the default policies
$IPCH -P input DENY
$IPCH -P forward DENY
$IPCH -P output ACCEPT

### PORT REDIRection stuff ###

# send syslog to gnat
$IPMASQADM portfw -a -P udp -L Good.Net.139.70 514 -R 192.168.4.1 514

# send http to weasel
$IPMASQADM portfw -a -P tcp -L Good.Net.139.71 80 -R 192.168.3.2 80

# send pop3 to weasel
$IPMASQADM portfw -a -P tcp -L Good.Net.139.71 110 -R 192.168.3.2 110

# send smtp to weasel
$IPMASQADM portfw -a -P tcp -L Good.Net.139.71 25 -R 192.168.3.2 25

# send incoming smtp to ferret
$IPMASQADM portfw -a -P tcp -L Good.Net.139.72 25 -R 192.168.3.10 25

# send incoming http to ferret
$IPMASQADM portfw -a -P tcp -L Good.Net.139.72 80 -R 192.168.3.10 80

# send all incmoing ssh on grizzly's outside to gnat then to scorpian
# this is for vpn
#$IPMASQADM portfw -a -P tcp -L Good.Net.139.70 22 -R 192.168.4.1 22

# don't think will need
# do not enable this
# send mail from psn to exchange
##$IPMASQADM portfw -a -P tcp -L 192.168.3.1 25 -R 192.168.4.1 25

# Rules for the bait machine
# will put more of these in later
#$IPMASQADM portfw -a -P tcp -L Some.Net.4.100 23 -R 192.168.5.1 230

### End port redirection ###

# Yes I want all fragments denied
```

```
# Yes I know it will break stuff
# deny all fragments the -f flag means fragments
$IPCH -A input -s 0.0.0.0/0.0.0.0 -d 0.0.0.0/0.0.0.0 -j DENY -f -l
$IPCH -A forward -s 0.0.0.0/0.0.0.0 -d 0.0.0.0/0.0.0.0 -j DENY -f -l
$IPCH -A output -s 0.0.0.0/0.0.0.0 -d 0.0.0.0/0.0.0.0 -j DENY -f -l

# anti spoof rules
# these are only rules for eth0
$IPCH -A input -s Good.Net.139.70/255.255.255.255 -d 0.0.0.0/0.0.0.0 -i eth0 -j DENY -l
$IPCH -A input -s Good.Net.139.71/255.255.255.255 -d 0.0.0.0/0.0.0.0 -i eth0 -j DENY -l
$IPCH -A input -s Good.Net.139.72/255.255.255.255 -d 0.0.0.0/0.0.0.0 -i eth0 -j DENY -l
$IPCH -A input -s Good.Net.139.73/255.255.255.255 -d 0.0.0.0/0.0.0.0 -i eth0 -j DENY -l

#$IPCH -A input -s Some.Net.4.100/255.255.255.255 -d 0.0.0.0/0.0.0.0 -i eth0 -j DENY -l
#$IPCH -A input -s Some.Net.4.105/255.255.255.255 -d 0.0.0.0/0.0.0.0 -i eth0 -j DENY -l
#$IPCH -A input -s Some.Net.4.106/255.255.255.255 -d 0.0.0.0/0.0.0.0 -i eth0 -j DENY -l
#$IPCH -A input -s 192.168.1.0/255.255.255.0 -d 0.0.0.0/0.0.0.0 -i eth0 -j DENY -l
#$IPCH -A input -s 192.168.2.0/255.255.255.0 -d 0.0.0.0/0.0.0.0 -i eth0 -j DENY -l
#$IPCH -A input -s 192.168.3.0/255.255.255.0 -d 0.0.0.0/0.0.0.0 -i eth0 -j DENY -l
#$IPCH -A input -s 192.168.4.0/255.255.255.0 -d 0.0.0.0/0.0.0.0 -i eth0 -j DENY -l

# route from the inside to the psn
# that's route not masq
$IPCH -A forward -s 192.168.4.0/255.255.255.0 -d 192.168.3.0/255.255.255.0 -j ACCEPT
$IPCH -A forward -s 192.168.3.0/255.255.255.0 -d 192.168.4.0/255.255.255.0 -j ACCEPT


# Set up masquerading timout values
# these are the defaults values that come with LRP.
#$IPCH -M -S 14400 0 0

# ip masqurading for outside
$IPCH -A forward -s 192.168.3.0/255.255.255.0 -d 0.0.0.0/0.0.0.0 -i eth0 -j MASQ
$IPCH -A forward -s 192.168.4.0/255.255.255.0 -d 0.0.0.0/0.0.0.0 -i eth0 -j MASQ

# NOTE: commented out cuz we have no eth3
#$IPCH -A forward -s 192.168.3.0/255.255.255.0 -d 0.0.0.0/0.0.0.0 -i eth3 -j MASQ
#$IPCH -A forward -s 192.168.4.0/255.255.255.0 -d 0.0.0.0/0.0.0.0 -i eth3 -j MASQ

# permit incoming DNS from T****
$IPCH -A input -s 199.185.220.36/255.255.255.255 53 -d 0.0.0.0/0.0.0.0 -p TCP -j ACCEPT
$IPCH -A input -s 199.185.220.55/255.255.255.255 53 -d 0.0.0.0/0.0.0.0 -p TCP -j ACCEPT
```

```
$IPCH -A input -s 199.185.220.36/255.255.255.255 53 -d 0.0.0.0/0.0.0.0 -p UDP -j ACCEPT
$IPCH -A input -s 199.185.220.55/255.255.255.255 53 -d 0.0.0.0/0.0.0.0 -p UDP -j ACCEPT

$IPCH -A input -s 198.80.55.1/255.255.255.255 53 -d 0.0.0.0/0.0.0.0 -p TCP -j ACCEPT
$IPCH -A input -s 198.161.156.1/255.255.255.255 53 -d 0.0.0.0/0.0.0.0 -p TCP -j ACCEPT
$IPCH -A input -s 198.80.55.1/255.255.255.255 53 -d 0.0.0.0/0.0.0.0 -p UDP -j ACCEPT
$IPCH -A input -s 198.161.156.1/255.255.255.255 53 -d 0.0.0.0/0.0.0.0 -p UDP -j ACCEPT

# INCOMING SERVICES

# permit incoming syslog from ftp to gnat
$IPCH -A input -s Good.Net.139.74/255.255.255.255 -d Good.Net.139.70/255.255.255.255 514 -p UDP -j ACCEPT

# permit incoming http to weasel
$IPCH -A input -s 0.0.0.0/0.0.0.0 -d Good.Net.139.71/255.255.255.255 80 -p TCP -j ACCEPT

# permit incoming pop3 to weasel
$IPCH -A input -s 0.0.0.0/0.0.0.0 -d Good.Net.139.71/255.255.255.255 110 -p TCP -j ACCEPT

# permit incoming smtp to weasel
$IPCH -A input -s 0.0.0.0/0.0.0.0 -d Good.Net.139.71/255.255.255.255 25 -p TCP -j ACCEPT

# permit incoming smtp to ferret
$IPCH -A input -s 0.0.0.0/0.0.0.0 -d Good.Net.139.72/255.255.255.255 25 -p TCP -j ACCEPT

# permit incoming http to ferret
$IPCH -A input -s 0.0.0.0/0.0.0.0 -d Good.Net.139.72/255.255.255.255 80 -p TCP -j ACCEPT

# permit incoming vpn
#$IPCH -A input -s 0.0.0.0/0.0.0.0 -d Good.Net.139.70/255.255.255.255 22 -p TCP -j ACCEPT

# permit incoming ESTABLISHED

# permit incoming established on t**** adsl
$IPCH -A input -s 0.0.0.0/0.0.0.0 -d Good.Net.139.70/255.255.255.255 -p TCP -j ACCEPT ! -y

# permit incoming established to weasel
$IPCH -A input -s 0.0.0.0/0.0.0.0 -d Good.Net.139.71/255.255.255.255 -p TCP -j ACCEPT ! -y

# permit incoming established to ferret
$IPCH -A input -s 0.0.0.0/0.0.0.0 -d Good.Net.139.72/255.255.255.255 -p TCP -j ACCEPT ! -y
```

```
# permit incoming

# rules for psn
# permit incoming dns from weasel
# to the 4.0 subnet
$IPCH -A input -s 192.168.3.2/255.255.255.255 53 -d 192.168.4.0/255.255.255.0 -p TCP -j ACCEPT
$IPCH -A input -s 192.168.3.2/255.255.255.255 53 -d 192.168.4.0/255.255.255.0 -p UDP -j ACCEPT

# to the 2.0 subnet
$IPCH -A input -s 192.168.3.2/255.255.255.255 53 -d 192.168.2.0/255.255.255.0 -p TCP -j ACCEPT
$IPCH -A input -s 192.168.3.2/255.255.255.255 53 -d 192.168.2.0/255.255.255.0 -p UDP -j ACCEPT

# ALLOW established to 4.0 from the psn
$IPCH -A input -s 192.168.3.0/255.255.255.0 -d 192.168.4.0/255.255.255.0 -p TCP -j ACCEPT ! -y

# ALLOW established to 2.0 from the psn
$IPCH -A input -s 192.168.3.0/255.255.255.0 -d 192.168.2.0/255.255.255.0 -p TCP -j ACCEPT ! -y

# permit everything incoming from the psn
#$IPCH -A input -s 192.168.3.0/255.255.255.0 -d 192.168.4.0/255.255.255.0 -p TCP -j ACCEPT -l

# permit incoming established from psn to inside
$IPCH -A input -s 192.168.3.0/255.255.255.0 -d 192.168.4.0/255.255.255.0 -p TCP -i eth2  -j ACCEPT ! -y

# FTP allow incoming ftpdata
# cringe this looks ugly
# and scarry
$IPCH -A input -s 0.0.0.0/0.0.0.0 20 -d 0.0.0.0/0.0.0.0 -p TCP -j ACCEPT

### If you want smb filters or to filter
### any other outgoing trafic stick it here.

# permit outgoing
$IPCH -A input -s 0.0.0.0/0.0.0.0 -d 0.0.0.0/0.0.0.0 -i eth1 -j ACCEPT
$IPCH -A input -s 0.0.0.0/0.0.0.0 -d 0.0.0.0/0.0.0.0 -i eth2 -j ACCEPT
## BAD BAD ## $IPCH -A input -s 0.0.0.0/0.0.0.0 -d 0.0.0.0/0.0.0.0 -i eth3 -j ACCEPT

# permit all ping execpt redirects
# i don't know if I need the forward one but...
# i want to make sure we are not touching redirects
# because the are ugly.
$IPCH -A forward -j DENY -p icmp --icmp-type redirect -l
```

```
$IPCH -A input -j DENY -p icmp --icmp-type redirect -l
$IPCH -A input -s 0.0.0.0/0.0.0.0 -d 0.0.0.0/0.0.0.0 -p ICMP -j ACCEPT

# deny all incoming on eth0
$IPCH -A input -s 0.0.0.0/0.0.0.0 -d 0.0.0.0/0.0.0.0 -i eth0 -j DENY -l

}
```

Assignment 1 - Network Detects

Detect 1 - Sub Seven Version 1 & Version 2 Trojan Scans

```
Incident 1 - @ Home Network - Redwood CA USA
Jul 13 1:27:40 wren kernel: Packet log: input DENY eth0 PROTO=6 24.18.114.56:4574 Good.Net.139.70:1243 L=48 S=0x00 I=59804 F=0x4000 T=114 SYN (#34)
Jul 13 1:27:40 wren kernel: Packet log: input DENY eth0 PROTO=6 24.18.114.56:4575 Good.Net.139.71:1243 L=48 S=0x00 I=60060 F=0x4000 T=114 SYN (#34)
Jul 13 1:27:40 wren kernel: Packet log: input DENY eth0 PROTO=6 24.18.114.56:4576 Good.Net.139.72:1243 L=48 S=0x00 I=60316 F=0x4000 T=114 SYN (#34)
Incident 2 = France Telcom
Jul 31 15:40:10 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 193.250.123.31:1647 Good.Net.139.70:1243 L=48 S=0x00 I=51760 F=0x4000 T=110 SYN (#35)
Jul 31 15:40:10 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 193.250.123.31:1648 Good.Net.139.71:1243 L=48 S=0x00 I=52272 F=0x4000 T=110 SYN (#35)
Jul 31 15:40:10 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 193.250.123.31:1649 Good.Net.139.72:1243 L=48 S=0x00 I=52528 F=0x4000 T=110 SYN (#35)
Incident 3 - PSINET - VA USA
Aug 9 3:24:37 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 38.29.61.236:2801 Good.Net.139.70:27374 L=48 S=0x00 I=20515 F=0x4000 T=114 SYN (#35)
Aug 9 3:24:40 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 38.29.61.236:2801 Good.Net.139.70:27374 L=48 S=0x00 I=49443 F=0x4000 T=114 SYN (#35)
Aug 9 3:24:40 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 38.29.61.236:2802 Good.Net.139.71:27374 L=48 S=0x00 I=46627 F=0x4000 T=114 SYN (#35)
Aug 9 3:24:44 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 38.29.61.236:2803 Good.Net.139.72:27374 L=48 S=0x00 I=21027 F=0x4000 T=114 SYN (#35)
Incident 4 - Bresnan Communications NY USA
Aug 12 20:05:37 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 24.213.34.165:3076 Good.Net.139.70:27374 L=48 S=0x00 I=35678 F=0x4000 T=116 SYN (#35)
Aug 12 20:05:37 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 24.213.34.165:3077 Good.Net.139.71:27374 L=48 S=0x00 I=35934 F=0x4000 T=116 SYN (#35)
Aug 12 20:05:38 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 24.213.34.165:3076 Good.Net.139.70:27374 L=48 S=0x00 I=41566 F=0x4000 T=116 SYN (#35)
Aug 12 20:05:38 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 24.213.34.165:3077 Good.Net.139.71:27374 L=48 S=0x00 I=40030 F=0x4000 T=116 SYN (#35)
Aug 12 20:05:38 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 24.213.34.165:3078 Good.Net.139.72:27374 L=48 S=0x00 I=36190 F=0x4000 T=116 SYN (#35)
Aug 12 20:05:38 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 24.213.34.165:3078 Good.Net.139.72:27374 L=48 S=0x00 I=41822 F=0x4000 T=116 SYN (#35)
Aug 12 20:05:44 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 24.213.34.165:3076 Good.Net.139.70:27374 L=48 S=0x00 I=45918 F=0x4000 T=116 SYN (#35)
Aug 12 20:05:44 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 24.213.34.165:3077 Good.Net.139.71:27374 L=48 S=0x00 I=44382 F=0x4000 T=116 SYN (#35)
Aug 12 20:05:44 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 24.213.34.165:3078 Good.Net.139.72:27374 L=48 S=0x00 I=46174 F=0x4000 T=116 SYN (#35)
Aug 12 20:05:56 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 24.213.34.165:3076 Good.Net.139.70:27374 L=48 S=0x00 I=4447 F=0x4000 T=116 SYN (#35)
Aug 12 20:05:56 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 24.213.34.165:3077 Good.Net.139.71:27374 L=48 S=0x00 I=3167 F=0x4000 T=116 SYN (#35)
Aug 12 20:05:56 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 24.213.34.165:3078 Good.Net.139.72:27374 L=48 S=0x00 I=4703 F=0x4000 T=116 SYN (#35)
Incident 5 - @ Home Toronto CANADA
Aug 12 20:23:25 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 24.43.136.223:1387 Good.Net.139.70:27374 L=44 S=0x00 I=4997 F=0x4000 T=17 SYN (#35)
Aug 12 20:23:25 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 24.43.136.223:1388 Good.Net.139.71:27374 L=44 S=0x00 I=5253 F=0x4000 T=17 SYN (#35)
Aug 12 20:23:26 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 24.43.136.223:1389 Good.Net.139.72:27374 L=44 S=0x00 I=5509 F=0x4000 T=17 SYN (#35)
Aug 12 20:23:27 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 24.43.136.223:1387 Good.Net.139.70:27374 L=44 S=0x00 I=18565 F=0x4000 T=17 SYN (#35)
Aug 12 20:23:27 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 24.43.136.223:1388 Good.Net.139.71:27374 L=44 S=0x00 I=18821 F=0x4000 T=17 SYN (#35)
Aug 12 20:23:27 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 24.43.136.223:1389 Good.Net.139.72:27374 L=44 S=0x00 I=19077 F=0x4000 T=17 SYN (#35)
Aug 12 20:23:34 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 24.43.136.223:1387 Good.Net.139.70:27374 L=44 S=0x00 I=33669 F=0x4000 T=17 SYN (#35)
Aug 12 20:23:34 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 24.43.136.223:1388 Good.Net.139.71:27374 L=44 S=0x00 I=33925 F=0x4000 T=17 SYN (#35)
Aug 12 20:23:34 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 24.43.136.223:1389 Good.Net.139.72:27374 L=44 S=0x00 I=34181 F=0x4000 T=17 SYN (#35)
Aug 12 20:23:46 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 24.43.136.223:1387 Good.Net.139.70:27374 L=44 S=0x00 I=58245 F=0x4000 T=17 SYN (#35)
Aug 12 20:23:46 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 24.43.136.223:1388 Good.Net.139.71:27374 L=44 S=0x00 I=58501 F=0x4000 T=17 SYN (#35)
Aug 12 20:23:46 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 24.43.136.223:1389 Good.Net.139.72:27374 L=44 S=0x00 I=58757 F=0x4000 T=17 SYN (#35)
```

Incident 6- @ Home Toronto CANADA

```
Aug 12 20:17:14 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 24.43.21.4:1382 Good.Net.139.70:27374 L=48 S=0x00 I=383 F=0x4000 T=113 SYN (#35)
Aug 12 20:17:14 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 24.43.21.4:1383 Good.Net.139.71:27374 L=48 S=0x00 I=639 F=0x4000 T=113 SYN (#35)
Aug 12 20:17:14 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 24.43.21.4:1384 Good.Net.139.72:27374 L=48 S=0x00 I=895 F=0x4000 T=113 SYN (#35)
Aug 12 20:17:16 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 24.43.21.4:1383 Good.Net.139.71:27374 L=48 S=0x00 I=3967 F=0x4000 T=113 SYN (#35)
Aug 12 20:17:17 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 24.43.21.4:1382 Good.Net.139.70:27374 L=48 S=0x00 I=6015 F=0x4000 T=113 SYN (#35)
Aug 12 20:17:17 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 24.43.21.4:1384 Good.Net.139.72:27374 L=48 S=0x00 I=6271 F=0x4000 T=113 SYN (#35)
Aug 12 20:17:22 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 24.43.21.4:1383 Good.Net.139.71:27374 L=48 S=0x00 I=9599 F=0x4000 T=113 SYN (#35)
Aug 12 20:17:23 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 24.43.21.4:1382 Good.Net.139.70:27374 L=48 S=0x00 I=11391 F=0x4000 T=113 SYN (#35)
Aug 12 20:17:23 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 24.43.21.4:1384 Good.Net.139.72:27374 L=48 S=0x00 I=11647 F=0x4000 T=113 SYN (#35)
Aug 12 20:17:34 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 24.43.21.4:1383 Good.Net.139.71:27374 L=48 S=0x00 I=33407 F=0x4000 T=113 SYN (#35)
Aug 12 20:17:35 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 24.43.21.4:1382 Good.Net.139.70:27374 L=48 S=0x00 I=35199 F=0x4000 T=113 SYN (#35)
Aug 12 20:17:35 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 24.43.21.4:1384 Good.Net.139.72:27374 L=48 S=0x00 I=35455 F=0x4000 T=113 SYN (#35)
```

Incident 7- Planet Online - Leeds UK

```
Aug 22 13:20:58 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 62.137.120.243:4244 Good.Net.139.70:27374 L=48 S=0x00 I=45247 F=0x4000 T=107 SYN (#35)
Aug 22 13:20:58 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=6 62.137.120.243:4245 Good.Net.139.71:27374 L=48 S=0x00 I=45248 F=0x4000 T=107 SYN (#35)
```

Incident 8 - BBN Planet - MA USA

```
Jul 12 16:17:27 wren kernel: Packet log: input DENY eth0 PROTO=6 4.34.137.234:4634 Good.Net.139.70:27374 L=48 S=0x00 I=61776 F=0x4000 T=120 SYN (#34)
Jul 12 16:17:27 wren kernel: Packet log: input DENY eth0 PROTO=6 4.34.137.234:4636 Good.Net.139.71:27374 L=48 S=0x00 I=64336 F=0x4000 T=120 SYN (#34)
Jul 12 16:17:30 wren kernel: Packet log: input DENY eth0 PROTO=6 4.34.137.234:4634 Good.Net.139.70:27374 L=48 S=0x00 I=29777 F=0x4000 T=120 SYN (#34)
Jul 12 16:17:30 wren kernel: Packet log: input DENY eth0 PROTO=6 4.34.137.234:4636 Good.Net.139.71:27374 L=48 S=0x00 I=33105 F=0x4000 T=120 SYN (#34)
Jul 12 16:17:36 wren kernel: Packet log: input DENY eth0 PROTO=6 4.34.137.234:4634 Good.Net.139.70:27374 L=48 S=0x00 I=3922 F=0x4000 T=120 SYN (#34)
Jul 12 16:17:36 wren kernel: Packet log: input DENY eth0 PROTO=6 4.34.137.234:4636 Good.Net.139.71:27374 L=48 S=0x00 I=4946 F=0x4000 T=120 SYN (#34)
Jul 12 16:17:48 wren kernel: Packet log: input DENY eth0 PROTO=6 4.34.137.234:4634 Good.Net.139.70:27374 L=48 S=0x00 I=19282 F=0x4000 T=120 SYN (#34)
Jul 12 16:17:48 wren kernel: Packet log: input DENY eth0 PROTO=6 4.34.137.234:4636 Good.Net.139.71:27374 L=48 S=0x00 I=20050 F=0x4000 T=120 SYN (#34)
```

Source of Trace:

Trace came from DND Contractors Network

Detect Generated by:

IPCHAINS v 1.3.9, on Linux Router Project (LRP) V 2.9.4 Materhorn, Linux Kernel Version 2.2.13

Probability of Spoofing:

Very low for all the attempts since the scan requires a 3 way handshake to identify the presence of the Trojan.

Attack Description:

This scan is an attempt to find a known Trojan (Sub-Seven). Incident 1 and 2 are attempts to find the old version of Sub-Seven and the others are attempts on the signature TCP port for Sub-Seven Version 2.1. The scan sends a SYN on the prescribed port(s) as above looking for the completion of a three way hand shake.

A more detailed description of the SubSeven Trojan is included below and is found at the following URL from SANS: http://www.sans.org/newlook/resources/IDFAQ/subseven.htm)

SubSeven is a Trojan for the windows platform. It comes at least in two parts a client and a server. The client is used by the hacker to connect to the victim' s machine. Once the server.exe is installed on the victim's machine the hacker has full access to the victim's machine.

The zip-file I downloaded contained 3 executables:

server.exe The real Trojan, which is installed on the victim's machine sub7.exe The client used by the hacker to connect to his victim's machine EditServer.exe A configuration utility to set several configuration options on server.exe.

The EditServer.exe gives the hacker the opportunity to configure:

- the port used by server.exe
- to set a password for the server
- several other values

and most important to set some notification options, to notify the hacker when his victim(s) is online. This notification can be done using ICQ, IRC, or e mail.

Known Information about SubSeven

Known TCP ports for SubSeven:

- 1243
- 6711
- 6712
- 6713
- 6776

Known TCP ports for SubSeven 2.1

- 27374

Files on an infected machine:

- server.exe
- rundll1.exe
- systray.dl
- Task_bar.exe
- FAVPNMCFEE.dll

- MVOKH_32.dll
- nodll.exe
- watching.dll

Entries in configuration files:

- in system.ini:

an entry on the line containing "shell="

- in win.ini:

an entry on the line containing "load=" or "run= "

- in the registry:

HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\Current\Version\Run
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServices

To be able to connect to the victims machine the hacker needs the ip address of that machine. There are two methods to get this ip address:

1.Using ICQ
If the victim has not enabled IP Hiding in his ICQ User Profile, then the hacker can retrieve this information from the victim's profile.
2.To use the notification option of the trojan. That way the hacker is always notified when his victim(s) connect to the internet. He will even get the IP address and the port number delivered.

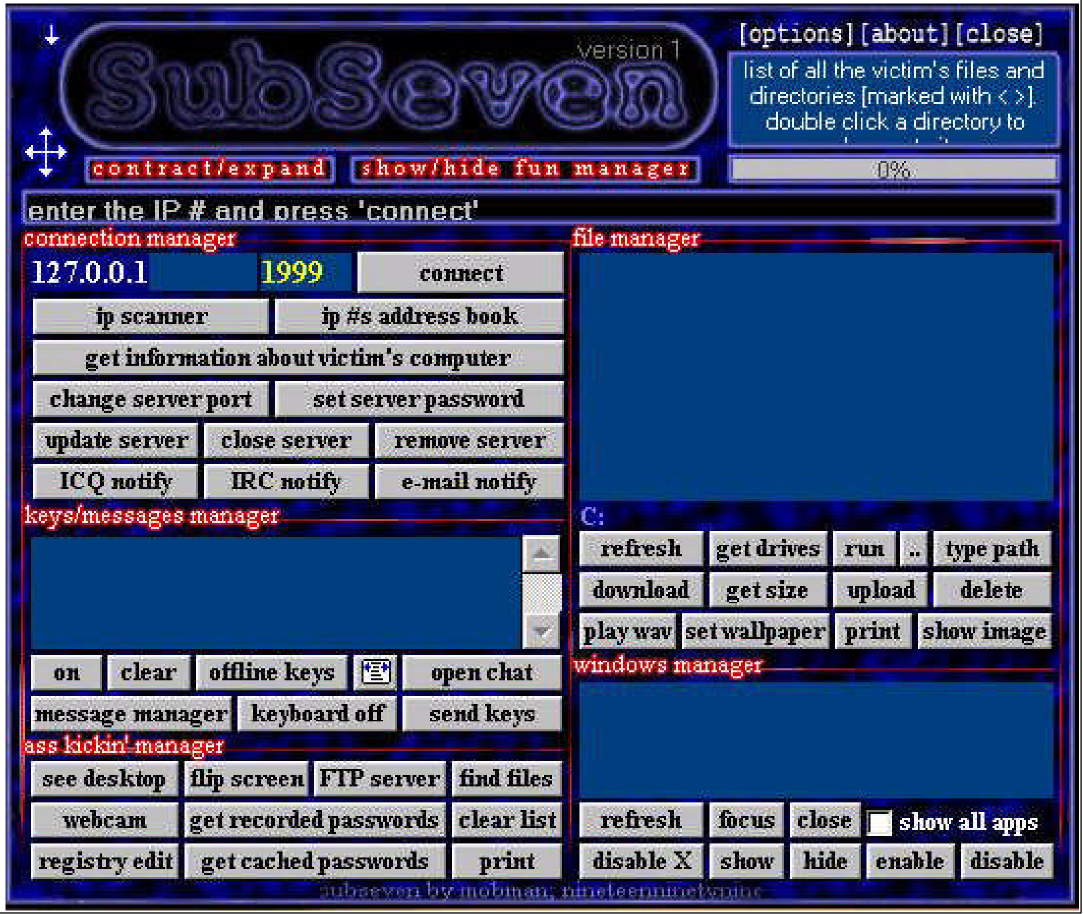
It is claimed in the description of SubSeven that most Antivirus Software won't be able to detect newer versions of it. Have a look in your registry whether the strings SubSeven, "Sub Seven" or "Sub 7" are found. If yes, your machine got infected. If no, well that does not mean that your machine is not infected, since the hacker can set the values used in the registry with the EditServer.exe.

Correlation:

SubSeven is a well documented Trojan attack documented by SANS at the following URL http://www.sans.org/newlook/resources/IDFAQ/subseven.htm)
Also, due to the method used in this practical, Incident 1 and 2 correlate as well as incidents 3-8.

This scan would fall under CVE candidate **CAN-1999-0660**

Screen capture of SubSeven Client Screen



Evidence of Active Targeting:

By taking a general look at the IP ID's in the incidents it looks like most of these were parts of larger scans. Incident 7 shows sequential IP ID's which might indicate a more directed scan.

Severity:

(4 + 3) - (4 +4) = -2

Servers targeted run Web, Mail and DNS (4); SubSeven can be rather revealing (3); Traffic stopped at F.W. (4); Server pretty well maintained (4)

Defensive Recommendations:

No change. Firewall denied access to this scan.

Multiple Choice Test Question:

What other ports has SubSeven been well documented on besides 1243 and 27374?

a) 6711
b) 6713
c) 6776
d) all the above

Answer - d

Detect 2 - Deep Throat Trojan Scan

```
Incident 1 - Austin State University TX USA
Jul  3 16:05:10 wren kernel: Packet log: input DENY eth0 PROTO=17 144.96.153.56:1044 Good.Net.139.70:2140 L=30 S=0x00 I=13111 F=0x0000 T=113 (#35)
Incident 2 - UK-JAK GB
Jul 17  2:58:18 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=17 212.41.33.189:60000 Good.Net.139.70:2140 L=30 S=0x00 I=24462 F=0x0000 T=110 (#34)
Jul 17  4:30:28 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=17 212.41.33.189:60000 Good.Net.139.70:2140 L=30 S=0x00 I=25016 F=0x0000 T=110 (#34)
Jul 17  4:30:28 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=17 212.41.33.189:60000 Good.Net.139.71:2140 L=30 S=0x00 I=25272 F=0x0000 T=110 (#34)
Incident 3 - UK-JAK GB
Jul 19  5:05:43 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=17 212.41.37.86:60000 Good.Net.139.70:2140 L=30 S=0x00 I=25735 F=0x0000 T=110 (#34)
Jul 19  5:05:43 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=17 212.41.37.86:60000 Good.Net.139.71:2140 L=30 S=0x00 I=25991 F=0x0000 T=110 (#34)
Incident 4 - BTNet GB
Jul 30 12:35:17 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=17 213.1.195.133:60000 Good.Net.139.70:2140 L=30 S=0x00 I=35906 F=0x0000 T=111 (#35)
Jul 30 12:35:17 yyy.yyy.4.254 kernel: Packet log: input DENY eth0 PROTO=17 213.1.195.133:60000 Good.Net.139.71:2140 L=30 S=0x00 I=36162 F=0x0000 T=111 (#35)
```

Source of Trace:

Trace came from DND Contractors Network

Detect Generated by:

IPCHAINS v 1.3.9, on Linux Router Project (LRP) V 2.9.4 Materhorn, Linux Kernel Version 2.2.13

Probability of Spoofing:

Very low for all the attempts since the scan requires a 3 way handshake to identify the presence of the Trojan.

Attack Description:

The following description of the Deep Throat Trojan is as found on the SANS web pages at the following URL http://www.sans.org/newlook/resources/IDFAQ/DT.htm

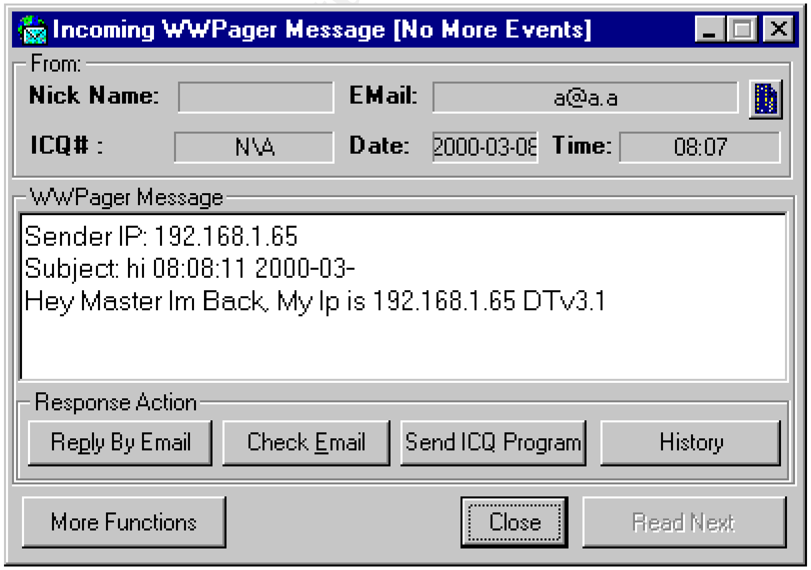
> Using outbound source port 60000, the DT client sends UDP to port 2140. If successful in finding the DT server (compromised box) the DT client initiates a back door, BO-like remote session using ports 2140 and 3150. The log posted is typical of DT's client/server communication when connecting to a compromised box.
>
> We have seen TCP 6670 and TCP 6671 interspersed amongst large-scale multi-port robes. Versions of DT server listen on ports TCP 6670 or TCP 6671 by default, making them detectable by common TCP port scanners.
>
> If a TCP or UDP probe to one of the high listening ports reaches a DT server, DT phones home using ICQ. Either configured at build, or set by the DT handler during a remote session, DT encodes the ICQ User Identification Number in a DAT file in the %WINDIR%\System directory.

Once activated, either by passage of time or inbound port probe, DT negotiates a connection with wwp.mirabilis.com and notifies its maker by HTTP post:

```
Flags: 0x00
  Status: 0x00
  Length: 187
  Time: 08:08:12.336000 03/08/2000
Ethernet Header
  Dest: 00:00:00:00:00:00 [0-5]
  Src: 00:00:00:00:00:00 [6-11]
  Type: 08-00 IP [12-13]
IP Header - Internet Protocol Datagram
  Ver: 4 [14 Mask 0xf0]
  HLng: 5 [14 Mask 0xf]
  Prec: 0 [15 Mask 0xe0]
  TOS: %0000 [15 Mask 0x1e]
  Un: %0 [15 Mask 0x1]
  Lng: 169 [16-17]
  Id: 1024 [18-19]
  FrFg: %010 Do Not Fragment [20 Mask 0xe0]
  FrgO: 0 [20-22 Mask 0x1fffff]
  TTL: 128
  Type: 0x06 TCP [23]
  Sum: 0x2ff2 [24-25]
  Src: 00.00.00.00 [26-29]
  Dest: 205.188.147.55 [30-33]
  No Internet Datagram Options
TCP - Transport Control Protocol
  SPrt: 1027 [34-35]
  DPrt: 80 World Wide Web HTTP [36-37]
  Seq: 478158 [38-41]
  Ack: 1160820900 [42-45]
  Off: 5 [46 Mask 0xf0]
  Rsvd: %000000 [46 Mask 0xfc0]
  Code: %011000 Ack Push [47 Mask 0x3f]
  Win: 8576 [48-49]
  Sum: 0x4a34 [50-51]
  Urg: 0 [52-53]
  No TCP Options
HTTP - HyperText Transfer Protocol
  from=DTv3.1&from 66 72 6f 6d 3d 44 54 76 33 2e 31 26 66 72 6f 6d
```

```
   [54-69]
   email=a@a.a&subj 65 6d 61 69 6c 3d 61 40 61 2e 61 26 73 75 62 6a
   [70-85]
   ect=hi 08:08:11  65 63 74 3d 68 69 20 30 38 3a 30 38 3a 31 31 20
   [86-101]
   2000-03-08&body= 32 30 30 30 2d 30 33 2d 30 38 26 62 6f 64 79 3d
   [102-117]
   Hey Master Im Ba 48 65 79 20 4d 61 73 74 65 72 20 49 6d 20 42 61
   [118-133]
   ck, My Ip is 192 63 6b 2c 20 4d 79 20 49 70 20 69 73 20 31 39 32
   [134-149]
   .168.1.65 DTv3.1 2e 31 36 38 2e 31 2e 36 35 20 44 54 76 33 2e 31
   [150-165]
   &to=66816189&sen 26 74 6f 3d 36 36 38 31 36 31 38 39 26 73 65 6e
   [166-181]
   d  64 [182]
 CkSeq: 0x00000000
```

This results in the BadGuy receiving this ICQ page:



Since source code for DT is available, these ports are changeable. The DT distributor's website includes instructions and wrappers for disguising DT's installation within another Windows executable such as a game.

The significant issue here is that DT can be configured at build to announce its presence instead of waiting for a prober to find it. Alternatively, DT can be configured at build to listen on a port frequently probed by hackers. In either case, DT handlers can passively wait for confirmation that their Trojan is up, running, and waiting for exploitation.

Correlation:

The Deep Throat Trojan has been well documented. Correlations can be found at the following URL's:
http://www.nsclean.com/psc-dt.html
http://www.sans.org/newlook/resources/IDFAQ/DT.htm
http://www.securityfocus.com/archive/75/51754

This scan would fall under CVE candidate **CAN-1999-0660**

Evidence of Active Targeting:

Incident 2 could show evidence of active targeting due to the few hours between scans. The first time only one server is tried. The second time two servers are tested.

Severity:

(4 + 2) - (4 +4) = -3

Servers targeted run Web, Mail and DNS (4); Can cause its share of trouble (2); Traffic stopped at F.W. (4); Server pretty well maintained (4)

Defensive Recommendations:

No change. Firewall denied access to this scan.

Multiple Choice Test Question:

Deep Throat uses what protocol number for it's scan?

a) 6
b) 17
c) 1
d) 21

Deep Throat Client Screen Capture

Answer - b

Detect 3 - RPC Call on port 111

Incident 1 - Korea Network Information Center

Jul	23	20:04:59	yyy.yyy.4.254	kernel:	Packet	log:	input	DENY	eth0	PROTO=6	211.36.42.222:4477	Good.Net.139.70:111	L=60	S=0x00	I=14124	F=0x4000	T=50	SYN (#35)
Jul	23	20:04:59	yyy.yyy.4.254	kernel:	Packet	log:	input	DENY	eth0	PROTO=6	211.36.42.222:4478	Good.Net.139.71:111	L=60	S=0x00	I=14125	F=0x4000	T=50	SYN (#35)
Jul	23	20:05:02	yyy.yyy.4.254	kernel:	Packet	log:	input	DENY	eth0	PROTO=6	211.36.42.222:4477	Good.Net.139.70:111	L=60	S=0x00	I=14602	F=0x4000	T=50	SYN (#35)
Jul	23	20:05:02	yyy.yyy.4.254	kernel:	Packet	log:	input	DENY	eth0	PROTO=6	211.36.42.222:4478	Good.Net.139.71:111	L=60	S=0x00	I=14603	F=0x4000	T=50	SYN (#35)

Source of Trace:

Trace came from DND Contractors Network

Detect Generated by:

IPCHAINS v 1.3.9, on Linux Router Project (LRP) V 2.9.4 Materhorn, Linux Kernel Version 2.2.13

Probability of Spoofing:

Very low since the scan requires a 3 way handshake to identify the presence of the Trojan.

Attack Description:

This attack is another well documented scan for information. The standard rpcbind shipped with Solaris 2.x systems displays this behavior as documented at the following SANS URL http://www.sans.org/newlook/resources/IDFAQ/blocking.htm. an excerpt of which is included here:

> RPC information located at Port 111 is a place to find out where services are running. Numerous vulnerabilities exist, along with exploits ready and waiting for services such as rpcbind and rpcmountd. Network File Service (NFS) has a known rpc-update exploit, the Network Information Service (NIS) update daemon rpc.ypupdated contains vulnerabilities in how it passes commands to certain function calls. This could allow a remote attacker to trick the service into executing arbitrary commands on the system with root privileges. Additionally, client server environments that use remote program calls and port 111 to register and make themselves available, are unfortunately also listing their availability to the less-than nice people who are trying to crack your system. For the unprotected systems that have portmapper running on port 111, a simple "rpcinfo" request is adequate for the potential exploiter to obtain a list of all services running.

This does not appear to be a standard scan in that both UDP and TCP ports 111 are "usually" checked. Here we only see two brief attempts on two external boxes using TCP only.

Correlation:

This vulnerability has been well documented. More information can be found at the following URL's

http://www.cert.org/advisories/CA-94.15.NFS.Vulnerabilities.html
http://www.sans.org/newlook/resources/IDFAQ/blocking.htm

CVE candidate number CAN-1999-0568

Evidence of Active Targeting:

Two external machines received packets from the Korean source. Most likely this would be an attempt to gather information.

Severity:

(4 + 2) - (4 +4) = -3

Servers targeted run Web, Mail and DNS (4); More than likely a reconnaissance attempt (2); Traffic stopped at F.W. (4); Server pretty well maintained (4)

Defensive Recommendations:

Even though the firewall stopped the packets, it is advisable to turn off ANY services that are not specifically in use on exposed systems to reduce the possibility of successful reconnaissance or an intrusion. In this case, we are not running Solaris OS.

Multiple Choice Test Question:

Port mapper scans generally use what protocol(s)?

a) TCP port 111
b) UDP port 111
c) TCP & UDP port 111
d) None of the above

Answer C

Detect 4 - Port 8010 Scan

Incident 1 - OGERTEL - Saudi Arabia

Aug	26	12:59:40	yyy.yyy.4.254	kernel:	Packet	log:	input	DENY	eth0	PROTO=6	212.119.74.92:4395	Good.Net.139.70:8010	L=48	S=0x00	I=10109	F=0x4000	T=114	SYN	(#35)
Aug	26	12:59:40	yyy.yyy.4.254	kernel:	Packet	log:	input	DENY	eth0	PROTO=6	212.119.74.92:4396	Good.Net.139.71:8010	L=48	S=0x00	I=10365	F=0x4000	T=114	SYN	(#35)
Aug	26	12:59:43	yyy.yyy.4.254	kernel:	Packet	log:	input	DENY	eth0	PROTO=6	212.119.74.92:4395	Good.Net.139.70:8010	L=48	S=0x00	I=28285	F=0x4000	T=114	SYN	(#35)
Aug	26	12:59:43	yyy.yyy.4.254	kernel:	Packet	log:	input	DENY	eth0	PROTO=6	212.119.74.92:4396	Good.Net.139.71:8010	L=48	S=0x00	I=27005	F=0x4000	T=114	SYN	(#35)
Aug	26	12:59:43	yyy.yyy.4.254	kernel:	Packet	log:	input	DENY	eth0	PROTO=6	212.119.74.92:4397	Good.Net.139.72:8010	L=48	S=0x00	I=28541	F=0x4000	T=114	SYN	(#35)
Aug	26	12:59:49	yyy.yyy.4.254	kernel:	Packet	log:	input	DENY	eth0	PROTO=6	212.119.74.92:4397	Good.Net.139.72:8010	L=48	S=0x00	I=10621	F=0x4000	T=114	SYN	(#35)

Source of Trace:

Trace came from DND Contractors Network

Detect Generated by:

IPCHAINS v 1.3.9, on Linux Router Project (LRP) V 2.9.4 Materhorn, Linux Kernel Version 2.2.13

Probability of Spoofing:

Very low since the scan requires a 3 way handshake to identify the presence of the Trojan.

Attack Description:

Looking for a vulnerability in WinGate machines.

Correlation:

This vulnerability has been well documented and more information can be found at the following sites:

http://advice.networkice.com/advice/Exploits/Ports/8010/default.htm

BugtraqID: 507 http://www.securityfocus.com/bid/507.html

CVE Candidate: CAN-1999-0508

Evidence of Active Targeting:

All three outside machines are scanned in a very short period of time. This Vulnerability would allow read access to any files located on the directory where WinGate is installed. Interesting that this came out of Saudi Arabia.

Severity:

(4 + 2) - (4 +4) = -3

Servers targeted run Web, Mail and DNS (4); More than likely a reconnaissance attempt (2); Traffic stopped at F.W. and not running WinGate (4); Server pretty well maintained (4)

Defensive Recommendations:

No further action required.

Multiple Choice Test Question:

What port number through WinGate may leave files on the same drive vulnerable to be read?

a) 110
b) 8080
c) 1080
d) 8010

Answer D

Curious Detect 5 - Port 4499 Scan

(NOTE: Only 4 detects are required, but this interested me too much to leave out)

Incident 1 - Rheinisch Bergische Presse-Data - Duesseldorf - Germany

```
Aug  21  9:06:21 yyy.yyy.4.254  kernel: Packet log: input DENY eth0 PROTO=17 149.221.232.18:4156  Good.Net.139.70:4499  L=794 S=0x00 I=26776 F=0x0000 T=108 (#35)
Aug  21  9:06:23 yyy.yyy.4.254  kernel: Packet log: input DENY eth0 PROTO=17 149.221.232.18:4156  Good.Net.139.70:4499  L=794 S=0x00 I=28824 F=0x0000 T=108 (#35)
Aug  21  9:06:25 yyy.yyy.4.254  kernel: Packet log: input DENY eth0 PROTO=17 149.221.232.18:4156  Good.Net.139.70:4499  L=794 S=0x00 I=31128 F=0x0000 T=108 (#35)
Aug  21  9:06:27 yyy.yyy.4.254  kernel: Packet log: input DENY eth0 PROTO=17 149.221.232.18:4156  Good.Net.139.70:4499  L=794 S=0x00 I=33176 F=0x0000 T=108 (#35)
```

Source of Trace:

Trace came from DND Contractors Network

Detect Generated by:

IPCHAINS v 1.3.9, on Linux Router Project (LRP) V 2.9.4 Materhorn, Linux Kernel Version 2.2.13

Probability of Spoofing:

Very low. Since this is UDP it appears as a type of scan that would require a reply of some sort. Spoofing the address would not allow for a reply to the probing machine.

Attack Description:

Unsure at this point. Location of the source led me to look at this a little further. Port 4499 is the last port in a block of unassigned ports (see http://www2.raidway.ne.jp/~sit-k/network/port/port04000-04499.html). All attempts are directed at the fire wall. Another strange item appears to be the size of this packet. Seems a little large to be a scan, unless it carries some sort of payload for a Trojan placed in this location.

Correlation:

Attempts to find correlations for this event have come up empty. I also checked with a Guy at the Canadian CERT (pun intended) and they had not seen this in the current year but recommended that I keep an eye on it and possibly report it to GIAC.

Evidence of Active Targeting:

Looking at IP ID's and the time these were received, it appears as though these packets are directly aimed at the firewall.

Severity:

Not enough information here to decide the severity of this log.

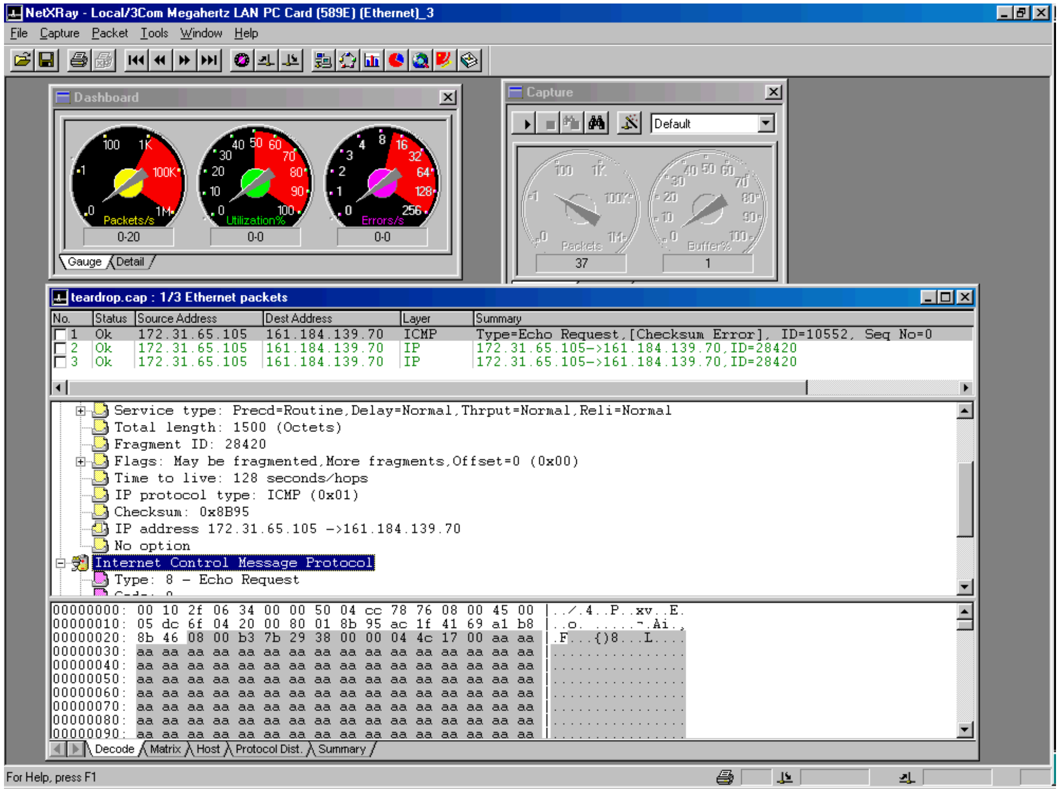
Defensive Recommendations:

Watch for any repeats of this pattern, or from this site. Also, check for any further correlations from other sites. Possibly report to GIAC.

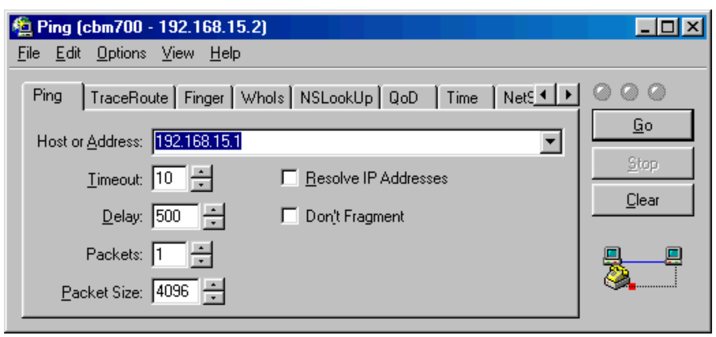
Multiple Choice Test Question:

N/A

Assignment 2 - Attack Evaluation



Screen Shot of NetXray



Screen Shot of Cyberkit

This is actually the last piece of the puzzle for me. I left this section until the end because I needed to gain the experience of looking through the log files and getting a sense for what the attacks were like.

My first thought was to try and re-create a Tear Drop type of attack. What I had done was use an older tool that I still have available to me called Net X-ray. Some of you will remember this product. Network General bought them and incorporated the GUI interface into their Sniffer product, which was then purchased by Network Associates.

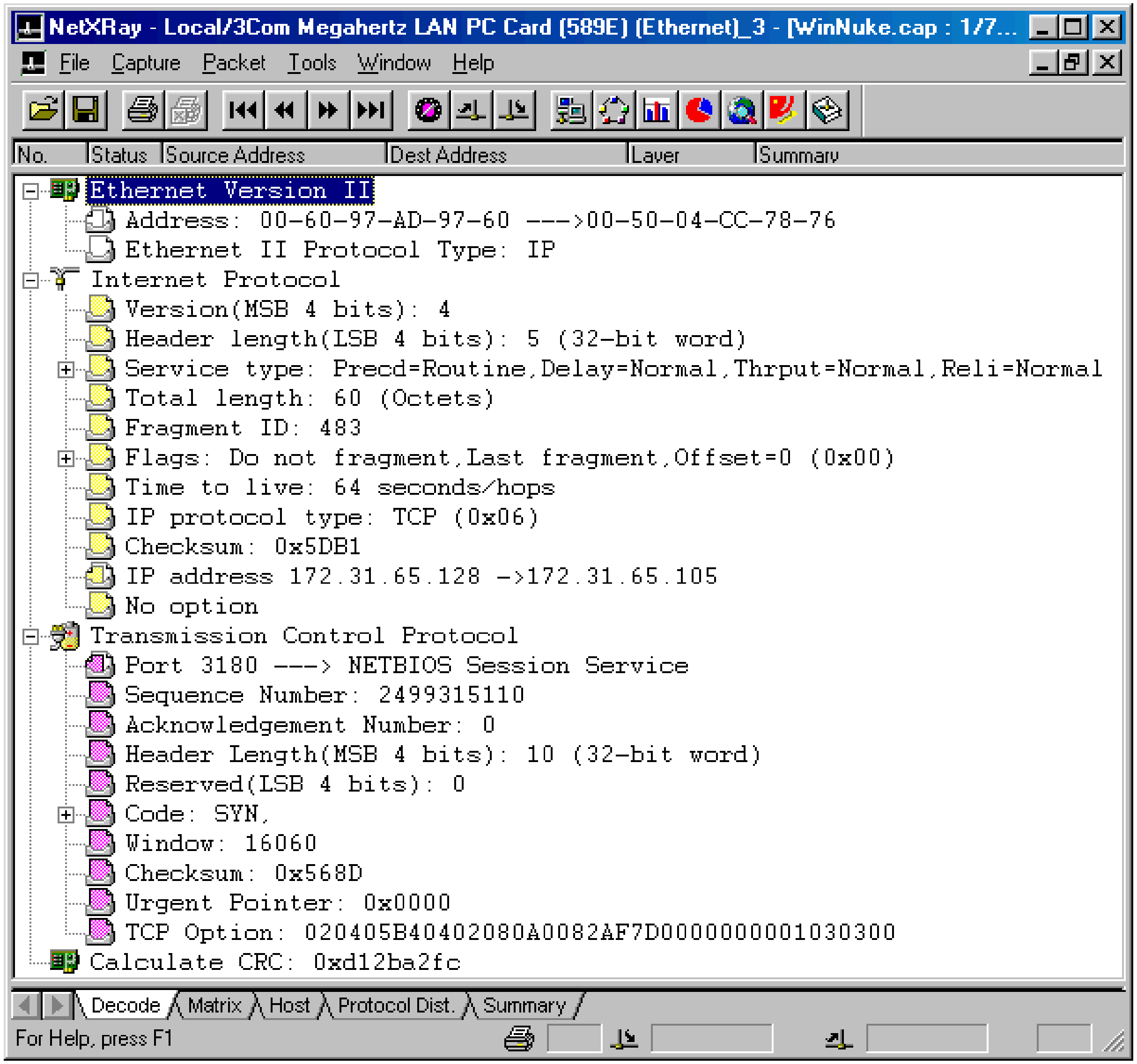
I started NetXray and used a tool called Cyberkit (see http://www.cyberkit.net). With this tool I told it to send a 4096 byte packet to a destination on an Ethernet network. It fragmented the packet up into three packets (two 1514's and one 1158). There is a tool in NetXray that allows you to look at a packet and then modify it for sending. Unfortunately I could not get all three packets modified or even one packet modified and then send all three packets. I would have simply modified the offset and made the first and second packet overlap. If it was the thought that counts I would have been done at this point.

But this is not the case. So I searched around to see if I could come up with another one. I realize that it is not very glamorous but I

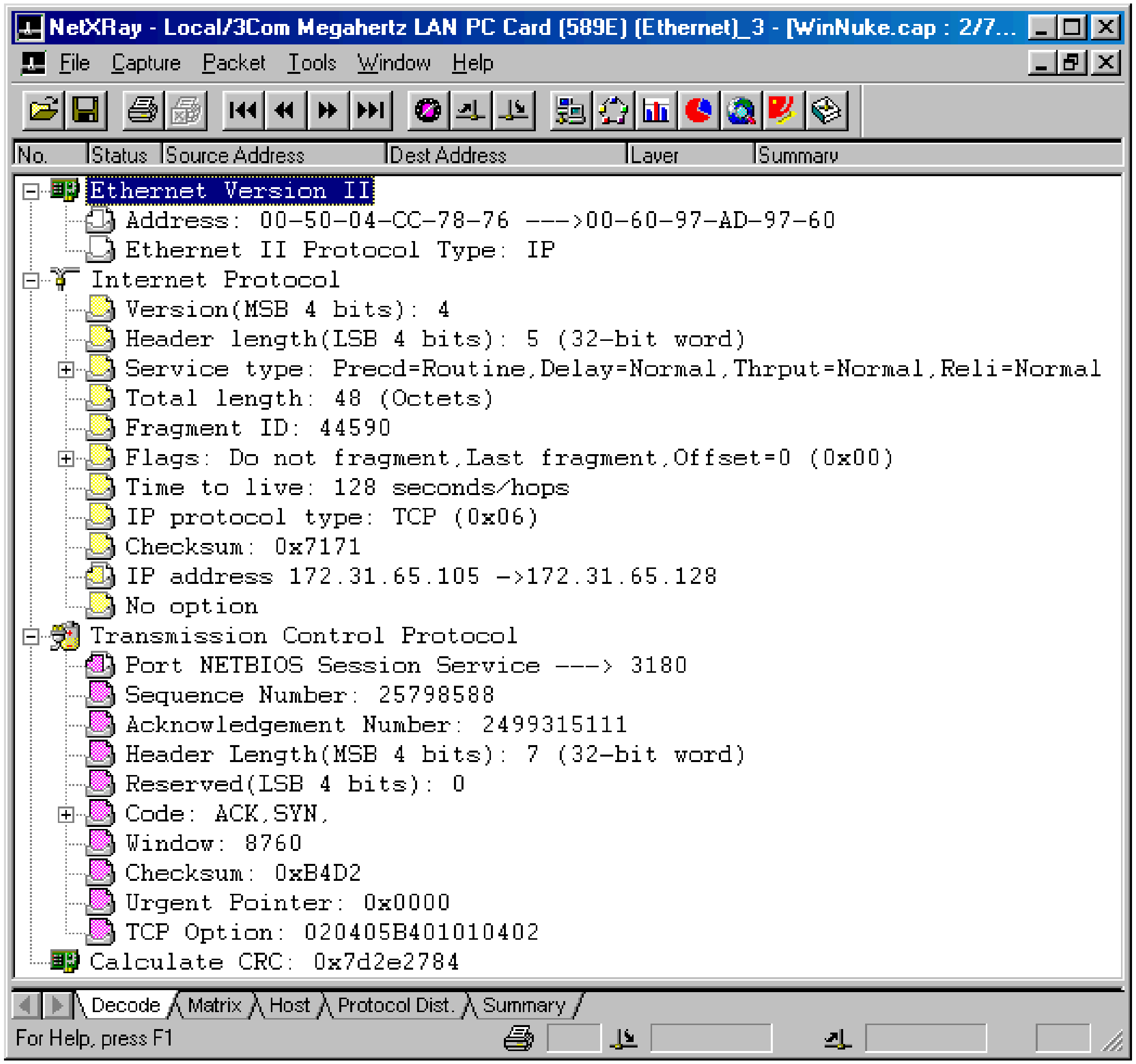
managed to find a copy of the C source code for WinNuke http://www.users.fast.net/~lnsmall/winnuke.htm .
Two days before my partner had set up a Linux machine for NMAP (to test his firewall configuration for the SANS Firewalls and Perimeter Protection Track). Since I still have some learning to do as far as Linux goes, I asked for his help and he successfully compiled a copy of the program. I then turned on my NetXray and fired a WinNuke at my Windows 98 laptop. I was pretty sure it would not crash my machine since this has been around for a while and my O/S is at the latest levels. My machine stayed up and I captured the whole thing with NetXray.

What follows is a listing and explanation of what I saw on the trace. I am having a strange error when I try and export the information so I'll show it here with screen captures.

Laptop = 172.31.65.105
Linux station = 172.31.65.128

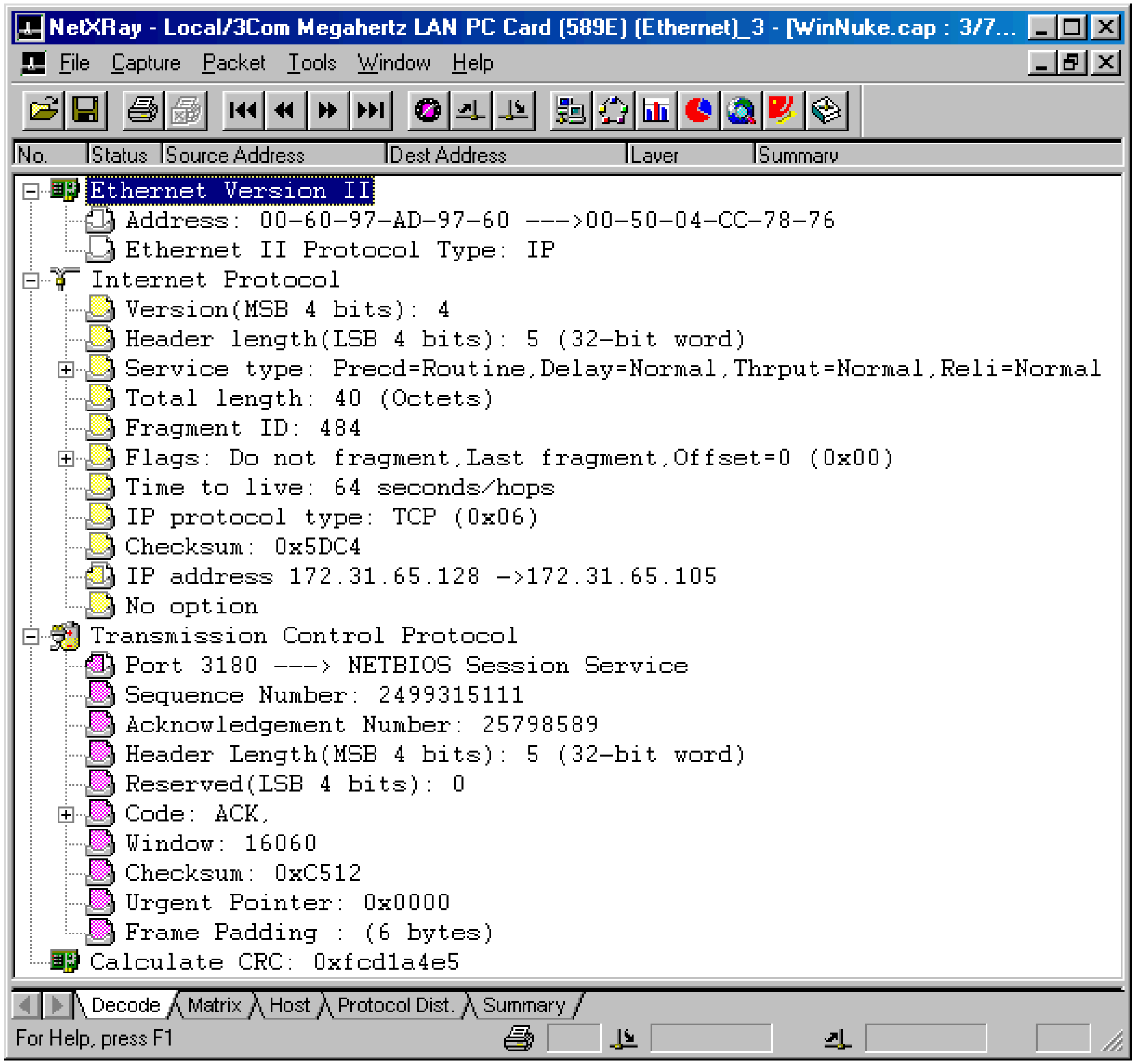


Packet number 1 that is sent out from the Linux station to the laptop. In the TCP portion of the frame you can see that the source port is 3180 and the destination port is 139 (Netbios Session Service). A little further down you can see the SYN, part 1 of the three way hand shake.

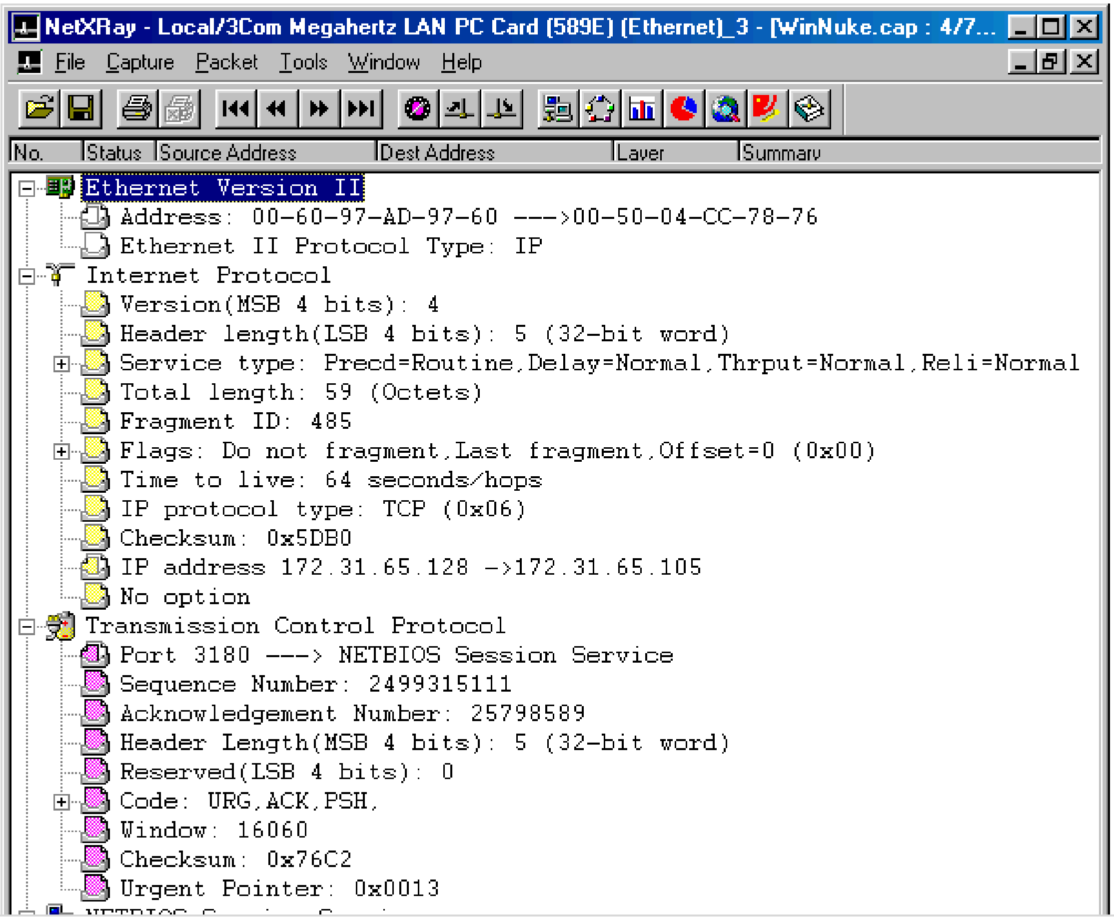


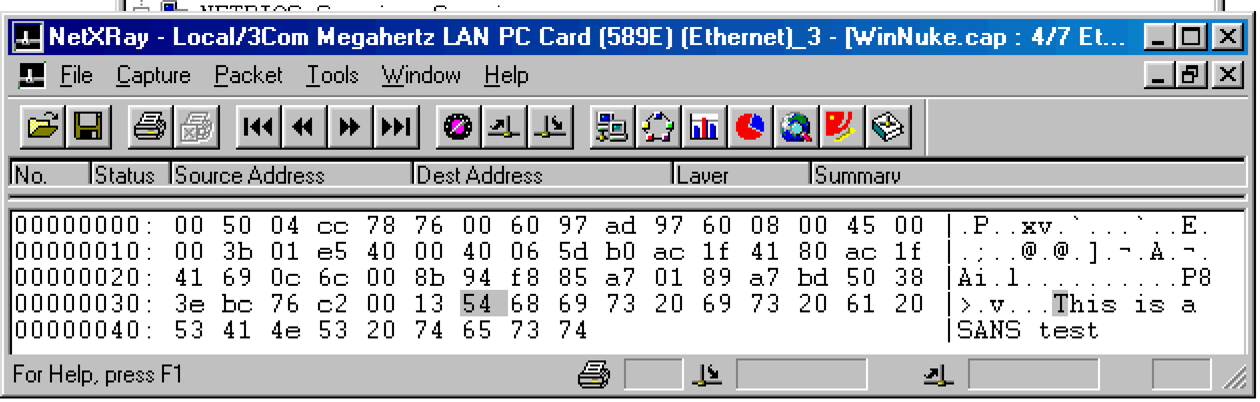
Packet 2 is the reply from my Windows 98 laptop as can be seen from the IP addresses and from the ports (139 -> 3180).

Also note the SYN, ACK combination which means that the connection is now half open.



Packet 3 shows the ACK coming back from the Linux station to complete the 3 way handshake and the session opening with port 139 on the Windows 98 station.

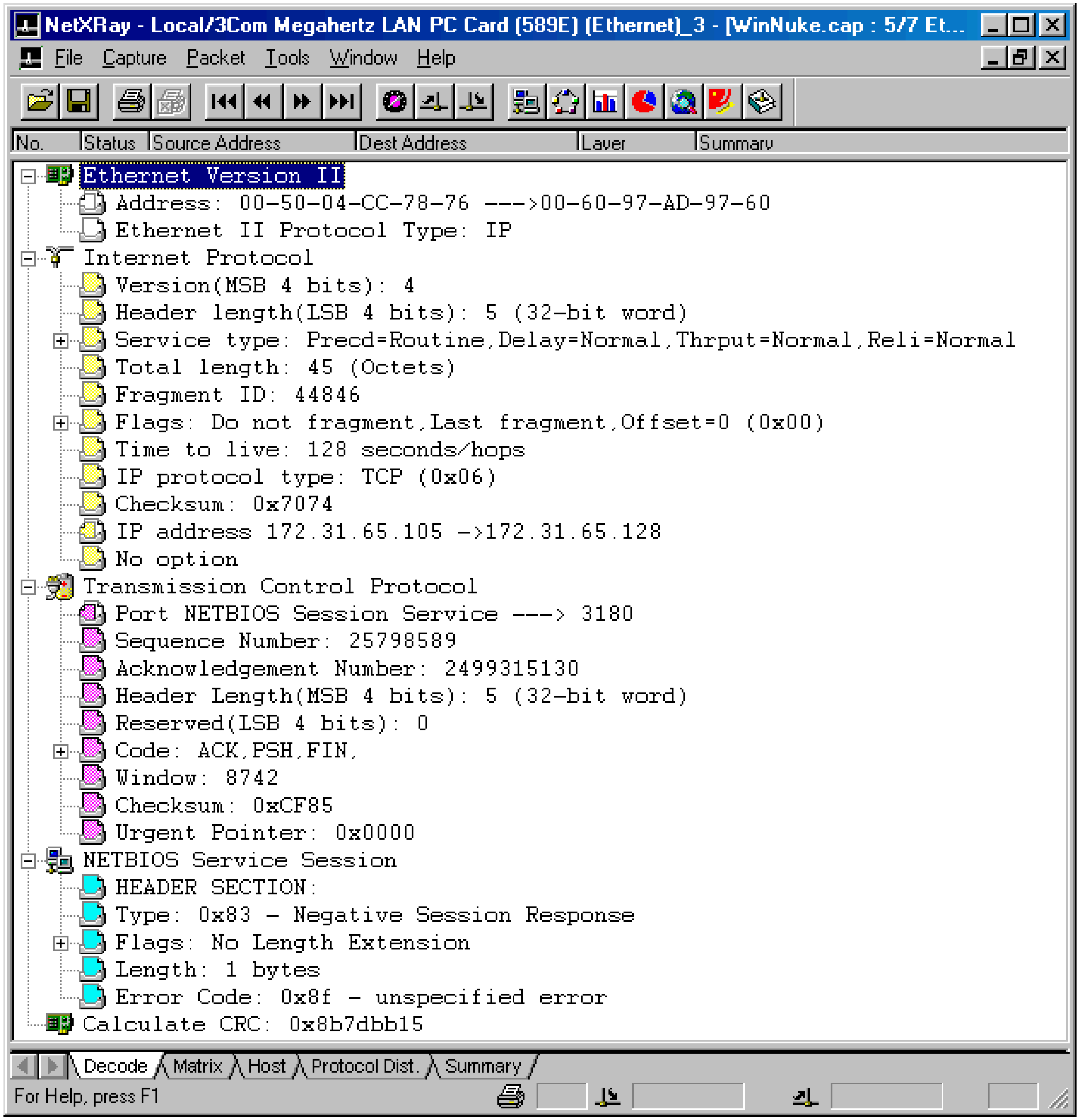




LEFT: Packet 4 is where the knockout punch is delivered. As you can see there is a new portion of the frame present dedicated to Netbios.

Also notice that the TCP flags have three values set: URG, ACK and PSH.

BELOW: is a hex dump of the same packet to the left. Notice the phrase put in after the Netbios Header in the "Reserved" section of the Netbios frame. It is this inconsistency that the unsuspecting versions of windows could not handle and froze. Notice the words chosen "This is a SANS test".



On an un-patched machine, this frame would not have come back because the code that handles the Netbios negotiation would be in a precarious state. Because this version has been fixed, it responds back to the WinNuke Linux machine with a HEX error code of - 8f and continues on it's merry way.

Basically this attack had taken advantage of some code that did not have a path to traverse if data was sent in the header of the Netbios negotiations.

--0--

Side note: Another example of this comes to mind that is relevant here. About 4 years ago I came to work for a company that had about a 2500 node network, in a flat configuration with both Ethernet and Token Ring. To get between the two environments we bridged. Yes, we bridged the traffic. You can imagine the complexity of the code required to bridge between these two environments.

Anyway, as the situation goes we began having problems with the 8229 bridge (an IBM device) where for no apparent reason it would stop forwarding frames on it's Ethernet interface. Needless to say this caused quite a bit of commotion. So much so that VP's began calling to see what the problem was.

The straw that broke the Network Guy's back, so to speak was the night he (me) was woken up 4 times during the night to reset the 8229. The next day a gentleman from IBM got on a plane and came out to lend us hand. It was almost impossible for us to figure out what was going on. A sniffer couldn't help because we had no trigger for the event. When Brian arrived he set up a special dump on the box so that when it hung he could see what was going on.

It took two days before it happened again. The problem was being caused by a specific device that was gerating an IPX packet with a correct header but an empty data portion of the frame. As the logic was traversed to convert the frame from Ethernet to Token Ring it hit a dead end path because the micro-code was not prepared to handle an IPX frame with no data. At that point it stopped forwarding frames.

The ending of the story is that Brian wrote us a micro-code patch and we stopped having that problem.

The moral of the story is: there are holes in most code - sometimes the bad guys find them and sometimes regular situations find them.

Assignment 3 - Scenario Analysis

Two distinct sets of log files were provided for this part of the practical. The first set contained a date and time, a short description of what the alarm was, Source IP and port, and Destination IP and port.

Type 1
07/14-00:03:20.138859 [**] WinGate 1080 Attempt [**] 168.120.16.250:55067 -> MY.NET.97.135:1080
07/14-00:04:04.529242 [**] WinGate 1080 Attempt [**] 203.155.129.248:4387 -> MY.NET.97.135:1080
07/14-00:16:55.256883 [**] WinGate 1080 Attempt [**] 203.155.129.248:4524 -> MY.NET.97.135:1080
07/14-00:25:31.576247 [**] WinGate 1080 Attempt [**] 168.120.16.250:55837 -> MY.NET.97.135:1080

The second log format contained a date and time (in a slightly different layout than the type 1 logs, a source IP and port, a destination IP and port, the protocol or flag set, the flags, and sometimes a description of what was strange in the log file.

Type 1	Type 2	Type 1	Type 2	Type 1
06/27	06/27	07/14	07/14	07/27
06/28		07/17	07/17	07/28
06/29				07/29
06/30	06/30	07/19		07/30
07/08	07/8		07/24	08/01
	07/9			
07/10				08/03
07/11	07/11			08/04
07/12				08/05
07/13				
		07/26	07/26	

Log file dates

Type of Snort Alarm	Count
Attempted Sun-RPC-high-port access	2318
External RPC call	8
FTP-bad-login	1
GIAC-000218-VA-CIRT port 34555	206
GIAC-000218-VA-CIRT port 35555	186
Happy 99 Virus	4
IDS08 TELNET daemon-active	1
IDS127-Telnet Login Incorrect	7
IDS246 MISC Large ICMP Packet	5
IDS247-MISC-Large UDP Packet	1170
Napster 7777 Data	170
Napster 8888 Data	323
Napster Client Data	12
NMAP TCP ping!	46
Null scan!	98
PING-ICMP Destination Unreachable	12313
PING-ICMP Source Quench	1
PING-ICMP Time Exceeded	6690
Possible wu-ftpd GIAC000623	7
Probable NMAP Fingerprint attempt	1
Queso Fingerprint	11
SMB Name Wildcard	240
SNMP public access	1188
SUNRPC highport access!	20
SYN-FIN scan!	20067
Watchlist 220 IL-ISDNNET-990517	13972
Watchlist 222 NET-NCFC	4776
WinGate 1080 Attempt	2240
WinGate 8080 Attempt	3291

Type 1 Log File Summary

Type 2
Jul 27 01:38:40 205.188.247.194:21 -> MY.NET.97.211:3991 UNKNOWN *1**R*** RESERVEDBITS
Jul 27 02:03:07 211.60.222.33:1323 -> MY.NET.1.0:53 SYN **S*****

Jul 27 02:03:07 211.60.222.33:1324 -> MY.NET.1.1:53 SYN **S*****
Jul 27 02:03:07 211.60.222.33:1327 -> MY.NET.1.4:53 SYN **S*****

As is noticeable from the figure on the page, there are large gaps in the log files. These gaps, at times, can be like trying to do a puzzle without all the pieces. And without all the pieces you can be left wondering what the picture is. It is important to attempt to correct this for the future where possible. Some suggestions might be to add portable UPS to the sensor systems and the network devices the connect to. Make sure you have an appropriate amount of memory and disk space in the machine. These things will help to collect more a more complete logging picture for the future.

Let's begin by looking at what kind of traffic is present in the Type 1 log files and where possible incorporate the Type 2 logs. A summary of the results from the type 1 logs is listed to the right. Each type of Snort alarm is described in detail below.

Type of Snort Alarm	Count
Attempted Sun-RPC-high-port access	2318
External RPC call	8
FTP-bad-login	1
GIAC-000218-VA-CIRT port 34555	206
GIAC-000218-VA-CIRT port 35555	186
Happy 99 Virus	4
IDS08 TELNET daemon-active	1
IDS127-Telnet Login Incorrect	7
IDS246 MISC Large ICMP Packet	5
IDS247-MISC-Large UDP Packet	1170
Napster 7777 Data	170
Napster 8888 Data	323
Napster Client Data	12
NMAP TCP ping!	46
Null scan!	98
PING-ICMP Destination Unreachable	12313
PING-ICMP Source Quench	1
PING-ICMP Time Exceeded	6690
Possible wu-ftpd GIAC000623	7
Probable NMAP Fingerprint attempt	1
Queso Fingerprint	11
SMB Name Wildcard	240
SNMP public access	1188
SUNRPC highport access!	20
SYN-FIN scan!	20067
Watchlist 220 IL-ISDNNET-990517	13972
Watchlist 222 NET-NCFC	4776
WinGate 1080 Attempt	2240
WinGate 8080 Attempt	3291

Type 1 Log File Summary

Attempted Sun-RPC-high-port access - This large amount of traffic is destined for Sun Solaris machines. This could be an attempt at mapping what service are avaialble on the affected Solaris O/S's using a high UDP port. Since little is know about the actual site it is advisable to make sure any Sun Solaris boxes are at the latest patch levels and this kind of traffic should be blocked at the perimeter.

Type 1	Type 2	Type 1	Type 2	Type 1
06/27	06/27	07/14	07/14	07/27
06/28		07/17	07/17	07/28
06/29				07/29
06/30	06/30	07/19		07/30
07/08	07/8		07/24	08/01
	07/9			
07/10				08/03
07/11	07/11			08/04
07/12				08/05
07/13				
		07/26	07/26	

Log file dates

External RPC call - This is an attempt to use the SUN RPC function to gather information about the system. This is only an issue if the destination is a Solaris machine that is not appropriately patched to protect against these attempts. All 8 attempts appear to come out of the US (Only 2 source IP's) and all were directed at the local machine MY.NET.6.15. This traffic should not come through the perimeter and should be blocked at the firewall. Although this traffic may not have been hostile, the information gathered may allow for a serious attack.

FTP-bad-login - This appears to be a failed login attempt to a Bell Atlantic device from an inside machine. After checking the Type 2 logs to find no activity in the same time frame, we can assume that this is innocuous traffic.

GIAC-000218-VA-CIRT port 34555 - Information about this item can be found at the following SANS link http://www.sans.org/y2k/021800.htm . The information suggests that the Back Orifice Trojan was the means by which another Trojan was placed on the system to perform UDP floods. The key ports seem to be 34555 and 35555. Looking through the attempts we do not see any high port access to the machines in question. This may be leading to a false positive. Upon further investigation of the available logs, only one B.O. scan was launched and appeared to be unsuccessful (also aimed at a different machine than these).

Destination
MY.NET.10.89
MY.NET.130.65
MY.NET.162.200
MY.NET.201.2
MY.NET.217.70
MY.NET.97.204
MY.NET.97.229
MY.NET.97.230
MY.NET.98.136

Napster

GIAC-000218-VA-CIRT port 35555 - Please see above for explanation.

Happy 99 Virus - These 4 alarms all happen at different times and all to the well known port 25 (SMTP). It appears as though these are legitimate signatures and the following list of machines should be examined for the virus: MY.NET.6.34, MY.NET.253.42, MY.NET.6.47, and MY.NET.110.15. Since little is known about this specific site it may be advisable to install a virus scanner on the mail server for the corporation and individual virus scanners on the workstations to help minimize the effect of mail viruses.

IDS08 TELNET daemon-active - 1 incident where an internal station responded to an external telnet connection from 24.25.111.117 (An ISP from VA). MY.NET.99.51 needs to be examined very carefully if it is an internal machine. Telnet should not be allowed passed the perimeter of a network. Anything to be telneted to should be placed in a DMZ that is appropriately secured.

Source_Port
195.25.86.2:80
195.54.105.6:53
195.54.105.6:80
205.128.11.157:53
205.128.11.157:80
209.218.228.201:53
209.218.228.201:80
209.218.228.46:53
209.218.228.46:80
216.127.150.136:57882

NMAP

IDS127-Telnet Login Incorrect - 7 attempted logins on two internal machines that appeared to have failed. Telnet should not be permitted from an insecure site. Maybe an FTP server might be in order here?

IDS246 MISC Large ICMP Packet - 5 attempts to MY.NET.70.121 may show a Ping'O-Death signature. MY.NET.70.121 should be checked for updated O/S to make sure it is not susceptible to such attacks and Only ICMP echo replies should be statefully allowed in to the network to aid in prevention of this type of event.

IDS247-MISC-Large UDP Packet - After looking at the UDP port one might conclude that this is Real Audio which can be found here or on port 6971 as well. Looking up the source shows that the traffic is originating from the Korea Network Information Center. A quick search through SANS will show you that many scans and attacks have come from this area. I would say they are looking for the Gatecrasher Trojan. Once again, secure the internal machines with virus scanners to assist in preventing infections. Also, you may want to simply block the range of IP's that this is coming from. If you don't do business with them the traffic need not even come your way especially if it's hostile in nature when it does.

Napster 7777 Data / **Napster 8888 Data / Napster Client Data** - These three groups all are related to the Napster MP3 sharing programs. These offer IRC chat, a form of FTP and MP3 sharing. A list of devices that received Napster data are listed to the left. Many places have banned the use of Napster clients because of the rather large amount of bandwidth that they consume, not to mention the IRC security risk. It is recommended that the company develop a policy on Napster and take steps to block this kind of traffic (see http://www.und.edu/dept/CC/announce/napster.html for an example of a site that has banned Napster).

NMAP TCP ping! - NMAP is a powerful mapping tool. It can most certainly be used for reconnaissance by those with less that good intentions towards a corporate network. As can be seen by the IP's listed to the right there were 6 separate IP addresses attempting NMAP scans. NMAP also provides a facility to mask its true identity. Both the 195.25 and the 195.54 addresses come out of Sweden. The 205.128 address is registered to HeadHunter.net out of Atlanta Georgia. 209.118.228 is registered to RND Networks out of NJ. But the most interesting of the bunch is the last one:

07/28-23:32:23.408944 NMAP TCP ping! 216.127.150.136:57882 -> MY.NET.253.114:1

This scan is coming from Xecunet, LLC. 5744-R Industry Ln., Frederick, MD 21704, US. And appears to be looking for IRIX machines running tcpmux (IRIX is Silicon Graphics flavor of Unix). A vulnerability here could allow the source to log on as guest. IRIX machines have been know to ship with accounts without passwords (see http://www.cert.org/incident_notes/IN-98.01.irix.html). If this is an IRIX machine it should be looked at closely to determine if it has been compromised. Also, this traffic should be blocked at the firewall.

Null scan! - There are a number of apparent explanations for this traffic. First, a bunch of what is seen here appears to be Napster traffic. It is possible that the client that is being used was poorly written so that it does not include any flags.

```
07/12-13:16:01.954025 Null scan! * [**] 141.44.164.142:1644 -> MY.NET.70.227:6699
07/12-13:36:19.825612 Null scan! * [**] 141.44.164.142:1870 -> MY.NET.70.227:6699
07/11-15:31:18.493853 Null scan! * [**] 141.44.164.142:3375 -> MY.NET.70.227:6699
07/28-12:39:47.189229 Null scan! * [**] 144.41.242.202:1102 -> MY.NET.97.210:6699
```

The other possibility is that there is a piece of network gear that is clearing the flags as they pass through (maybe overloaded). This would explain the few number of packets per session/time that we see displaying this scenario.

Conversely, buried in the NMAP scans the following can be found:

```
08/03-19:59:23.823304 Null scan! * [**] 149.225.111.69:7904 -> MY.NET.60.14:37
08/03-19:59:23.729894 Null scan! * [**] 149.225.111.69:7904 -> MY.NET.60.14:7
08/03-19:59:23.877719 Null scan! * [**] 149.225.111.69:7904 -> MY.NET.60.14:137
08/03-19:59:23.971660 Null scan! * [**] 149.225.111.69:7904 -> MY.NET.60.14:513
08/03-19:59:23.774250 Null scan! * [**] 149.225.111.69:7904 -> MY.NET.60.14:22
```

Although not in time order you can see by the destination ports that this is a very specific scan on a selected number of well known ports. The location from which the scan takes place might also help to identify that someone may be looking:

EUnet Deutschland GmbH (NET-CUMULUS-)
Emil-Figge-Str. 80

D-44227 Dortmund
DE

Also appears to be a null scan for a WinProxy device embedded in here:

07/12-05:16:01.431555 Null scan! [**] 62.161.99.120:1721 -> MY.NET.188.32:**8080**

This one originating out of France:

```
inetnum:     62.161.96.0 - 62.161.120.255
netname:     FR-FTCI-3
descr:       FTCI
descr:       40, rue Gabriel Crie
descr:       92240 Malakoff
country:     FR
admin-c:     CL1478-RIPE
tech-c:      LT723-RIPE
status:      ASSIGNED PA
mnt-by:      OLEANE-NOC
changed:     hostmaster@oleane.net 19991126
source:      RIPE
```

Anything legitimate here is reconnaissance. The above mentioned addresses should be monitored for any further activity.

PING-ICMP Destination Unreachable - It took a bit of looking through these to find any kind of recognizable pattern. It appears as though we are seeing the attempt at a Denial of Service directed towards MY.NET.70.121. In 1 min we see nearly 400 ICMP attempts from multiple machines. Two possible explanations are it is one machine that is spoofing addresses or a DDoS. I prefer the former, in that if it was DDoS we would see an even greater volume of traffic and machines involved. 91.8 % of these ICMP's are destined for MY.NET.70.121. May want to deny uninitiated ICMP reply messages at the firewall to prevent this kind of thing in the future.

PING-ICMP Source Quench - There is one of these. A device from Level 3 Communications is asking MY.NET.70.121 to slow down the traffic it is sending. This may be as a result of the DoS in the previous description. MY.NET.70.121 might be responding at such a high rate (or should I say the kind router in front of it which may very well be capable of sending back a high volume of ICMP messages) that a device up stream is asking it to slow down.

PING-ICMP Time Exceeded - MY.NET.140.9 appears to be suffering the result of someone using it's IP address for spoofing purposes. 5830 of the total 6690 are aimed at MY.NET140.9. Once again, with statefull inspection at the perimeter, these packets would not have made it past the firewall since MY.NET.140.9 did does not appear to have originated any of the traffic.

Possible wu-ftpd GIAC000623 - There are three major vulnerabilities associated with WU-FTPD, a fairly common FTP program (see http://www.cert.org/advisories/CA-99-13-wuftpd.html). The following logs warrant checking the two source machines for the wu-ftpd program. If present they should be patch to the most current levels.

```
06/30-16:33:57.773279 Possible   wu-ftpd GIAC000623   [**]   151.164.223.206:4499 ->   MY.NET.99.16:21
06/30-16:34:00.037398 Possible   wu-ftpd GIAC000623   [**]   151.164.223.206:4499 ->   MY.NET.99.16:21
```

06/30-16:35:11.406398	Possible	wu-ftpd GIAC000623	[**]	151.164.223.206:4500	->	MY.NET.144.59:21
06/30-16:35:13.560305	Possible	wu-ftpd GIAC000623	[**]	151.164.223.206:4500	->	MY.NET.144.59:21
06/30-16:35:13.626498	Possible	wu-ftpd GIAC000623	[**]	151.164.223.206:4500	->	MY.NET.144.59:21

Probable NMAP Fingerprint attempt - Not sure what actually set off this log entry. There is only one item in this category, so I'm not sure how the snort scanner decided that this could be a NMAP fingerprinting. I also checked the Type 2 log files for any correlation during the same time frame but there were no logs captured.

07/12-12:46:34.921774 Probable NMAP Fingerprint attempt [**] 24.200.160.45:1548 -> MY.NET.70.241:8899

Queso Fingerprint - Queso is a utility like NMAP that aids in determining what O/S is running on a machine by sending it strange packets and seeing how they respond. As stated in a detect by Mr. William Miller (see http://www.sans.org/y2k/072500.htm) Queso tends to use the reserved bits as can be correlated by some of the log entries from the Type 2 logs.

TYPE 1 Logs
07/17-15:20:37.812781 Queso Fingerprint [**] 193.233.7.254:3121 -> MY.NET.99.20:113
07/17-15:37:53.409978 Queso Fingerprint [**] 193.233.7.65:3138 -> MY.NET.99.23:113
07/17-15:41:44.730499 Queso Fingerprint [**] 193.233.7.65:3139 -> MY.NET.99.23:113

TYPE 2 Logs
07/17 15 20 37 193.233.7.254 3121 -> MY.NET.99.20 113 SYN **21S*****
07/17 15 37 53 193.233.7.65 3138 -> MY.NET.99.23 113 SYN **21S*****
07/17 15 41 44 193.233.7.65 3139 -> MY.NET.99.23 113 SYN **21S*****

Looking at these two you can see that the program who created the packets filled the two reserved bits with a 2 and a 1. This appears to be reconnaissance again. Out of spec packets should be dropped at the firewall (or at least set up the firewall to respond with a MAC O/S pattern no matter what Queso throws at it - ah come on it's the only time us security guys can have this kind of fun!)

SMB Name Wildcard - There are 240 of these logged in the Type 1 log files and most of them are between internal devices. This traffic is common between internal Windows NT machines. There are a few external devices which attempt this at internal machines. This should simply be blocked at the firewall.

SNMP public access - All of these alarms are generated by internal source machines. All 1188 destinations belong to MY.NET.101.192. Since it was public access probably no changes were made to the machines (would use private if they wanted to write using SNMP). The community string should be changed on MY.NET.101.192 to something other than public.

SUNRPC highport access! - There are 20 of these and they are similar to the **Attempted Sun-RPC-high-port access** description earlier in this assignment. This is active information gathering. Once again - all Sun stations should be kept at current patch levels.

SYN-FIN scan! - At 20067 log entries for this subject it is by far the largest item identified in this log. 20020 of these came from one address (202.0.178.98) who was attempting to map the entire company. This address is registered as follows:

inetnum 202.0.160.0 - 202.0.179.255

netname	CMNET-HK
descr	China Motion Telcom Holdings Ltd.
descr	Roaming Paging Services Provider
descr	Roaming Trunking Services Provider
descr	Hong Kong
country	HK

This kind of activity should be reported to the security people in Honk Kong. A few other addresses were active but nothing compared to the above. Or if it's a competitor, set up a couple of dummy stations that have misleading information on them.

Destination	Count
MY.NET.100.165	75
MY.NET.100.230	838
MY.NET.110.150	3
MY.NET.145.18	20
MY.NET.145.9	7
MY.NET.253.41	2133
MY.NET.253.42	703
MY.NET.253.43	811
MY.NET.253.52	3
MY.NET.253.53	4
MY.NET.6.35	46
MY.NET.6.7	133

Watchlist 222 Destinations

Watchlist 220 IL-ISDNNET-990517 - Well, this is a bit of a mystery to me. From what I can gather from reading other assignments, these log entries are present due to an entry placed in the Snort rules. To try and discover what this was all about I tried a number of searches on "Watchlist" and came up empty (other than the Watchlist log files in GIAC reports). One of the other assignments hinted that this entry was user initiated and was meant to specifically watch a certain type of traffic. So I next went to the Internet and from the Watchlist title I typed in the following: www.isdnnet.com and came up with a site that delivers non real-time audio. With a quick look at the traffic from the Snort log we can see that this looks just like the Napster traffic that was described earlier.

```
07/08-13:28:45.239555 Watchlist 220 IL-ISDNNET-990517 [**] 212.179.41.218:1032 ->
MY.NET.217.114:6688
07/08-13:28:45.548113 Watchlist 220 IL-ISDNNET-990517 [**] 212.179.41.218:1032 ->
MY.NET.217.114:6688
07/08-13:28:45.724503 Watchlist 220 IL-ISDNNET-990517 [**] 212.179.41.218:1032 ->
MY.NET.217.114:6688
07/08-13:28:45.885166 Watchlist 220 IL-ISDNNET-990517 [**] 212.179.41.218:1032 ->
MY.NET.217.114:6688
07/08-13:28:45.972355 Watchlist 220 IL-ISDNNET-990517 [**] 212.179.41.218:1032 ->
MY.NET.217.114:6688
```

This must be present in the logs because the company wants to know who is going to this site. A list including the IP address of internal machines and the log counts is included here.

Destination	Count
MY.NET.110.245	5
MY.NET.151.33	64
MY.NET.152.10	4
MY.NET.179.51	1702
MY.NET.179.77	5
MY.NET.181.242	2
MY.NET.181.88	85
MY.NET.182.94	1808
MY.NET.203.190	4
MY.NET.217.114	5202
MY.NET.217.38	4323
MY.NET.253.24	4
MY.NET.253.42	1
MY.NET.253.52	1
MY.NET.53.28	745
MY.NET.6.35	7
MY.NET.97.225	10

Watchlist 220 Destinations

Watchlist 222 NET-NCFC - Another class B that appears to be under scrutiny. Output from www.arin.net WHOIS is as follows:

The Computer Network Center Chinese Academy of Sciences (NET-NCFC)
P.O. Box 2704-10,
Institute of Computing Technology Chinese Academy of Sciences
Beijing 100080, China

Netname: NCFC
Netnumber: 159.226.0.0

Flags	Count
SF**	839
********	83
*1**R***	37
***FR*A*	27
*1*F***U	16
21*FRP**	12
***FRPA*	12
S*R*	11
21*FR**U	10
2*SF*P*U	8

Top 10 OOS flags

Coordinator:
Qian, Haulin (QH3-ARIN) hlqian@NS.CNC.AC.CN
+86 1 2569960
Again, for reference there is a list of destination addresses and the counts for this specific Watchlist.

WinGate 1080 Attempt / WinGate 8080 Attempt - There are 5531 of these in the Type 1 logs. I've examined these from a number of different angles. Over 2900 of the attempts are aimed MY.NET.253.105. This machine is highly suspect as over 52 unique addresses have accessed it on port 8080. Checking both type 1 and type 2 logs for any information about MY.NET.253.100's outbound activities has shown nothing (and may not because the traffic being proxied may be legitimate and not picked up by the sensors). This still warrants examining MY.NET.253.105 for some sort of compromise and monitoring the activity to/from it more closely.

Source_IP	Count
193.173.174.119	713
211.7.235.4	99
192.193.195.132	15
205.188.237.89	12
132.205.201.12	10
210.121.242.164	9
208.18.8.16	8
205.188.247.194	8
192.193.195.132	7
132.205.201.12	6

Top 10 IP's producing OOS Packets

Also, more than 1100 of the logs came from 128.231.171.123, which is registered as follows at www.arin.net

National Institutes of Health (NET-NIH-NET)
9000 Rockville Pike
Bethesda, MD 20892
Netname: NIHNET-1
Netnumber: 128.231.0.0

Coordinator:
Fajman, Roger (RF57-ARIN) rfajman@NIH.GOV
(301) 402-4265 (FAX) (301) 480-6041 (FAX) (301)480-6241

It may be worth contacting the owner to see if they might have experienced a compromise on 128.231.171.123 or a device behind it.

A quick look through the Type 2 logs showed a few interesting things.

First was the number of strange flag combinations (Out of Spec Packets) There were 184 unique bad flag patterns. The top ten are listed for reference here in the table on the left. The largest number of OOS packets are of course SYN,FIN's which translate nicely into SYN,FIN scans. A check for SYN,ACK turned up empty (which in this case is a good sign) In the Type 2 logs we find 489 unique addresses producing these packets with Out of Spec flag combinations. A table containing the top 10 IP's is on the right. The most offending IP is registered as follows:

inetnum: 193.173.160.0 - 193.173.192.255
netname: SCARAMEA
descr: e-commerce
descr: internet service provider

country: NL
admin-c: AAH13-RIPE
tech-c: AAH13-RIPE
status: ASSIGNED PA
notify: ipregistry@inet.kpn.com
mnt-by: AS1136-MNT
changed: symonel@inet.unisource.nl 20000127
changed: symonel@inet.unisource.nl 20000316
source: RIPE

A second sort on the Type 2 logs, by Destination Port, shows some interesting data. Port 21, the FTP control port is by far the most popular. Next is 53, which corresponds to DNS. This is most certainly made up of legitimate traffic like DNS queries. Port 98 is the TAC News port.

Port 27274 can be a SubSeven version 2 Trojan scan. Given the large number I would be fairly certain that most of the network has been scanned numerous times for this Trojan. It seems to be very popular at this time.

110 is POP3 mail, and 6970 seems to be the scan of choice from the Korean Information Network. A large portion of these 6970 port scans come from the following block:

inetnum 211.42.0.0 - 211.51.255.255
netname KRNIC-KR-23
descr KRNIC
descr Korea Network Information Center
country KR
admin-c WK1-AP, inverse
tech-c SL119-AP, inverse
remarks KRNIC Allocation Block
remarks Authoritative Information regarding assignments and
remarks allocations made from within this block can also be
remarks queried at whois.nic.or.kr

DPorts	t.Count
21	141591
53	57280
98	34631
27374	23996
110	1570
6970	1374
23	530
31337	473
44767	430
12345	225

Port 23 is standard telnet. 31337 is a standard port for Back Orafice and I'm most certain that a large majority of these are scanning for this. 44767 does not currently show up on my Trojan list - not sure what this traffic represents.

And lastly in the top 10 we have 225 attempts on port 12345 which has any number of names including GabanBus, My Pics, NetBus, Pie Bill Gates, Whack Job, and X-bill.

Once again, as I looked through the logs I specifically searched for any SYN,ACK's and did not find any. This may mean that as of today we do not have a Trojan that has responded during the recording of the logs that were taken. Virus scanners are a very good investment and should be followed up on if not already installed.

One more interesting thing I found after sorting the Type 2 logs by Destination Port is the following:

06/30 11 23 41 208.147.89.163 0 -> MY.NET.156.117 0 UDP

This is a UDP port 0 attempt. This comes out of the following block:

PROGRESSIVE NETWORKS (NETBLK-CW-208-147-89)
1111 THIRD AVENUE
SEATTLE, WA 98101
US

Assignment 4 - Analysis Process

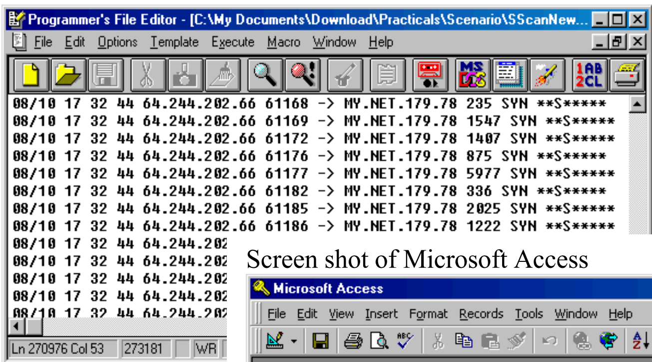
In the words of an Internationally renowned Intrusion Analyst "Start…Programs…Accessories…". Good advice as I now can personally attest to. Since this is very new to me I had to resort to tools that I was aware of (and one I'm very glad I learned about).

My first attempt was to load the provided text files into a Microsoft Excel spread sheet. It might have worked fine if there had not been so many lines in the log files. Excel has a limitation of under 66000 lines of data per sheet. This posed a little bit of a problem when you grasp that in the Assignment 3 of this practical that there were more than 350,000 lines of logs - ouch!

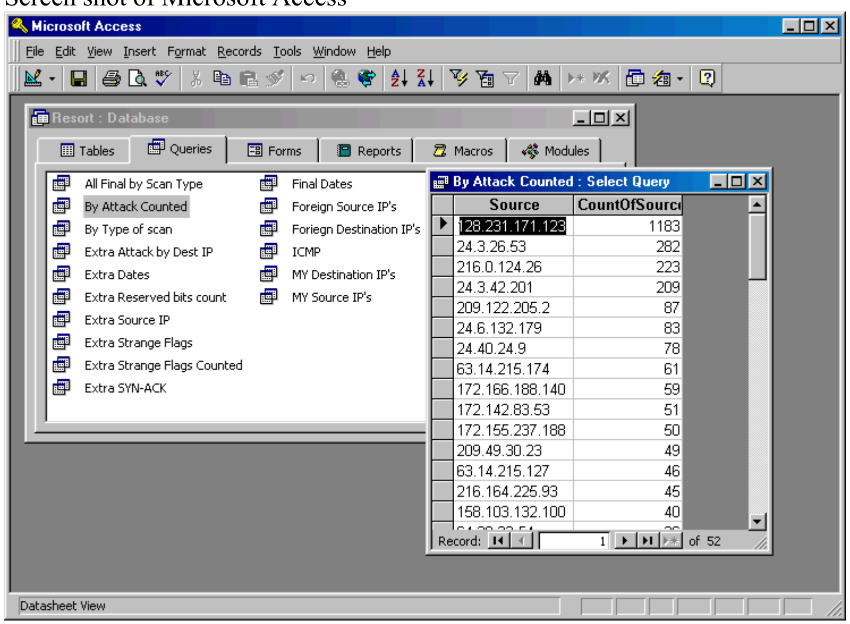
I certainly gave it the old college try with Excel spending a couple of nights adjusting the spacing in the first type of log file so that I could get all the fields to line up. I even split the first type of files into two sheets. Then I decided that I needed to get all the logs in the same place so I could massage them any way I wanted. It was at this point that I also realized that the two different forms of log files would not match up at all.

With this in mind, I imported the first group of files into an Microsoft Access database (authors note: boy do I wish that Microsoft would include all the same kind of functions in ALL of its programs. For example try and parse on a specific character position in a field in Access - Excel has the command - Access doesn't - ARG). Anyway, I looked at the second type of files and realized that they needed some significant work to attempt to match them up with the first type of files. After fooling around with Excel and Word for a few nights I decided I needed another tool to accomplish the task. Enter PFE.

Screen shot of PFE

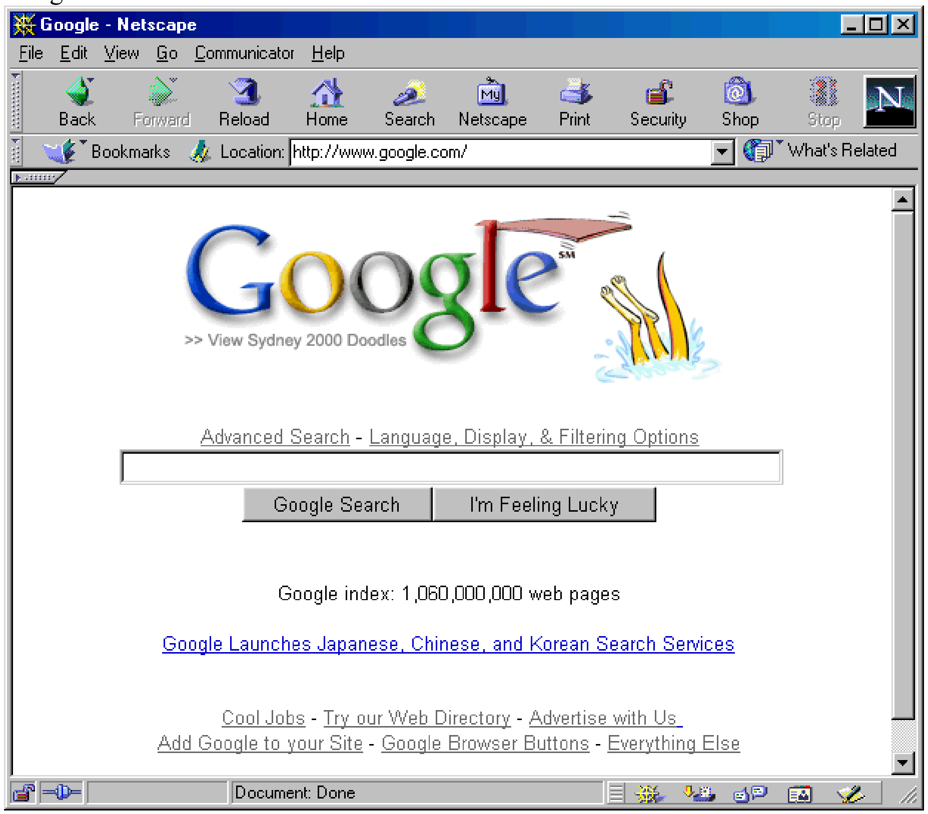


Screen shot of Microsoft Access



This is a great little 32 bit file editor (see http://www.lancs.ac.uk/people/cpaap/pfe/default.htm & Screen Shot). This thing saved my bacon. I used it to replace characters and put spaces where I needed them for importing into Access.

Google Screen Shot



Now I have two separate Access tables that I can search and sort and massage any way I want to look at the data. And that's exactly what I did. For each item I would start by limiting the chosen records to the specific alarm that I was looking at. I would then sort by different fields, including Source IP, Destination IP, Port, and Date and Time. In each case I would stroll though the data looking for something that stuck out. And in most cases it did. Other times I would check and see if there was a correlating type 2 log and would go through those sorted by time and date to attempt to match up some records. One specific case was helpful in the Queso Fingerprint explanation. Other times I would look at the traffic UN-filtered to see what it looked like in the raw log (which was sorted by date and time). Simple proximity to other traffic was helpful in some cases.

I also read a number of the other practicals as was suggested in the description of this assignment. This was very helpful in that I sort of had a second bearing on where I needed to go and how to get there.

Of course the GIAC/SANS site was great to look things up and help fill in the very large gaps in my brain with respect to IDS.

Another tool I used is as simple as a search engine. I don't know what the majority of people use but I am very surprised to find how many people have not heard of Google (www.google.com). I am a huge supporter of this (commercial free front page) search engine. If you have not been here you absolutely must check it out. I have found more things using this engine than any of the other big name search engines.

Other than that, it was mostly sheer determination not to let this practical beat me that pushed me through it. It is much harder than it looks and I have a much greater appreciation for Intrusion Detection people. They are out on the edge of all this stuff - all of the time.

I'll see you there.

-= End of Transmission =-