Global Information Assurance Certification Paper

Practical Assignment fir GIAC GCIA Certification

Milind Saraph
Office of Information Technologies
University of Notre Dame

November 21, 2000

Contents

1.0 Purpose

This report is submitted to fulfill partial requirement of GIAC GCIA certification (SANS Network Security, Monterey, CA - Oct 2000). in Oct 2000. The reports has three sections. The first section analyzes the 4 network detects, the second section evaluates a network attack and the third section analyses the data provided.

2.0 Network Detects

All the network detects were provided by Curt Freeland, GCIA from the University of Notre Dame. He is assisting a local educational institution in setting perimeter monitoring and defense. He has recently set up Snort 1.6 and Shadow to monitor the traffic coming from the ISP. He is still fine tuning the setup. Incidentally his name is still spelled as *Freedland* on GIAC web page.

2.1 Detect 1

Trace

The following trace spanning 20 seconds shows probes from the the IP number 128.2.168.45.1817 which resolves to wkh.res.cmu.edu, most likely a student workstation in a dormitory.

```
07:23:02.201177 128.2.168.45.1817 > MY.NET.5.216.27374: S 46091013:46091013(0) win 8192  (DF) [tos 0
07:23:02.201480 128.2.168.45.1818 > MY.NET.5.216.12345: S 46091013:46091013(0) win 8192  (DF)
07:23:02.201804 128.2.168.45.1819 > MY.NET.5.216.139: S 46091013:46091013(0) win 8192  (DF)
07:23:05.164496 128.2.168.45.1818 > MY.NET.5.216.12345: S 46091013:46091013(0) win 8192  (DF)
07:23:05.164606 128.2.168.45.1819 > MY.NET.5.216.139: S 46091013:46091013(0) win 8192  (DF)
07:23:05.166778 128.2.168.45.1817 > MY.NET.5.216.27374: S 46091013:46091013(0) win 8192  (DF)
07:23:11.179669 128.2.168.45.1818 > MY.NET.5.216.12345: S 46091013:46091013(0) win 8192  (DF)
07:23:11.179979 128.2.168.45.1819 > MY.NET.5.216.139: S 46091013:46091013(0) win 8192  (DF)
07:23:11.180238 128.2.168.45.1817 > MY.NET.5.216.27374: S 46091013:46091013(0) w in 8192  (DF)
07:23:23.402022 128.2.168.45.1818 > MY.NET.5.216.12345: S 46091013:46091013(0) win 8192  (DF)
07:23:23.402307 128.2.168.45.1819 > MY.NET.5.216.139: S 46091013:46091013(0) win 8192  (DF)
07:23:23.402646 128.2.168.45.1817 > MY.NET.5.216.27374: S 46091013:46091013(0) win 8192  (DF)
```

A similar trace also spanning about twenty seconds from the IP number 24.42.102.22, which resolves to cr101170-a.ym1.on.wave.home.com shows up another day.

```
09:28:21.427914 24.42.102.22.3138 > MY.NET.41.112.27374: S 318106877:318106877(0) win 8192  (DF)
09:28:21.441662 24.42.102.22.3139 > MY.NET.41.112.12345: S 318106877:318106877(0) win 8192  (DF)
09:28:21.452162 24.42.102.22.3140 > MY.NET.41.112.139: S 318106877:318106877(0) win 8192  (DF)
09:28:24.331398 24.42.102.22.3139 > MY.NET.41.112.12345: S 318106877:318106877(0) win 8192  (DF)
09:28:24.355687 24.42.102.22.3140 > MY.NET.41.112.139: S 318106877:318106877(0) win 8192  (DF)
09:28:24.403806 24.42.102.22.3138 > MY.NET.41.112.27374: S 318106877:318106877(0) win 8192  (DF)
09:28:30.363283 24.42.102.22.3139 > MY.NET.41.112.12345: S 318106877:318106877(0) win 8192  (DF)
09:28:30.392557 24.42.102.22.3140 > MY.NET.41.112.139: S 318106877:318106877(0) win 8192  (DF)
09:28:30.415083 24.42.102.22.3138 > MY.NET.41.112.27374: S 318106877:318106877(0) win 8192  (DF)
09:28:42.367368 24.42.102.22.3139 > MY.NET.41.112.12345: S 318106877:318106877(0) win 8192  (DF)
09:28:42.379585 24.42.102.22.3140 > MY.NET.41.112.139: S 318106877:318106877(0) win 8192  (DF)
09:28:42.462937 24.42.102.22.3138 > MY.NET.41.112.27374: S 318106877:318106877(0) win 8192  (DF)
```

Source of Trace

This trace was provided by Curt Freeland, GCIA (for details see beginning of Section 2)

Detect generated by

The detect was generated by Snort Version 1.6.

Probability the Source was spoofed

It seems unlikely that the source address is spoofed. The sources seem to be interested in knowing whether the vulnerability exists.

Description of Attack

Since a similar pattern is seen from two different sites, it is likely that a kind of automated tool/script is being used to probe.

Attack Mechanism

The sequence number appears to be constant during a session. Three consecutive source port numbers are used and the starting source port number appears configurable.

Any TCP source port to destination port 27374 is a signature of SubSeven scan (CAN-1999-0660). Any TCP source port to destination port 12345 is NetBus getinfo request (CAN-1999-0660). A destination port of 139 is most likely being used for finding unprotected shares and files on Windows systems.

Correlation

I have not had time do check this.

Evidence of Active Targetting

Obviously it is a probe, it is reasonable to expect an attack if it succeeds.

Severity

- Criticality = 1 (This is a non targeted probe looking for vulnerable for Windows hosts).
- Lethality = 3 (There are no Windows running critical infrastructure services).
- System Countermeasures = 3 (It is probable that the majority of Windows hosts are behind in patches).
- Network Countermeasures = 3 (The filtering is performed on a CISCO router, the filter rules are in a state of flux).

- Severity = (Criticality metric + Lethality metric) - Countermeasures.

The Severity is less than zero.

Defensive Recommendation

The system countermeasures for many Windows systems on the campus at this time are unknown. All the unprotected shares and files on Windows system need to be identified and closed.

Possible Multiple Choice Question

The trace shows:

- (a) NetBus scan
- (b) SubSeven scan
- (c) Attempt to discover open files/shares in Windows machines
- (d) All of the above.

The correct is answer is (d).

2.2 Detect 2

Trace

The following partial listing of a trace shows two hosts sending the packets to non-existent hosts. This has been going on for last three months - a cease and desist request was ineffective.

```
00:53:09.476650 210.118.175.2.49215 > MY.NET.3.201.2099: SR 254082193:254082193(0) win 4096
00:56:52.106126 210.118.175.2.52570 > MY.NET.3.201.2099: S 76844350:76844350(0) win 2048
00:56:56.103402 210.118.175.2.52570 > MY.NET.3.201.2099: S 76844350:76844350(0) win 2048
00:57:00.018882 210.118.175.2.52570 > MY.NET.3.201.2099: S 76844350:76844350(0) win 2048
00:57:03.979997 210.118.175.2.52570 > MY.NET.3.201.2099: S 76844350:76844350(0) win 2048
00:57:07.859617 210.118.175.2.52570 > MY.NET.3.201.2099: S 76844350:76844350(0) win 2048
00:57:11.854059 210.118.175.2.52570 > MY.NET.3.201.2099: S 76844350:76844350(0) win 2048

00:04:39.471805 210.118.175.17.12254 > MY.NET.3.201.2099: SR 11592194:11592194(0) win 2048
00:04:41.698344 210.118.175.17.12498 > MY.NET.3.201.2099: S 99411462:99411462(0) win 2048
00:04:45.641395 210.118.175.17.12498 > MY.NET.3.201.2099: S 99411462:99411462(0) win 2048
00:04:49.567009 210.118.175.17.12498 > MY.NET.3.201.2099: S 99411462:99411462(0) win 2048
00:04:53.543811 210.118.175.17.12498 > MY.NET.3.201.2099: S 99411462:99411462(0) win 2048
00:04:57.486737 210.118.175.17.12498 > MY.NET.3.201.2099: S 99411462:99411462(0) win 2048
```

The packets contents for two packets are shown below.

```
Using device /dev/le (promiscuous mode)
ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 1 arrived at 12:16:18.81
ETHER:  Packet size = 60 bytes
ETHER:  Destination = 0:10:f6:b7:80:0,
ETHER:  Source      = 0:30:80:da:d3:20,
```

```
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 44 bytes
IP:   Identification = 2522
IP:   Flags = 0x0
IP:         .0.. .... = may fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 44 seconds/hops
IP:   Protocol = 6 (TCP)
IP:   Header checksum = 6c6c
IP:   Source address = 210.118.175.17, 210.118.175.17
IP:   Destination address = MY.NET.3.201, MY.NET.3.201
IP:   No options
IP:
TCP:  ----- TCP Header -----
TCP:
TCP:  Source port = 21160
TCP:  Destination port = 2099
TCP:  Sequence number = 237273605
TCP:  Acknowledgement number = 1
TCP:  Data offset = 24 bytes
TCP:  Flags = 0x02
TCP:        ..0. .... = No urgent pointer
TCP:        ...0 .... = No acknowledgement
TCP:        .... 0... = No push
TCP:        .... .0.. = No reset
TCP:        .... ..1. = Syn
TCP:        .... ...0 = No Fin
TCP:  Window = 2048
TCP:  Checksum = 0x8c9a
TCP:  Urgent pointer = 0
TCP:  Options: (4 bytes)
TCP:    - Maximum segment size = 1460 bytes
TCP:


ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 2 arrived at 12:16:22.78
ETHER:  Packet size = 60 bytes
ETHER:  Destination = 0:10:f6:b7:80:0,
ETHER:  Source      = 0:30:80:da:d3:20,
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 44 bytes
IP:   Identification = 2523
IP:   Flags = 0x0
IP:         .0.. .... = may fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 44 seconds/hops
IP:   Protocol = 6 (TCP)
IP:   Header checksum = 6c6b
IP:   Source address = 210.118.175.17, 210.118.175.17
IP:   Destination address = MY.NET.3.201, MY.NET.3.201
IP:   No options
```

```
IP:
TCP:  ----- TCP Header -----
TCP:
TCP:  Source port = 21160
TCP:  Destination port = 2099
TCP:  Sequence number = 237273605
TCP:  Acknowledgement number = 1
TCP:  Data offset = 24 bytes
TCP:  Flags = 0x02
TCP:        ..0. .... = No urgent pointer
TCP:        ...0 .... = No acknowledgement
TCP:        .... 0... = No push
TCP:        .... .0.. = No reset
TCP:        .... ..1. = Syn
TCP:        .... ...0 = No Fin
TCP:  Window = 2048
TCP:  Checksum = 0x8c9a
TCP:  Urgent pointer = 0
TCP:  Options: (4 bytes)
TCP:    - Maximum segment size = 1460 bytes
```

Source of Trace

This trace was provided by Curt Freeland, GCIA (for details see beginning of Section 2)

Detect generated by

The detect was generated by Snort Version 1.6.

Probability the Source was spoofed

It is possible that the source IP address is spoofed.

Description of Attack

From the data, it is unclear whether it is a probe or an attack.

Attack Mechanism

The sequence numbers appears constant during a session. The source port and destination ports and IP numbers stay the same. It appears that the packets have been crafted.

A search on ports TCP source ports and destination ports was unproductive.

Correlation

I have not had time to do this.

Evidence of Active Targetting

It is certainly active targeting, although of non-existent hosts. I have no explanation for this.

Severity

- Criticality = 0 (This is a targeted probe/attack, albeit non-existent hosts)
- Lethality = 1 (Even if the packets reach the hosts, the purpose is unknown) services).
- System Countermeasures = 3 (There are no hosts at the target IP address)
- Network Countermeasures = 3 (The filtering is performed on a CISCO router, the filter rules are in a state of flux).

- Severity = (Criticality metric + Lethality metric) - Countermeasures.

The Severity is less than zero.

Defensive Recommendation

Due to the long duration the source IP numbers may be blocked.

Possible Multiple Choice Question

For the trace,the pattern for which has repeated for 3 months

- (a) This may be a probe
- (b) This may be an attack
- (c) The IP number is spoofed
- (d) No coclusive statement can be made.

The correct is answer is (d).

2.3 Detect 3

Trace

The following trace shows probe of many popular Windows vulnerabilities. This repeats a few times.

```
[**] BACKDOOR ATTEMPT-Netbus/GabanBus [**]
 09/27-16:05:13.323082 0:30:80:DA:D3:20 -> 0:10:F6:B7:80:0 type:0x800 len:0x3E
 200.53.178.2:3257 -> 147.53.65.110:12345 TCP TTL:51 TOS:0x0 ID:4799
 **S***** Seq: 0x1059F5D Ack: 0x0 Win: 0x2000
 TCP Options => MSS: 536 NOP NOP SackOK

 [**] BACKDOOR ATTEMPT-Netbus/GabanBus [**]
 09/27-16:05:13.323133 0:30:80:DA:D3:20 -> 0:10:F6:B7:80:0 type:0x800 len:0x3E
 200.53.178.2:3258 -> 147.53.65.110:12346 TCP TTL:51 TOS:0x0 ID:5055
 **S***** Seq: 0x1059F61 Ack: 0x0 Win: 0x2000
 TCP Options => MSS: 536 NOP NOP SackOK

 [**] BACKDOOR ATTEMPT-Backorifice [**]
 09/27-16:05:13.323183 0:30:80:DA:D3:20 -> 0:10:F6:B7:80:0 type:0x800 len:0x3E
 200.53.178.2:3259 -> 147.53.65.110:31337 TCP TTL:51 TOS:0x0 ID:5311
 **S***** Seq: 0x1059F66 Ack: 0x0 Win: 0x2000
 TCP Options => MSS: 536 NOP NOP SackOK

 [**] MISC-WinGate-1080-Attempt [**]
 09/27-16:05:13.341542 0:30:80:DA:D3:20 -> 0:10:F6:B7:80:0 type:0x800 len:0x3E
 200.53.178.2:3260 -> 147.53.65.110:1080 TCP TTL:51 TOS:0x0 ID:5567
 **S***** Seq: 0x1059F8E Ack: 0x0 Win: 0x2000
 TCP Options => MSS: 536 NOP NOP SackOK
```

Source of Trace

This trace was provided by Curt Freeland, GCIA (for details see beginning of Section 2)

Detect generated by

The detect was generated by Snort Version 1.6.

Probability the Source was spoofed

The IP address is not spoofed, it resolves to dialupdig2.ifxnw.com.mx. The attacker is attempting to exploit known Windows vulnerabilities, possibly some kind of script kiddie.

Description of Attack

Netbus/GabanBus and BackOrifice (CAN-1999-0660) are similar tools which allow a hacker to remotely control Windows 95/98 machine connected to a network. The WinGate attempts (CVE-1999-0291) are most likely is a probe to find unpassworded WinGate proxy installation.

Attack Mechanism

Most likely this is a script kiddie.

Correlation

I have not had time to do this.

Evidence of Active Targetting

The attacker appears to know that a Windows host resides that IP address. This points to the possibility that the attacker may have mapped, at least partially the target network.

Severity

- Criticality = 2 (This is a targeted attack, the attacker knows that a Windows machine is at the target IP address. The Windows host is providing some services to a workgroup).
- Lethality = 4 (Can get control of the host)
- System Countermeasures = 3 (Not uptodate in patches)
- Network Countermeasures = 3 (The filtering is performed on a CISCO router, the filter rules are in a state of flux).

- Severity = (Criticality metric + Lethality metric) - Countermeasures.

The Severity is zero.

Defensive Recommendation

The probes have been going on for a long time, the IP number may be blocked.

Possible Multiple Choice Question

For the trace above (remove the alerts):

- (a) Netbus attack
- (b) Back Orifice
- (c) WinGate
- (d) All of the above

The correct is answer is (d).

2.4 Detect 4

Trace

The following trace shows a walk of class B IP address space probing for portmapper on Unix hosts.

```
10923:18:44:38.635070 62.100.65.8.4834 > MY.NET.0.1.111: S 2540917915:2540917915(0) win 32120  (DF)
10923:18:44:38.652517 62.100.65.8.4835 > MY.NET.0.2.111: S 2545278974:2545278974(0) win 32120  (DF)
10923:18:44:38.669458 62.100.65.8.4836 > MY.NET.0.3.111: S 2534790607:2534790607(0) win 32120  (DF)
```

```
10923:18:44:38.669548 62.100.65.8.4837 > MY.NET.0.4.111: S 2535608405:2535608405(0) win 32120  (DF)
10923:18:44:38.686077 62.100.65.8.4838 > MY.NET.0.5.111: S 2533814712:2533814712(0) win 32120  (DF)
10923:18:44:38.694322 62.100.65.8.4839 > MY.NET.0.6.111: S 2544779869:2544779869(0) win 32120  (DF)
10923:18:44:38.694421 62.100.65.8.4840 > MY.NET.0.7.111: S 2547905877:2547905877(0) win 32120  (DF)
10923:18:44:38.694517 62.100.65.8.4841 > MY.NET.0.8.111: S 2533860811:2533860811(0) win 32120  (DF)
10923:18:44:38.694613 62.100.65.8.4842 > MY.NET.0.9.111: S 2532810550:2532810550(0) win 32120  (DF)
10923:18:44:38.694713 62.100.65.8.4843 > MY.NET.0.10.111: S 2542689605:2542689605(0) win 32120  (DF)
10923:18:44:38.694940 62.100.65.8.4844 > MY.NET.0.11.111: S 2541587940:2541587940(0) win 32120  (DF)
10923:18:44:38.695330 62.100.65.8.4845 > MY.NET.0.12.111: S 2536658761:2536658761(0) win 32120  (DF)
10923:18:44:38.695655 62.100.65.8.4846 > MY.NET.0.13.111: S 2534478868:2534478868(0) win 32120  (DF)
<snipped>
```

Source of Trace

This trace was provided by Curt Freeland, GCIA (for details see beginning of Section 2)

Detect generated by

The detect was generated by Snort Version 1.6.

Probability the Source was spoofed

The IP address resolves to dns2.nsoft.it, the name suggests that it is a name server. There are two possibilities: first is that the name server is compromised and launching a noisy scan, seond the IP address is spoofed and the hacker does not really care about the noisy scan.

Description of Attack

A noisy scan for portmapper on Unix hosts.

Attack Mechanism

A probe, if successful to be followed by an attack.

Correlation

I have not had time to do this.

Evidence of Active Targetting

There is active targeting, the hacker is performing a portmapper scan.

Severity

- Criticality = 3 (This is a targeted probe for Unix hosts. Many infrastructure services run on Unix hosts)
- Lethality = 4 (Can get control of the host)
- System Countermeasures = 3 (Not uptodate in patches)
- Network Countermeasures = 3 (The filtering is performed on a CISCO router, the filter rules are in a state of flux).

- Severity = (Criticality metric + Lethality metric) - Countermeasures.

The Severity is one.

Defensive Recommendation

Block all traffic to port 111.

Possible Multiple Choice Question

For the trace above (remove the alerts):

- (a) Unix Portmapper scan
- (b) Windows Portmapper scan
- (c) News traffic
- (d) None of the above

The correct is answer is (a).

3.0 Evaluation of Dsniff

Introduction

Ethernet switches are used to help protect network traffic from snooping. There are two ways of snooping traffic on a switched segment - one is to place sniffer on the segment itself and the second is to instruct a remote machine to send traffic meant for a target segment to your segment, which then can be examined and forwarded to the target host. The dsniff package has a utility called arpredirect to do this to do this. This section presents a brief overview of the dsniff package. *In an academic environment these type of attacks on a LAN are important, especiall in a dormitory type setting.*

Dsniff package is an administrative or hacker toolset, depending on the perspective It is a good LAN auditing tool and could be used as a sniffer or denial of service tool in a switched LAN environment. It is written by Dug Song http://www.monkey.org/ dugsong and available at this web site http://www.monkey.org/ dugsong/dsniff. Articles about dsniff have appeared in Windows2000 and InfoWorld magazines.

The dsniff package has the following tools:

- *arpredirect* works by sending a forged arp packet to the target system instructing it to change its default gateway to the attacking system. All the traffic destined for default gateway is now sent to the attacking system which can examine the traffic and then forward it to the original destination. The attacking system needs to have Kernel-level IP forwarding turned or use fragrouter to perform packet forwarding on a Linux system. Once the traffic from the target system is received, it can be examined, analyzed and data such as passwords, mail, URLs can be extracted and logged.

- *dsniff* captures cleartext and poorly encrypted passwords from RIP, LDAP, YP, X11 CVS, AIM, ICQ, Napster, Microsoft SMB etc. and other protocols.
- *filesnarf* saves files sniffed from NFS traffic in a directory.
- *macof* floods the local network with random MAC addresses causing some switches to fail open. This could be used for denial of service.
- *mailsnarf* assembles e-mail traffic and displays in near real time.
- *tcpkill* kills specified in-progress TCP connections.
- *tcpnice* slows down TCP connections by sending ICMP source quench replies and by advertising small windows.
- *urlsnarf* captures and outputs URLs from HTTP traffic in CLF (Common Log Format) suitable for off-line processing with we log analysis tool.
- *webspy* does the same thing as urlsnarf but displays data in real time in a Netscape browser window. As the target surfs the web, the browser is updated.

2.2 Description of Dsniff Package

Dsniff utilities are built on top *lipcap*, *libnet* and *libnids*. The following sections describe each of these libraries. *As a synopsis the libpcap is used to capture packets, libnet is used craft and inject packets and libnids is used to provide session abstraction.*

Libpcap

Libpcap is a system-independent interface for user-level packet capture. Tcpdump a tool for network monitoring and data acquisition is based on this library. The latest versions of this library are available from http://www.tcpdump.org.

Libnet

Libnet or *Lipwrite* library provides a simple API to craft and write arbitrary network packets both at the IP level and link layer level. Unfortunately at this time it supports only IPv4 and not IPv6. A user manual can be found at http://www.packetfactory.net/libnet/manual.

Libnids

The *nids* part of *libnids* stands for Network Intrusion Detection System and is built on top *lipcap* and *libnet*. *libnids* provides functionality to assemble TCP segments into a TCP stream, assemble IP fragments (similar to Linux 2.0.36 kernels) and detect port scans. It correctly handles all the attacks implemented in fragrouter written by Dug Song.

Installation and Testing

I downloaded the following versions of the libraries:

- libpcap-0.5
- Libnet-1.0.1b
- libnids-1.14
- dsniff-2.2

I also needed to install Berkeley database library, I chose db-2.7.7 in *compat185* mode for proper installation of *libnids-1.14*. From the configure script it appears that *db* may not be necessary, however I could not build it without *db*.

I first tried to install *dsniff* on a Solaris 7 workstation. The installation for *libnids* died during configuration. Rather than spend time trying to fix it, I set up Linux RH7.0 on an old PC and installed *dsniff* on it. I tried *dsniff* for ftp and telnet, it works really well. *mailsnarf* and *urlsnarf* are also very easy to use. I did not want to try *arpredirect*, *macof*, *tcpkill* and *tcpnice* on a production network. I do plan to try these on a a test LAN with *arpwatch* to examine the traffic. I did not try *filesnarf* and *webspy* but I have no reason to believe they would not work as advertised. I have not included any screen snapshots because they do not add anything to what I have said in words.

It would have interesting to setup *arpredirect* and install arpwatch

2.3 Defensive Measures

There are two possible measures against the Dsniff type attacks. One is use tools like *arpwatch* to watch arp entries in a LAN and generate alerts when these change. The second one is encryption of the data.

2.4 Miscellaneous

I examined the source code for *dsniff*, it is clean and compact. I have no idea about the performance of *libnids*. It will be interesting to see how it performs on a busy network.

4.0 "Analyze this" Scenario

The analysis of data would have been much easier with the following information:

- Approximate number of hosts in the network.
- Brief description of network topology

- Hostnames and OS of infrastructure servers such as DNS.
- Location of Snort sensor and snort-rules used.

Without this information and limited time, it is difficult to do a comprehensive analysis. In any intrusion analysis the resources in terms of staff time have to prioritized based on the importance of services and machines.

The first step is to organize data in a format which makes it easy to recognize patterns. The second step is to focus on probes, scans and attacks against infrastructure hosts such DNS, Mail and Web servers. The third is to check whether the OS and patch levels on these critical hosts are uptodate. Finally if there is any time left, pay attention to other non-critical and desktop hosts.

4.1 Description of Data

The following table summarizes the availability of the data for about a month starting Aug 11,2000 and ending Sep 14, 2000. The duration of data for alert and scan files does not quite 24 hours but is reasonably close. The availability times does not add anything to the analysis and is not shown.

There is no information about snort rules used, it is difficult to figure out what passed through without triggering any alerts.

Table 1: Description of Data

Date	Alert File	Scan File	?? File
Aug 11	SnortAle.txt		
Aug 15	SnortA2.txt	SnortSca.txt	
Aug 16	SnortA4.txt	SnortS3.txt	
Aug 17	SnortA3.txt	SnortS2.txt	
Aug 18	SnortA5.txt	SnortS6.txt	
Aug 19	SnortA7.txt		
Aug 20	SnortA6.txt		
Aug 28		SnortS7.txt	SOOS.txt
Aug 29			SOOS2.txt
Aug 31			SOOS3.txt
Sep 01			SOOS12.txt
Sep 02	SnortA11.txt	SnortS9.txt	SOOS4.txt
Sep 03	SnortA12.txt	SnortS10.txt	SOOS5.txt
Sep 04		SnortS16.txt	SOOS17.txt
Sep 05	SnortA14.txt	SnortS11.txt	SOOS6.txt
Sep 06	SnortA15.txt	SnortS12.txt	SOOS7.txt
Sep 07	SnortA16.txt	SnortS13.txt	SOOS8.txt
Sep 08	SnortA17.txt	SnortS14.txt	SOOS9.txt
Sep 09	SnortA18.txt	SnortS15.txt	SOOS11.txt
Sep 10	SnortA19.txt	SnortS17.txt	SOOS18.txt
Sep 11	SnortA20.txt	SnortS18.txt	SOOS19.txt
Sep 12	SnortA22.txt		SOOS20.txt
Sep 13	SnortA23.txt	SnortS19.txt	SOOS21.txt
Sep 14	SnortA24.txt	SnortS20.txt	SOOS22.txt

- The alert files SnortA20.txt and SnortA21.txt are the same, the file SnortA21.txt is dropped.

- The scan files SnortS20.txt and SnortS21.txt are the same, the file SnortS21.txt is dropped.
- The files SOOS9.txt and SOO10.txt are the same, the file SOOS10.txt is dropped.
- It is not clear what the "OOS" stands for SOOS files. Looking at SOOS11.txt, it appears that these are mutant (improper combination of flags) packets which may have been crafted.

Snortsnarf was used to generate a summary of all the alerts. There are two minor details: MY.NET needs to be replaced with IP subnet number and the initial lines befores alerts need to be stripped. Here is a snapshot of snortsnarf summary (I tried to use snort_stat but had some difficulty getting it going).

```
Snortsnarf: Snort signatures in SnortAle.txt et al

33366 alerts processed.

Files included:
```

```
      SnortAle.txt
      SnortA2.txt
      SnortA4.txt
      SnortA3.txt
      SnortA5.txt
      SnortA7.txt
      SnortA6.txt
      SnortA11.txt
      SnortA12.txt
      SnortA14.txt
      SnortA15.txt
      SnortA16.txt
      SnortA17.txt
      SnortA18.txt
      SnortA19.txt
      SnortA20.txt
      SnortA22.txt
      SnortA23.txt
      SnortA24.txt

Earliest alert at 00:33:46.103627 on 08/11
Latest alert at 23:21:39.338983 on 09/14
```

Table 2: Summary of Alerts

Signature	Alerts	Sources	Destinations
Happy 99 Virus	2	2	2
Possible wu-ftpd exploit - GIAC000623	2	1	2
site exec - Possible wu-ftpd exploit - GIAC000623	4	1	3
TCP SMTP Source Port traffic	8	2	2
Tiny Fragments - Possible Hostile Activity	9	5	7
Queso fingerprint	11	6	8
Probable NMAP fingerprint attempt	33	5	24
External RPC call	38	5	3
Null scan!	54	21	31
SUNRPC highport access!	62	5	3
NMAP TCP ping!	92	10	37
SMB Name Wildcard	316	17	15
SNMP public access	797	16	1
Attempted Sun RPC high port access	1756	8	11
SYN-FIN scan!	2801	4	2756
WinGate 1080 Attempt	3515	330	1905
Watchlist 000220 IL-ISDNNET-990517	5264	19	21
Watchlist 000222 NET-NCFC	18602	45	19

4.2 Coarse Analysis of Alerts

The alerts are categorized as follows:

- Reconnaissance i.e. attempt to map the network, including discovery of active hosts and the operating system. Queso, NMAP fingerprinting, Null and SYN-FIN scans, SMB Name Wildcard fall into this category.

- Probes for vulnerabilities i.e. find vulnerable or misconfigured hosts. WinGate 1080 Attempts, attempted Sun RPC high port access and SMB Wildcard fall into this category.

- Attempted Attacks i.e. attempts to compromise or access systems. SNMP Public access, SUNRPC highport access, wu-ftpd exploits, external RPC calls fall into this category.

- Potentially successful attacks. The SUNRPC high port access and external RPC calls fall into this category. To ascertain the success or failure of Attacks, more information about the hosts, their operating system, the services they are running is helpful.

- Watchlists i.e. traffic from certain sites. The NET-NCFC watchlist refers to traffic from Chinese Academy of Sciences domain, this was determined using http://www.samspade.org. From the name IL-ISDNNET appears to refer to some ISDN network in Illinois.

4.3 Analysis of Two Potential Compromise Alerts

In this section two alerts are analyzed, SUNRPC high port access and external RPC calls. To analyze data for more than 24 hours it is necessary to load the data in a database and map out attacking host timelines. While hunting for a potential compromise, depending on the target host operating system, all the scans and alerts unrelated to the OS can be ignored, for instance all the alerts related to Windows SMB for a Unix host can be ignored unless it is running a SAMBA server.

The external RPC calls are targeted towards three hosts MY.NET.6.15, NY.NET.100.130 and MY.NET.15.127. For all the target hosts the destination port is 111 i.e. portmapper port. The attacker is trying to determine the RPC services offered. Of the five sources two resolve to flutter.mit.edu, and frankenstein.nwnii.com and the other three can not be resolved. There does not appear to be any coordination between these sources. Just after probing the targets, flutter.mit.edu also performed a SYN-FIN scan. Apart from this there is no other activity from these source hosts.

The high port access alert was generated for three internal hosts on port 32771. The hosts are MY.NET.211.2, MY.NET.6.15 and MY.NET.210.2. Many organizations locate their infrastructure related servers at low IP numbers. It will be interesting to see what services MY.NET.211.2 offers. According to http://advice.networkice.com/advice/Exploits/Ports/32771/default.htm the port 32771 is used by SunOS and some other Unix machines for ghost portmapper services.

It appears that both these alerts are similar in the sense that the attacker is trying to get to a "real" or a "ghost" portmapper.

4.4 Detailed Analysis of Alerts for Sep 9, 2000

I picked randomly the date of Sep 9, 2000 for more detailed look at the data.

Table 3: Summary of Alerts for Sep 09, 2000

Signature	Alerts	Sources	Destinations
Null scan!	3	3	2
SMB Name Wildcard	2	2	2
SNMP public access	5	1	1
Attempted Sun RPC high port access	167	2	2
WinGate 1080 Attempt	44	21	22
Watchlist 000220 IL-ISDNNET-990517	612	1	1
Watchlist 000222 NET-NCFC	36	5	5

SMB alerts are from two internal hosts to other internal hosts. Without knowing the location of the Snort sensor it is difficult to say what to make of these. Same is true for SNMP access attempts from one internal host to the other.

There is lot of traffic from Chinese Academy of Sciences to probes for SMTP port, port 113 and some other high ports. It seems very unlikely that community members of Chinese Academy of Sciences may be so interested in our (hypothetical) dotcom enterprise. Most likely it is a hacker's paradise. The information about NET-NCFC watchlist was discovered using http://www.samspade.org.

All of the traffic on IL-ISDNNET-990517 watchlist is from 212.179.66.2 to MY.NET.221.94 port 6699. This appears to be napster traffic.

The host 205.188.153.98 has targeted MY.NET.217.82 port 32771. The host 205.188.153.115 has targeted MY.NET.53.15 port 32771. Most likely it is the same individual looking for ghost portmapper.

There are 21 sources targetting 22 destinations looking for port 1080 a Windows Firewall/proxy.

4.5 Detailed Analysis of Scans for Sep 9, 2000

Snortsnarf was used to generate a summary of scans. In this section only UDP and TCP Syn scans are discussed in detail. There are eight TCP Null scans, 3 TCP FIN scans and about a dozen TCP scans with improper flag settings.

There are a total of 1493 UDP scans from 5 different sources to about thirteen destinations. Out of these thirteen destinations traffic to MY.NET.1.3, MY.NET.1.4 and MY.NET.1.5 seems legitimate except for two packets. A bulk of port scan alerts for UDP traffic are to the hosts MY.NET.213.10, MY.NET.204.166 and MY.NET.204.126 from 63.248.55.245 (ports 7777 and 7778) which appears to be napster traffic. There is some traffic from umbi.umd.edu port 53 (136.160.7.2) to MY.NET.115.115.

There are a total 11610 TPC scans from 5 source targeted towards 9481 destinations, which may be connected, disconnected or non-existent. The five sources are:

Table 4: Summary of TCP scans for Sep 09, 2000

Source	Alerts (sig)	Alerts (total)	Dest (sig)	Dest (total)
206.186.79.9	8159	8164	6796	6799
210.55.227.138	3234	3234	2672	2672
147.208.171.139	187	187	1	1
213.188.8.45	20	20	14	14
203.176.29.200	10	10	1	1

The TCP SYN probes from host ns.arex.com (206.186.79.9) are all directed towards port 53 looking for nameserver. It is unclear as to why this nameserver, if uncompromised is scanning all hosts.

The host pp2-138.world-net.co.nz (210.55.227.138) is systematically scanning hosts for 12346 (NetBus probe) and 27374 (SubSeven) ports.

The host security.norton.com (147.208.171.139) is sending SYN packets to MY.NET.97.230 at various ports, it may be looking for open ports.

The host albatross.fast.no (213.188.8.45) is scanning for ftp port (port 21) on 14 different hosts.

The host ip200.sanmiguel.com.ph (203.176.29.200) is scanning only 1 host MY.NET.97.144 for the following ports:

- 12345 (NetBus Getinfo port)
- 6670 (VocalTec Internet Phone/Deep Throat trojan port)
- 31337 (BackOrifice port)
- 4950, 1001, 21554, 20000, 6400 ports

4.6 Analysis of SOOS11.txt File

```
09/09-02:00:36.041801 24.112.166.228:1050 -> MY.NET.202.202:6699
                      cr642074-a.ym1.on.wave.home.com
09/09-04:17:55.702479 24.6.140.249:0      -> MY.NET.130.190:1241
                      cc337279-a.owml1.md.home.com
09/09-08:03:08.790028 24.6.140.249:1437   -> MY.NET.130.190:20
                      cc337279-a.owml1.md.home.com
09/09-11:18:26.912510 MY.NET.222.250:1638 -> 128.11.68.63:110
                                             (pop.vip.suc.yahoo.com)
09/09-12:03:45.146450 MY.NET.220.142:0    -> 64.14.113.148:1294
09/09-12:11:59.543234 MY.NET.220.142:1296 -> 64.14.113.154:4000
09/09-14:47:52.635007 24.21.252.15:26     -> MY.NET.201.198:6699
                      cc369098-a.jmsil1.sc.home.com
09/09-19:30:21.065626 MY.NET.217.154:218  -> 209.1.224.16:2325
                                             (res6.geocities.yahoo.com)
09/09-21:04:45.229272 MY.NET.205.226:0    -> 207.87.20.98:1066
```

I dont know what to make of this data. Looking at the TCP flags, the packets are mutants i.e. they are not supposed to be this way.

4.7 Tools and Techniques

Other than standard Unix utilities, quick and dirty Perl scripts, the only tool I used is Snortsnarf. I started rolling my own script to do Snortsnarf type analysis - the only difference is I had "object oriented" approach in mind and different abstractions in mind. The script is half-baked in every way and thus not included.

4.8 Conclusions and Recommendations

- There are a number of attempts to search for portmapper both at port 111 and ghost port 32771. Possibly there was information divulged by total of 6 hosts from portmapper. The very first thing to do is check these hosts for OS, whether they are patched and what RPC services they offer. The next step is to audit the system and check the integrity of the binaries for ps, top, ifconfig, login, netstat, ls with lsof to check for open files.

- There are large number of WinGate alerts. I would check for SOCKS proxies and services such as IRCs on internal hosts.

- There are large number of SMB alerts. I would check for Windows hosts having open shares.

- As a minium a filtering router is required for this site to filter out SMB, SNMP, WinGate, RPC etc probes/attacks. Ideally speaking two Intrusion Detection Systems are needed, one before the filtering router and one after that. Depending on the budget, the site can opt for Snort/Shadow combo or get commercial IDS systems. After this is in place and some data is available about what to expect and evaluating what control the IT staff has over the internal hosts and network, the site should consider setting up a firewall.

- There are two possible ways of handling this: one is to rely on internal staff with appropriate training or contract it out. This depends on the size and skill level of internal IT staff.

4.9 Random Thoughts

The whole area of ID analysis reminds me of data spewed by Supercomputers. I strongly believe some novel visualization metaphors, if they already do not exists are needed to make analysis more tractable and easier. There are two directions in which this problem can be tackled. One is to develop some standard database schema to store the alerts, scans and other data. The second is to develop useful abstractions to extract and present the data to the analyst. Standard schema are necessary in order to develop analysis tools which can be shared with other members of the community.

Last Updated: October 22nd, 2016

Upcoming Training

SANS San Diego 2016	San Diego, CA	Oct 23, 2016 - Oct 28, 2016	Live Event
SOS SANS October Singapore 2016	Singapore, Singapore	Oct 24, 2016 - Nov 06, 2016	Live Event
SANS London 2016	London, United Kingdom	Nov 12, 2016 - Nov 21, 2016	Live Event
Community SANS Tysons Corner SEC503	Tysons Corner, VA	Nov 14, 2016 - Nov 19, 2016	Community SANS
Community SANS Indianapolis SEC503	Indianapolis, IN	Nov 14, 2016 - Nov 19, 2016	Community SANS
Community SANS Chicago SEC503	Chicago, IL	Dec 05, 2016 - Dec 10, 2016	Community SANS
SANS Cyber Defense Initiative 2016	Washington, DC	Dec 10, 2016 - Dec 17, 2016	Live Event
Community SANS Albany/Cohoes SEC503	Albany, NY	Dec 12, 2016 - Dec 17, 2016	Community SANS
SANS Security East 2017	New Orleans, LA	Jan 09, 2017 - Jan 14, 2017	Live Event
Community SANS Nashville SEC503	Nashville, TN	Jan 16, 2017 - Jan 21, 2017	Community SANS
SANS Brussels Winter 2017	Brussels, Belgium	Jan 16, 2017 - Jan 21, 2017	Live Event
SANS Oslo 2017	Oslo, Norway	Feb 06, 2017 - Feb 11, 2017	Live Event
Community SANS Las Vegas SEC503	Las Vegas, NV	Feb 06, 2017 - Feb 11, 2017	Community SANS
SANS Secure Japan 2017	Tokyo, Japan	Feb 13, 2017 - Feb 25, 2017	Live Event
SANS Scottsdale 2017	Scottsdale, AZ	Feb 20, 2017 - Feb 25, 2017	Live Event
SANS Secure India 2017	Bangalore, India	Feb 20, 2017 - Mar 14, 2017	Live Event
Community SANS Charleston SEC503	Charleston, SC	Mar 13, 2017 - Mar 18, 2017	Community SANS
SANS Tysons Corner Spring 2017	McLean, VA	Mar 20, 2017 - Mar 25, 2017	Live Event
SANS 2017	Orlando, FL	Apr 07, 2017 - Apr 14, 2017	Live Event
Community SANS New York SEC503	New York, NY	Apr 10, 2017 - Apr 15, 2017	Community SANS
SANS Baltimore Spring 2017	Baltimore, MD	Apr 24, 2017 - Apr 29, 2017	Live Event
SANS Security West 2017	San Diego, CA	May 09, 2017 - May 18, 2017	Live Event
Security Operations Center Summit & Training	Washington, DC	Jun 05, 2017 - Jun 12, 2017	Live Event
SANS Houston 2017	Houston, TX	Jun 05, 2017 - Jun 10, 2017	Live Event
SANS OnDemand	Online	Anytime	Self Paced
SANS SelfStudy	Books & MP3s Only	Anytime	Self Paced