Global Information Assurance Certification Paper

GCIA Training and Certification

Practical Assignment

Version 4.1

Richard Sillito

Tuesday, March 08, 2005

Table of Contents

Executive Summary

There has been a healthy note of pessimism regarding Intrusion Detection Systems and their role in network security. They certainly are easy to install, hard to tune (properly) and easy to ignore. But despite their short comings IDS's still provide functions that other devices simply can't. As an audit and warning system they do and will continue to play an important role in network security.

Evidence of this is obvious as your organization is currently seeing the benefits. After completing this report it was obvious that without this tool your organization would be oblivious to the attacks, scans and other malicious traffic that is happening on your network everyday.

Although this type of network traffic is evident in all Internet connections, the open learning environment of a University seems to attract even more of this activity. This dynamic makes securing these types of networks especially challenging.

But like any good craftsman the right tool is only the start. Along with the tool, care, attention and patients is needed and an IDS is no different. While reviewing the logs from your IDS environment I was able to ascertain the following information:

Strengths:

- Your organization has IDSs in place. This is a step up from many organizations.
- With a sensor on the Internet connection and another on the inside, it is obvious that your organization took some time to think about an effective implementation.
- Your organizations choice of Snort represents a cost effective solution that will be easy to ROI.

Opportunities for improvements:

- IDS Rules evolve over time, an older rule set will be less effective. It is advisable to ensure that your rule set and IDS software is current.
- Some of the rules (ECN aware traffic as an example) are appearing as noisy rules, tuning of the rules would be advisable.
- There is evidence that incidences are not being responded to, detect 3 has been occurring for over a year. An IDS is only as effective as the Analyst behind it. This could be a people resource issue or also a training issue.
- Rules on the firewall appear to be quite loose (again this could be the result of the "Open Nature" of educational resources) more aggressive rules on the firewall would block many of the attacks your organization is

seeing.

Detailed Analysis

Files used in this analysis

Alerts	Size	Scans	Size	Oos	Size
alert.040420	14003972	scans.040420	135564540	oos_report_040416	352256
alert.040421	27966087	scans.040421	271798176	oos_report_040417	327680
alert.040422	26644195	scans.040422	191550994	oos_report_040418	1735680

There were some issues with the files. The oos_report_ files contained a different day than that show in the file name. The dates covered in the files listed above were from 04/20 to 04/22. Also there was some corruption in the files as is shown below:

:4416 -> MY.NET.30.404/20-13:42:18.152264 [**] spp_portscan: portscan status from 213.180.193.68: 5 connections across 1 hosts: TCP(5), UDP(0) [**]

This corruption was likely caused by the sanitizing process, however this should be verified. It is possible this was the result of someone removing portions of the logs to cover their tracks.

Detect 1 - NETBIOS NT NULL session

Description

This attack involves using a feature in Microsoft Windows referred to as Interprocess Control. This feature allows connections, to Microsoft Windows machines, that query for a list of available resources. By default Microsoft Windows systems allow this to occur with a blank user name and password. The service that will listen for the request is the SMB (also referred to as the Server Service) . The attacker connects to a special built in share called IPC$. Once connected the attacker can get a list of users, machines, shares and much more (maybe even a set of Ginsu Knives). For a good example of the type of information available try out Jordan Ritter's enum program.

http://www.bindview.com/Support/RAZOR/Utilities/Windows/enum_readme.cfm

Looking at the logs we can see evidence of attempts to exploit this feature. Combining both alert and scan logs shows the attackers intend to gather reconnaissance information. In order to conserve space I have truncated down the entries:

Apr 22 21:14:15 67.104.112.42:1937 -> 130.85.70.18:57 SYN ******S*
Apr 22 21:14:15 67.104.112.42:4433 -> 130.85.70.27:57 SYN ******S*

```
Apr 22 21:14:15 67.104.112.42:3064 -> 130.85.70.41:57 SYN ******S*
.
.
.
Apr 22 21:14:23 67.104.112.42:1685 -> 130.85.70.50:57 SYN ******S*
Apr 22 21:14:25 67.104.112.42:2331 -> 130.85.70.187:57 SYN ******S*
Apr 22 21:14:26 67.104.112.42:2243 -> 130.85.70.154:57 SYN ******S*
Apr 22 21:14:26 67.104.112.42:4335 -> 130.85.70.185:57 SYN ******S*
04/22-21:30:45.893536  [**] NETBIOS NT NULL session [**] 67.104.112.42:3940 -> MY.NET.190.95:139
```

Reason this detect was selected

Without the full packet capture it is difficult to tell what the attacker was able to ascertain if anything at all. However, if they had connected to the domain controller and the domain controller allowed null sessions then the attacker would have been able to gain very valuable information, starting with a list of all user accounts.

However more concerning is that the SMB request resulted in a connection being established. SMB is a difficult protocol to secure and a Google search for "SMB Vulnerabilities" resulted in about 162,000 results.

Also by combining the scan log and the alert log we can see that this attacker was scanning for many machines at a fast pace. Basically we know this attacker is certainly not up to anything good and has established a connection to the inside network to a service that is hard to secure.

Detect was generated by

The rule that tripped the alert is shown below:

alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:"NETBIOS NT NULL session"; **flow:to_server,established**; **content:"|00 00 00 00|W|00|i|00|n|00|d|00|o|00|w|00|s|00| |00|N|00|T|00| |00|1|00|3|00|8|00|1"**; reference:arachnids,204; reference:bugtraq,1163; reference:cve,2000-0347; classtype:attempted-recon; sid:530; rev:10;)

For a detailed description of this rule go to http://www.snort.org/snort-db/sid.html?sid=530 or see Appendix A)

The bolded parts of the above rule indicate that the rule will only trigger if there is an established session between the attacker and the server and that the content was seen going from the attacker to the server.

Probability the source address was spoofed

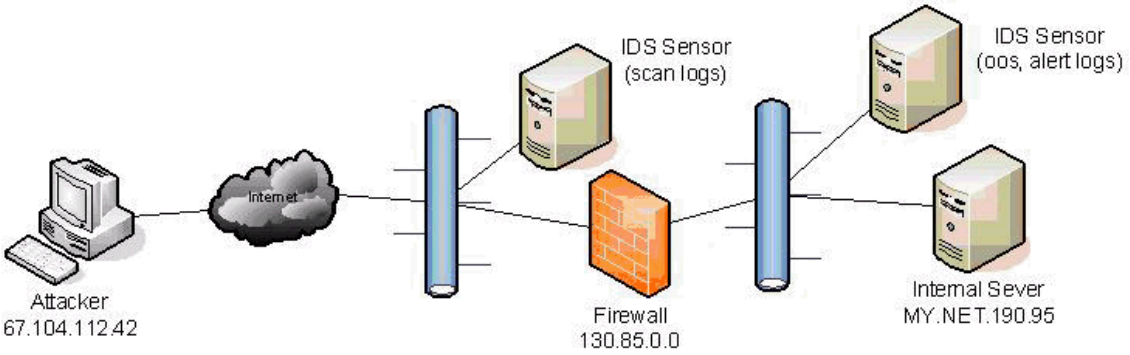
It is very unlikely that the source address was spoofed. In this attack the attacker would require both the port unreachable and the connection to the port 139 packets to be returned to them. Otherwise the attack would not have been successful.

Attack mechanism

It is difficult to determine what generated the Null Session as many tools even

the Microsoft Windows Command prompts are capable of this type of activity. However, the scan is likely to have originated from fx-tools, this scanner tool typically uses port 57, as this port usually has no service running. This indicates that the attacker is using inverse mapping. They are expecting to get back a port unreachable message which tells the attacker a live host is on the other end.

Link Diagram



Evidence of Active Targeting

The attacker appeared to randomly search IP address on port 57. This indicates that the attacker is in the early stages. The attacker did find a responding service and did connect to the service. No other activity has been detected coming from this attacker (within the three days reviewed).

Severity

3	**Criticality** Although it is not obvious what the targets systems function is. The SMB service on any inside machine is a great launching point for any attacker.
2	**Lethality** Connection to the SMB service and gaining a list of resources does not in itself mean a compromise is immanent. This is still at the reconnaissance phase.
1	**System countermeasures** As is typically the case with SMB there is little prevention on the client side. The assumption is that SMB would not likely be made available from the internet to the inside network.
1	**Network countermeasures** The evidence of the continued scans indicate that port unreachable message made it to the attacker and the fact that a connection existed on port 139 indicates the firewall did not block this activity.
3	**Severity** (Criticality + Lethality) – (System countermeasures + Network countermeasures)

Recommended Action

First recommendation would be to block any incoming traffic on ports 137-139 (SMB Ports), however being a “Open Resource” environment such as a University this may not be possible. If that is the case, all Windows systems should have the Null Session disabled. The link below explains the procedure:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/xpehelp/html/xeconreducenullsessionvulnerability.asp

Detect 2 - DDOS mstream handler to client

Description

After looking at the activity on MY.NET.84.235 it became obvious that something wasn't right. This system was (may still be) a client to a DDOS attacker for the time period which I evaluated the logs and looks to have participated in at least some scanning activity. Stepping through the log entries we can see quite the picture here:

First we see lots of DDOS shaft client to handler activity. Which, based on Snort.org description of the rule (see Appendix C or go to http://www.snort.org/snort-db/sid.html?sid=230) could be a false positive if MY.NET.84.235 is running a legitimate service on destination port 20432 or selects 20432 as an FTP data port (not likely the case given all the other alerts on this IP address):

```
04/20-14:45:09.171005  [**] DDOS shaft client to handler [**] 81.220.163.126:4662 -> MY.NET.84.235:20432
04/20-14:42:38.799893  [**] DDOS shaft client to handler [**] 81.220.163.126:4662 -> MY.NET.84.235:20432
.
.
.
04/20-14:43:20.615257  [**] DDOS shaft client to handler [**] 81.220.163.126:4662 -> MY.NET.84.235:20432
04/20-14:44:31.800532  [**] DDOS shaft client to handler [**] 81.220.163.126:4662 -> MY.NET.84.235:20432
```

Then we see this system perform a lot of scanning, I have included only a snippet of the log file for the sake of space:

```
04/20-20:28:19.267637  [**] spp_portscan: PORTSCAN DETECTED from MY.NET.84.235 (THRESHOLD 12 connections exceeded in 2 seconds) [**]
04/20-20:28:23.489296  [**] spp_portscan: portscan status from MY.NET.84.235: 13 connections across 13 hosts: TCP(12), UDP(1) [**]
04/20-20:28:26.954717  [**] spp_portscan: portscan status from MY.NET.84.235: 1 connections across 1 hosts: TCP(1), UDP(0) [**]
```

Then we see this alert log entry indicating possible Trojan server activity. Noting the port number of 27374 this shows possible infection of MY.NET.84.235 with the subseven Trojan. Possibly the tool used to setup the node in the DDOS network.

```
04/20-20:29:58.577139  [**] Possible trojan server activity [**] MY.NET.84.235:27374 -> 83.30.0.226:4662
```

Now we see MY.NET.84.235 talking as a handler to several different destinations. Snort.org has no false positives listed on this rule:

```
04/20-22:19:55.972148  [**] DDOS mstream handler to client [**] MY.NET.84.235:15104 -> 128.12.76.48:4662
04/20-22:20:01.320086  [**] DDOS mstream handler to client [**] MY.NET.84.235:15104 -> 128.12.76.48:4662
04/21-03:39:35.713651  [**] DDOS mstream handler to client [**] MY.NET.84.235:12754 -> 172.174.69.186:1164
04/21-03:39:46.241262  [**] DDOS mstream client to handler [**] 172.174.69.186:1164 -> MY.NET.84.235:12754
04/21-15:45:54.296147  [**] DDOS mstream client to handler [**] 63.166.3.20:80 -> MY.NET.84.235:12754
```

The lack of explanation for a false positive on the last set of alerts substantiates the concerns relating to the other log entries observed. It is reasonable to assume this machine has been compromised. Snort.org recommendation was to quarantine the machine and reload the system.

Reason this detect was selected

This machine is showing signs participating in questionable actions. The excessive scanning activity and the connections to DDOS clients and the possible sub seven Trojan activities all indicates that this system has been

compromised.

Detect was generated by

The rule that tripped the alert is shown below (Also see Appendix B):

alert tcp $HOME_NET 12754 -> $EXTERNAL_NET any (msg:"DDOS mstream handler to client";
flow:to_client,established; **content:">"**; reference:cve,2000-0138; classtype:attempted-dos; sid:248; rev:4;)

For a detailed description of this rule go to http://www.snort.org/snort-db/sid.html?sid=248 or see Appendix B)

The bolded parts show the rule will only trigger if a connection was established from the host towards the client. In this case the host was MY.NET.84.235 and the client was, among others, 128.12.76.48. Note the existence of a connect option as well to help reduce false positives. Snort.org reports no known false positive situation for this rule.

Probability the source address was spoofed

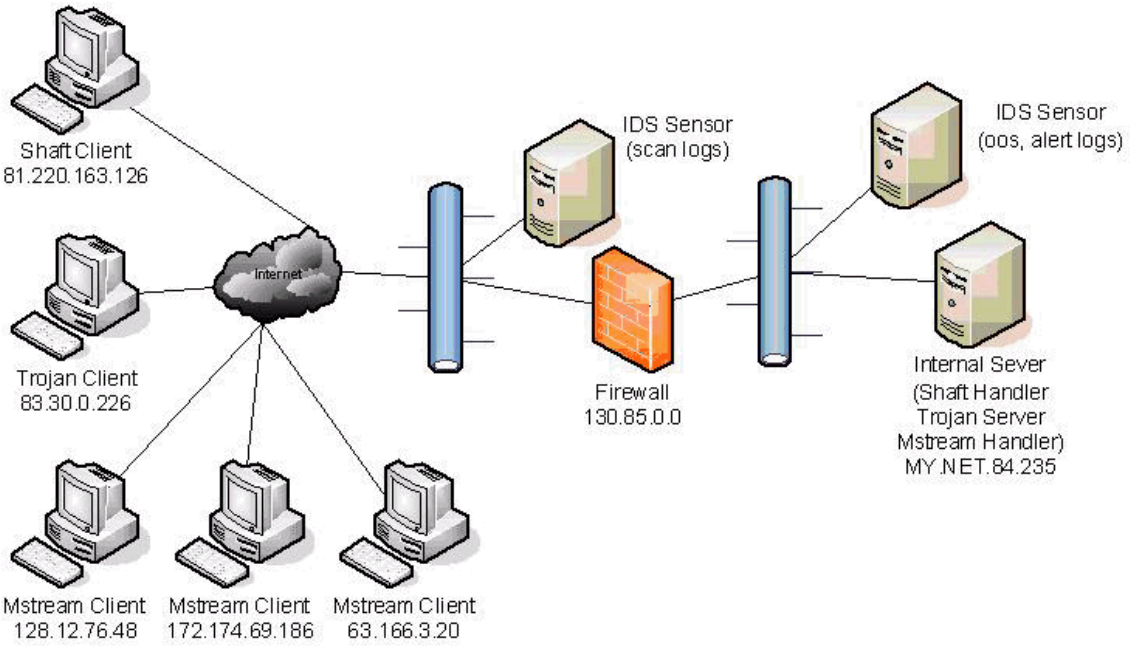
It is unlikely the address is spoofed as the DDOS client requires a response from the handlers in order to retrieve reconnaissance information.

Attack mechanism

Because the machine was compromised before the date of the logs, some assumptions will have to be made. The machine in question was either installed with or a service was added that was vulnerable to an attack. The attacker found this vulnerability and was able to use it. This usually occurs because security patches have not been applied in a timely fashion.

Once the attacker discovered this, they used it to install “Subseven”, a program that makes your machine a slave to the attackers commands. Using this program they sent the software and the commands to install it. This established your machine in their Distributed Denial of Service network. A Distributed Denial of Service network is a collection of machines that act as one force. Much like an army of soldiers, they obey commands from the attacker, commands that perform reconnaissance and attacks. Your machine is acting as both a soldier and leader in the army, both receiving commands and passing commands onto other soldiers.

Link Diagram



Correlations

Doug Kite, in his paper on "Intrusion Detection in Depth GCIA Practical Assignment, v3.3, SANS Virginia Beach, July 2002" (http://www.giac.org/certified_professionals/practicals/gcia/0609.php) observed similar activity in his investigation. Although the attack that he is describing differs you can see by his log trace that the machine he is investigating has similar alert entries:

"Since a trojaned machine would be listening on port 27374 and responding with
that as its source port, the most interesting entry above is the one from MY.NET.140.47, especially considering that the destination port (7777) is another known trojan port. This machine does appear to have been trojaned and should be quarantined. This was confirmed by searching alerts for other entries involving MY.NET.140.47, which revealed many portscans originating from this host, and the following:"

12/20-22:26:01.667209 [**] **DDOS** shaft client to handler [**] 64.230.128.142:3456 -> MY.NET.140.47:20432
12/20-22:26:06.560561 [**] **DDOS** shaft client to handler [**] 64.230.128.142:3456 -> MY.NET.140.47:20432

This indicates this problem has existed for sometime, a review of network countermeasures would be in order.

Evidence of Active Targeting

Of the IP addresses listed above there was no other activity found for the three days evaluated. This indicates that the DDOS node was already established and all participants where aware of its existence.

Severity

3	**Criticality** No other activity was noted so it is difficult to tell what function this system is performing. Therefore I gave this a moderate score.
5	**Lethality** The DDOS shaft activity and likely Subseven activities indicate that this system has a known vulnerability that is likely not patched. This machine is also in the control of the attacker.
1	**System countermeasures** As there is established connection from the outside to this machine and it appears to be under the control of outside influences I would say any system countermeasures have failed.
2	**Network countermeasures** It should be noted that the network sensor did pick up the attack, however there was no egress blocking on known bad ports. Of particular interest is the subseven port, 27343, being permitted from the inside towards the outside. However, the activity only appears in one of the three days I evaluated, indicating that the incident could have been responded to.
5	**Severity** (Criticality + Lethality) – (System countermeasures + Network countermeasures)

Recommended Action

Several steps should be taken here:

1. Start by enabling firewall rules to contain the situation.
2. Evaluate the system in question to determine if the system will require rebuilding.
3. If this is a student resource, there should be a review of the Policy permitting the students access. This policy should clearly state that they must have all security patches installed and be running anti-virus software with a current signature database.
4. If this is not a student resource it should be rebuild, anti-virus software installed, a policy created that identifies how and when signature updates are to occur. Also a policy created that identifies how and when security patches should be installed.

Detect 3 – Null scan! (Stealth) of POP3 Mail Server

Description

While reviewing the oos (Out of Specification) file I noticed IP address 68.121.194.43 with some suspicious traffic directed to 130.85.12.4. Of the given traffic Null Scan, packets that contain no TCP flag set at all, make up most of the traffic. I have pieced together portions of the oos, alert and scan logs to show the related activity.

Let's start with some of the snippets from the oos log, here we can see the elapsed time between the first entry and the second entry is approximately 40 minutes apart. Also note the duplication of the IP ID 4660. Both characteristics are consistent with oos activity from 68.121.194.43. Finally note the ******** flag setting, this is not in accordance with RFC and is referred to as a Null Scan:

```
04/20-00:21:36.980980 68.121.194.43:27401 -> MY.NET.12.4:110
TCP TTL:78 TOS:0x0 ID:4660 IpLen:20 DgmLen:40
******** Seq: 0xB24F001  Ack: 0x38122EE4  Win: 0x800  TcpLen: 20

04/20-01:05:23.770837 68.121.194.43:27913 -> MY.NET.12.4:110
TCP TTL:78 TOS:0x0 ID:4660 IpLen:20 DgmLen:40
******** Seq: 0xB462001  Ack: 0xDCDA562C  Win: 0x800  TcpLen: 20
```

Simply put a Null Scan comes from a packet that has all of the TCP Flags turned off. When this packet is received by the intended target they will respond one of two ways. If they have no listening service on the given port then the target will respond using a packet with the Reset Flag set. If the target has a listening service then it will not respond at all. This is referred to as inverse scanning. Even though there is a predictable response to Null Scan packets these packets do not conform to RFC and should never be seen in regular network traffic.

Next let's review some output from a summary report. It shows very clearly the correlation of the events from the oos, alert and scans files. Note the long delay between sets of activity indicating that the attacker is attempting to stay low and slow, trying not to be noticed:

Source IP	Src Port	Date	Time	Event	Dest. IP	Dst Port	log
68.121.194.43	37385	04/20	14:37:50	Null scan!	MY.NET.12.4	110	alert
68.121.194.43	37385	04/20	14:37:50	NULL ********	130.85.12.4	110	scans
68.121.194.43	37385	04/20	14:38:12	UNKNOWN *2*A**** RESERVEDBITS	130.85.12.4	110	scans
68.121.194.43	37641	04/20	15:00:59	Null scan!	MY.NET.12.4	110	alert
68.121.194.43	37641	04/20	15:00:59	NULL ********	130.85.12.4	110	scans
68.121.194.43	37897	04/20	15:22:53	Null scan!	MY.NET.12.4	110	alert
68.121.194.43	37897	04/20	15:22:53	NULL ********	130.85.12.4	110	scans
68.121.194.43	39945	04/20	18:19:00	Null scan!	MY.NET.12.4	110	alert
68.121.194.43	39945	04/20	18:19:00	NULL ********	130.85.12.4	110	scans
68.121.194.43	40457	04/20	19:02:48	Null scan!	MY.NET.12.4	110	alert
68.121.194.43	40457	04/20	19:02:48	NULL ********	130.85.12.4	110	scans
68.121.194.43	40713	04/20	19:24:41	Null scan!	MY.NET.12.4	110	alert
68.121.194.43	40713	04/20	19:24:41	NULL ********	130.85.12.4	110	scans
68.121.194.43	40969	04/20	19:46:35	Null scan!	MY.NET.12.4	110	alert

68.121.194.43	40969	04/20	19:46:35	NULL ********	130.85.12.4	110	scans
68.121.194.43	41481	04/20	20:30:22	SYN ******S*	130.85.12.4	110	scans
68.121.194.43	41481	04/20	20:30:23	Null scan!	MY.NET.12.4	110	alert
68.121.194.43	41481	04/20	20:30:23	NULL ********	130.85.12.4	110	scans
68.121.194.43	41737	04/20	20:53:07	Null scan!	MY.NET.12.4	110	alert
68.121.194.43	41737	04/20	20:53:07	SYN ******S*	130.85.12.4	110	scans
68.121.194.43	41737	04/20	20:53:07	NULL ********	130.85.12.4	110	scans
68.121.194.43	41993	04/20	21:15:01	Null scan!	MY.NET.12.4	110	alert
68.121.194.43	41993	04/20	21:15:01	SYN ******S*	130.85.12.4	110	scans
68.121.194.43	41993	04/20	21:15:01	NULL ********	130.85.12.4	110	scans
68.121.194.43	42249	04/20	21:36:55	Null scan!	MY.NET.12.4	110	alert
68.121.194.43	42249	04/20	21:36:55	NULL ********	130.85.12.4	110	scans
68.121.194.43	42505	04/20	21:58:48	SYN ******S*	130.85.12.4	110	scans
68.121.194.43	42505	04/20	21:58:49	Null scan!	MY.NET.12.4	110	alert
68.121.194.43	42505	04/20	21:58:49	NULL ********	130.85.12.4	110	scans
68.121.194.43	42761	04/20	22:20:43	Null scan!	MY.NET.12.4	110	alert
68.121.194.43	42761	04/20	22:20:43	NULL ********	130.85.12.4	110	scans
68.121.194.43	44041	04/21	00:10:52	Null scan!	MY.NET.12.4	110	alert
68.121.194.43	44041	04/21	00:10:52	NULL ********	130.85.12.4	110	scans
68.121.194.43	44041	04/21	00:10:52	oos	MY.NET.12.4	110	oos
68.121.194.43	44553	04/21	00:54:45	Null scan!	MY.NET.12.4	110	alert
68.121.194.43	44553	04/21	00:54:45	SYN ******S*	130.85.12.4	110	scans
68.121.194.43	44553	04/21	00:54:45	NULL ********	130.85.12.4	110	scans
68.121.194.43	44553	04/21	00:54:45	oos	MY.NET.12.4	110	oos
68.121.194.43	44809	04/21	01:16:38	Null scan!	MY.NET.12.4	110	alert
68.121.194.43	44809	04/21	01:16:38	NULL ********	130.85.12.4	110	scans
68.121.194.43	44809	04/21	01:16:38	oos	MY.NET.12.4	110	oos
68.121.194.43	45065	04/21	01:38:32	Null scan!	MY.NET.12.4	110	alert
68.121.194.43	45065	04/21	01:38:32	SYN ******S*	130.85.12.4	110	scans
68.121.194.43	45065	04/21	01:38:32	NULL ********	130.85.12.4	110	scans
68.121.194.43	45065	04/21	01:38:32	oos	MY.NET.12.4	110	oos
68.121.194.43	45321	04/21	02:00:26	Null scan!	MY.NET.12.4	110	alert
68.121.194.43	45321	04/21	02:00:26	NULL ********	130.85.12.4	110	scans
68.121.194.43	45321	04/21	02:00:26	oos	MY.NET.12.4	110	oos
68.121.194.43	45577	04/21	02:22:20	Null scan!	MY.NET.12.4	110	alert
68.121.194.43	45577	04/21	02:22:20	NULL ********	130.85.12.4	110	scans
68.121.194.43	45577	04/21	02:22:20	oos	MY.NET.12.4	110	oos
68.121.194.43	45833	04/21	02:44:13	oos	MY.NET.12.4	110	oos
68.121.194.43	46089	04/21	03:06:07	oos	MY.NET.12.4	110	oos

Next let's review the Apr 21 00:10:52 events. This is what I call "stumbling home at 2:00 AM events" where you knock over everything in sight. The attacker has tripped an entry in all three logs. Looking at the entries below, once again we see evidence of a Null scan. Also it becomes obvious that 130.85.12.4 and MY.NET.12.4 are IP addresses to the same machine. One interesting note is that looking at the summary report above not all scan attempts generated an entry in the oos file (as is noted below):

04/21-00:10:52.460379 [**] Null scan! [**] 68.121.194.43:44041 -> MY.NET.12.4:110

Apr 21 00:10:52 68.121.194.43:44041 -> 130.85.12.4:110 NULL ********

04/21-00:10:52.460370 68.121.194.43:44041 -> MY.NET.12.4:110
TCP TTL:78 TOS:0x0 ID:4660 IpLen:20 DgmLen:40
******** Seq: 0x8C4E001 Ack: 0x4C8CFF27 Win: 0x800 TcpLen: 20

There are a few likely reasons, the sensor generating the oos file was not listening at the time, or the sensor was overrun with packets and simply dropped them. Another explanation is the attacker may have found a way to evade the IDS sensor. Performing a network capture would be the best approach to answering this question.

It should be noted that this attack could simply be a miss-configured client sending bad packets. However there is enough compelling evidence to warrant a further investigation.

Reason this detect was selected

I struggled with selecting this detect as there is no clear sign of a compromise and no one sign of an attack. There is however a lot of circumstantial evidence that either the machine is compromised or that one is imminent. First we have a 1 to 1 attack meaning that 68.121.194.43 is working on 130.85.12.4 and only that machine. The attacker is keeping their activity low and slow meaning they are being patient. They could be trying to brute force a password, or might be trying different vulnerabilities, in hopes of finding one that works.

It is reasonable to assume that the attacker knows what pop server is being used. Although the banner is somewhat generic "+OK POP3 mr5.umbc.edu v2003.83 server ready" the error message resulting from the lack of connection is not. The message "-ERR Autologout; idle for too long" easily pointed to this application:

```
/*
 * Program:    IPOP3D - IMAP to POP3 conversion server
 *
 * Author:     Mark Crispin
 *             Networks and Distributed Computing
 *             Computing & Communications
 *             University of Washington
 *             Administration Building, AG-44
 *             Seattle, WA  98195
 *             Internet: MRC@CAC.Washington.EDU
 *
 * Date:       1 November 1990
 * Last Edited:       17 January 2003
 *
 * The IMAP toolkit provided in this Distribution is
 * Copyright 1988-2003 University of Washington.
 * The full text of our legal notices is contained in the file called
 * CPYRIGHT, included with this Distribution.
 */
```

Full source listing can be found in Appendix D. The source list matched on both the Banner and the Error message. Therefore it can be assumed that the attacker has this code listing and is familiar with this service.

The final reason for selecting this detect can be found in the correlation section. Jose Faial also recognized this detect back in April 2004, (http://www.giac.org/certified_professionals/practicals/gcia/0730.php) therefore this activity has been going on for almost a year now. This discovery indicates that either someone has a grossly miss-configured system (that is still working) or that this attacker has a continued reason for sending malformed packets.

All of this evidence brings to light the need to perform more investigation.

Detect was generated by

Although I was unable to locate the specific rule that generated the alert we can make some assumptions based on what we see in the oos entry. The entries in the oos file, with no flags set, show no restriction on ports along with no flags set:

```
04/21-00:10:52.460370 68.121.194.43:44041 -> MY.NET.12.4:110
TCP TTL:78 TOS:0x0 ID:4660 IpLen:20 DgmLen:40
******** Seq: 0x8C4E001  Ack: 0x4C8CFF27  Win: 0x800  TcpLen: 20
```

```
alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"Null Scan!"; flags 0; …)
```

Probability the source address was spoofed

This is difficult to determine, however one assumption is that the attacker is not simply scanning the machine. They already know port 110 on that machine is open. It is reasonable to assume that they are trying to push packets onto the machine either because of a miss-configured client or for malicious purposes. If it is a miss-configured client then the assumption would be, the address is not spoofed. If the intent is malicious then this could in fact be a spoofed address. Once again this activity begs for a full packet capture with further analysis. The analyst would want to look for established sessions and closely look at any traffic returned to 68.121.194.43.

Attack mechanism

Although this attack could simply be a miss-configured client, there is compelling evidence to suggest packet crafting is happening. The most compelling piece of evidence is the consistent IP ID of 4660. It is unlikely that a miss-configured client would respond in such a consistent manner. To prove this point, below is an example of how hping2 would be used to create such a packet.

Hping2 was able to recreate the packet information found in the oos file. The command used is listed below:

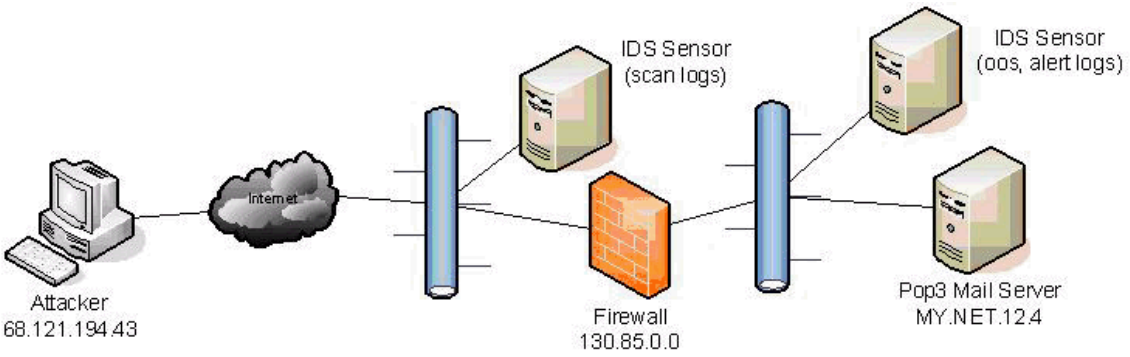
```
hping2 -t 78 -N 4660 -p 110 127.0.0.1
```

The result of this command, as shown by tcpdump, is listed below:

```
[root@localhost downloads]# tcpdump -xvn -i lo
tcpdump: listening on lo, link-type EN10MB (Ethernet), capture size 96 bytes
22:50:33.991361 IP (tos 0x0, ttl  78, id 4660, offset 0, flags [none], proto 6, length:
40) 127.0.0.1.1979 > 127.0.0.1.pop3: . [tcp sum ok] win 2048
        0x0000:  4500 0028 1234 0000 4e06 5c9a 7f00 0001  E..(.4..N.\.....
        0x0010:  7f00 0001 07bb 006e 48b8 5b89 66a5 25bd  .......nH.[.f.%.
        0x0020:  5000 0800 7115 0000                      P...q...
```

This does not prove hping2 created the packet but it does demonstrate that packet crafting tools are capable of doing so.

Link Diagram



Correlations

"Jose Faial observed back in April 2004:

```
04/11-00:19:17.322773 68.121.194.43:6663 -> 130.85.12.4:110 TCP TTL:78 TOS:0x0
ID:4660 IpLen:20 DgmLen:40 ******** Seq: 0xF7E6001 Ack: 0x1D10773 Win: 0x800
TcpLen: 20
04/11-00:41:14.795330 68.121.194.43:6919 -> 130.85.12.4:110 TCP TTL:78 TOS:0x0
ID:4660 IpLen:20 DgmLen:40 ******** Seq: 0xFA88001 Ack: 0x547783B5 Win: 0x800
TcpLen: 20
04/11-00:19:17.322776 [**] Null scan! [**] 68.121.194.43:6663 -> 130.85.12.4:110
04/11-00:41:14.795334 [**] Null scan! [**] 68.121.194.43:6919 -> 130.85.12.4:110
```

They correlate exactly. Most packets come from 68.121.194.43 and are directed to POP3 port of machine 130.85.12.4. Note that IP ID is always the same 4660. This is obviously a result of packet crafting or fragmentation, which would explain the lack of TCP flags as well, but there is no information that indicates these packets are fragments. I suggest take a close look at 130.85.12.4 (mail.umbc.edu), someone are certainly trying to break-in. A brute-force against a POP3 user account is my best guess for now. I will consider the remaining combinations as noise caused by packet corruption, because of its low number of hits."

Evidence of Active Targeting

The fact that there is no other activity for this IP address implies that advance reconnaissance has already been done. It also shows that the attacker knew this server was a pop3 server.

Severity

4	**Criticality** This is a pop3 mail server. A compromise of this server could result in mail being retrieved from the server by the attacker. This could result in corporate information leaking to competitors or privacy violations.
1	**Lethality** There is no clear evidence that an attacker has an attack that will compromise the machine. At most we see reconnaissance activity.
3	**System countermeasures** Without packet captures it is difficult to determine if a connection was made to the system. It is also difficult to say if the system responded to the inverse scan. Because there is a lack of evidence I will give it a moderate score.
1	**Network countermeasures** The firewall clearly allowed OOS packets to traverse the firewall. An appropriate response would be to silently drop all none RFC packets. This way inverse scans would not be successful.
1	**Severity** (Criticality + Lethality) – (System countermeasures + Network countermeasures)

Recommended Action

There is no clear sign of immediate danger and taking a mail server out of production, when students are so dependant on this resource, would be a mistake. A network capture device should be installed and this traffic should be captured and analyzed. If a compromise has occurred it would be best to contain the problem via firewall rules and asses how to rebuild the server during a regularly scheduled outage. Also a review should proceed that determines why the Null Scan packets are traversing the firewall.

Network Statistics

Top five talkers

When looking at the top five talkers, it's best to look at groups of IP addresses also known as subnets. When reviewing the logs there appeared to be three discernable groups of IP addresses, MY.NET which represented the internal network (behind the firewall), 130.85 which represented the addresses served externally by the firewall and all the other addresses which represent traffic coming from the Internet. It should be noted that IP's can be spoofed so there is no guarantee this approach will be 100% accurate, however the amount of spoofed traffic would be small (statistically speaking) and would have little bearing on the statistics shown. Using this approach will assist in any further investigation.

All of these reports where generated from a combined tally of events from all three files oos, alert and scans:

Top Five Talkers - Internal addresses Report (MY.NET.)

This report shows the top 5 talkers from the inside network. These should be investigated for possible compromises. If an internal machine has been compromised it will often become chatty. Especially if it's involved in a DDOS network:

SourceIP	Events
MY.NET.1.3	103902
MY.NET.1.4	83424
MY.NET.81.39	70498
MY.NET.112.189	23876
MY.NET.17.45	12710

Top Five Talkers – External Firewall Address Report (130.85.)

This report shows the top five talkers from the addresses served externally by the firewall. This traffic should be investigated to determine if an inside machine is compromised but also to review the effectiveness of the firewall, firewall rules, IDS and IDS rules. Perhaps your exterior firewall devices are permitting traffic they should not or the IDS rules are too noisy and require some tuning.

SourceIP	Events
130.85.1.3	2536679
130.85.17.45	1179172
130.85.1.4	747942
130.85.112.189	713579
130.85.81.39	694877

Top Five Talkers – Internet Address Report (All Other Addresses)
This report shows the top five talkers coming from the Internet. This list should be investigated to determine if these addresses are originating from people intending to attack the network.

SourceIP	Events
213.180.193.68	39516
220.197.192.39	31282
64.136.199.197	23712
134.192.42.11	21786
80.191.163.12	19244

Top targeted ports

When looking at the top five destination ports, it is equally effective to report based on the IP groups mentioned above. Again this will help determine a strategy for further investigation.

All of these reports where generated from a combined tally of events from all three files oos, alert and scans:

Top five targeted ports – Originating Internally (MY.NET) Report

This report shows the top five destination ports the internal machines are sending to. The traffic represented here should be investigated for possible Trojan activity. Of particular interest is port 65535 – Possible RC1-Trojan activity, port 27374 – Possible Exploit Translation Server, Kazimas, Remote Grab, SubSeven 2.1 Gold. Port 137 is likely miss-configured Windows traffic and port 25 is likely just email activity. However traffic on all of these ports should be investigated further.

DestinationPort	Events
65535	9736
137	5977
25	440
27374	201
7000	58

Top five targeted ports – Originating from Firewall (130.85) Report

This report shows the top five destination ports of traffic originating from the firewall's external addresses. This traffic should be investigated to determine if egress firewall rules are required. Again this traffic should be looked at as possible Trojan activity. Of particular interest is port 2745 – possible bagel activity. Port 135 is Windows DCOM service, which could be miss-configured clients or possible outgoing DCOM attacks, either way worth checking. Port 445 is likely just smb/tcp again likely miss-configured clients. The last two ports are quite expected with port 53 – DNS, port 80 – http.

DestinationPort	Events
53	3272789
135	1838879
2745	485985
80	468305
445	348356

Top five targeted ports - All Other Addresses

This report shows the top destination ports originating from the Internet. This activity should be monitored to understand what ports are commonly getting attacked. Of particular interest is 6129 used by Dameware a remote administration software package, having a history of being installed by viruses. Port 4899 associated with radmin another remote administration package. Port 20168 associated with LoveGate. The remaining port 443 and 80 are likely scanning activity from attackers looking for live hosts.

DestinationPort	Events
443	99525
6129	91049
80	72177
4899	55740
20168	41022

Profile of three most suspicious external Addresses

67.104.112.42

This address was selected due to it's involvement in detect 1.

```
OrgName:    XO Communications
OrgID:      XOXO
Address:    Corporate Headquarters
Address:    11111 Sunset Hills Road
City:       Reston
StateProv:  VA
PostalCode: 20190-5339
Country:    US

ReferralServer: rwhois://rwhois.eng.xo.com:4321/

NetRange:   67.104.0.0 - 67.111.255.255
CIDR:       67.104.0.0/13
NetName:    XOXO-BLK-17
NetHandle:  NET-67-104-0-0-1
Parent:     NET-67-0-0-0-0
NetType:    Direct Allocation
NameServer: NAMESERVER1.CONCENTRIC.NET
NameServer: NAMESERVER2.CONCENTRIC.NET
NameServer: NAMESERVER3.CONCENTRIC.NET
NameServer: NAMESERVER.CONCENTRIC.NET
Comment:
RegDate:
Updated:    2004-05-07

OrgAbuseHandle: XCNV-ARIN
OrgAbuseName:   XO Communications, Network Violations
OrgAbusePhone:  +1-866-285-6208
OrgAbuseEmail:  abuse@xo.com

OrgTechHandle: XCIA-ARIN
OrgTechName:   XO Communications, IP Administrator
OrgTechPhone:  +1-703-547-2000
OrgTechEmail:  ipadmin@eng.xo.com

# ARIN WHOIS database, last updated 2005-03-06 19:10
# Enter ? for additional hints on searching ARIN's WHOIS database.
```

213.180.193.68

This address was selected because it showed up on the top five talkers originating from the internet.

```
OrgName:    RIPE Network Coordination Centre
OrgID:      RIPE
Address:    P.O. Box 10096
City:       Amsterdam
StateProv:
PostalCode: 1001EB
Country:    NL

ReferralServer: whois://whois.ripe.net:43

NetRange:   213.0.0.0 - 213.255.255.255
CIDR:       213.0.0.0/8
NetName:    RIPE-213
NetHandle:  NET-213-0-0-0-1
Parent:
NetType:    Allocated to RIPE NCC
NameServer: NS-PRI.RIPE.NET
NameServer: NS3.NIC.FR
NameServer: SUNIC.SUNET.SE
NameServer: AUTH00.NS.UU.NET
NameServer: SEC1.APNIC.NET
NameServer: SEC3.APNIC.NET
NameServer: TINNIE.ARIN.NET
Comment:    These addresses have been further assigned to users in
Comment:    the RIPE NCC region. Contact information can be found in
Comment:    the RIPE database at http://www.ripe.net/whois
RegDate:
Updated:    2004-03-16

# ARIN WHOIS database, last updated 2005-03-06 19:10
# Enter ? for additional hints on searching ARIN's WHOIS database.
```

220.197.192.39

This address was selected because it showed up on the top five talkers originating from the internet.

```
OrgName:    Asia Pacific Network Information Centre
OrgID:      APNIC
Address:    PO Box 2131
City:       Milton
StateProv:  QLD
PostalCode: 4064
Country:    AU

ReferralServer: whois://whois.apnic.net

NetRange:   220.0.0.0 - 220.255.255.255
CIDR:       220.0.0.0/8
NetName:    APNIC6
NetHandle:  NET-220-0-0-0-1
Parent:
NetType:    Allocated to APNIC
NameServer: NS1.APNIC.NET
NameServer: NS3.APNIC.NET
NameServer: NS4.APNIC.NET
NameServer: NS.RIPE.NET
NameServer: TINNIE.ARIN.NET
Comment:    This IP address range is not registered in the ARIN
database.
Comment:    For details, refer to the APNIC Whois Database via
Comment:    WHOIS.APNIC.NET or http://www.apnic.net/apnic-
bin/whois2.pl
Comment:    ** IMPORTANT NOTE: APNIC is the Regional Internet
Registry
Comment:    for the Asia Pacific region. APNIC does not operate
networks
Comment:    using this IP address range and is not able to
investigate
Comment:    spam or abuse reports relating to these addresses. For
more
Comment:    help, refer to http://www.apnic.net/info/faq/abuse
Comment:
RegDate:
Updated:    2004-03-30

OrgTechHandle: AWC12-ARIN
OrgTechName:   APNIC Whois Contact
OrgTechPhone:  +61 7 3858 3100
OrgTechEmail:  search-apnic-not-arin@apnic.net

# ARIN WHOIS database, last updated 2005-03-06 19:10
# Enter ? for additional hints on searching ARIN's WHOIS database.
```

Analysis Process

This analysis was performed using the following hardware:

Hardware

Intel Celeron 400 Mhz
160 Meg of Ram
30 Meg Disk Space

Software

Fedora Core 2
Mysql 4.1.9.standard
Perl

Approach

Step	Description	Appendix
1	Created a database structure capable of loading the information.	E
2	Download the files from the web site	
3	Created PERL scripts to parse out the information into discernable fields	F
4	Uploaded the parsed information into a mysql database (1 table for each file)	G
5	Run a script that would summaries all the events from each table and placed the results into an events table	H

Advantages

1. The parsing of the data made me more conscious of the information contained in the logs.
2. The event table meant that as soon as I found a suspicious address I could immediately correlate all activity associated with the IP address.
3. I was able to quickly change my analysis approach by simply creating a new query on the fly. I found that most script approaches required you to rewrite the script.

Disadvantages

1. It was time consuming loading the information into the database. (This

was mitigated with the creation of a batch file that I could start and walk away).

2. The large number of records meant careful thought was required regarding indices, too many and the load took to long, not enough and the queries took too long.
3. Required learning mysql, I am a MCDBA so the transition was not too tough, but if you did not have previous DBA experience this could be difficult.
4. In looking for the “Coming home at 2:00 AM” IP’s (Tripped into all logs), I had to create another table to summarize results. This was because the properly formed query simply took forever to run (I finally cancelled it after hours of running).

List of References

Ritter, Jordan. "enum" Bind View. 8 Mar. 2005.
< http://www.bindview.com/Support/RAZOR/Utilities/Windows/enum_readme.cfm >.

"SID 530 - NETBIOS NT NULL session." Snort Signature Database. 8 Mar 2005
< http://www.snort.org/snort-db/sid.html?sid=530 >.

“Null Session Vulnerability.” MSDN. Sept 21, 2005, 8 Mar 2005
< http://msdn.microsoft.com/library/default.asp?url=/library/en-us/xpehelp/html/xeconreducenullsessionvulnerability.asp >

"SID 248 - DDOS mstream handler to client." Snort Signature Database. 8 Mar 2005
< http://www.snort.org/snort-db/sid.html?sid=248 >.

"SID 230 - DDOS shaft client login to handler." Snort Signature Database. 8 Mar 2005
< http://www.snort.org/snort-db/sid.html?sid=230 >.

Kite, Doug. “Intrusion Detection in Depth.” SANS Reading Room. July 2002, 8 March 2005
< http://www.giac.org/certified_professionals/practicals/gcia/0609.php>.

Faial, Jose. “GIAC Certified Intrusion Analyst – GCIA Practical Assignment.” SANS Reading Room. April 2002, 8 March 2005
< http://www.giac.org/certified_professionals/practicals/gcia/0609.php>.

Northcutt, Stephen. Snort 2.1 Intrusion Detection. Syngress, 2004.

Appendix

Appendix A (Snort's SID 530- NETBIOS NT NULL session)

GEN:SID	1:530
Message	NETBIOS NT NULL session
Rule	alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:"NETBIOS NT NULL session"; flow:to_server,established; content:"|00 00 00 00|W|00|i|00|n|00|d|00|o|00|w|00|s|00| |00|N|00|T|00| |00|1|00|3|00|8|00|1"; reference:arachnids,204; reference:bugtraq,1163; reference:cve,2000-0347; classtype:attempted-recon; sid:530; rev:10;)
Summary	This event is generated when an attacker sends a blank username and blank password in an attempt to connect to the IPC$ (Interprocess Communication) pipe.
Impact	Information gathering. This attack can permit the disclosure of sensitive information about the target host.
Detailed Information	Null sessions allow browsing of Windows hosts by the "Network Neighborhood" and other functions. A Null session permits access to a host using a blank user name and password. At attacker may attempt to perform a Null session connection, disclosing sensitive information about the target host such as available shares and user names.
Affected Systems	Microsoft Windows hosts
Attack Scenarios	An attacker can send a blank username and blank password to try to connect to the IPC$ hidden share on the target computer.
Ease of Attack	Simple.
False Positives	Null sessions may be used by legitimate processes in the same Windows domain. If you think this rule has a false positives, please help fill it out.
False Negatives	None Known If you think this rule has a false negatives, please help fill it out.
Corrective Action	On Windows NT, 2000, XP set the registry key /System/CurrentControlSet/Control/LSA/RestrictAnonymous value to 1.
Contributors	Original rule written by Ian Viket <ian.vitek@infosec.se> Documented by Nawapong Nakjang <tony@ksc.net, tonie@thai.com> Sourcefire Research Team Judy Novak <judy.novak@sourcefire.com>
Additional References	Arachnids http://www.whitehats.com/info/IDS204 CVE http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0519
Rule References	arachnids: 204 bugtraq: 1163 cve: 2000-0347

Appendix B (Snort's SID 248-DDOS mstream handler to client)

Snort Signature Database

GEN:SID

1:248

Message

DDOS mstream handler to client

Rule

alert tcp $HOME_NET 12754 -> $EXTERNAL_NET any (msg:"DDOS mstream handler to client"; flow:to_client,established; content:">"; reference:cve,2000-0138; classtype:attempted-dos; sid:248; rev:4;)

Summary

This event is generated when an mstream DDoS handler responds to an mstream client.

Impact

Severe. If the list source IP is in your network, it may be an mstream handler. If the listed destination IP is in your network, it may be an mstream client.

Detailed Information

The mstream DDoS uses a tiered structure of compromised hosts to coordinate and participate in a distributed denial of service attack. At the highest level, clients communicate with handlers to inform them to launch attacks. A client may communicate with a handler using a TCP packet to destination port 12754 with a string of ">" in the payload. A handler responds to this with a TCP source port of 12754 and a string of ">" in the payload.

Affected Systems

Any mstream compromised host.

Attack Scenarios

An mstream handler may be respond to a communication from an mstream client.

Ease of Attack

Simple. mstream code is freely available.

False Positives

None Known.
If you think this rule has a false positives, please help fill it out.

False Negatives

There are other known client-to-handler ports in addition to 12754.
If you think this rule has a false negatives, please help fill it out.

Corrective Action

Perform proper forensic analysis on the suspected compromised host to discover the means of compromise.

Rebuild a confirmed compromised host.

Use a packet-filtering firewall to block inappropriate traffic to the network to prevent hosts from being compromised.

Contributors

Original rule writer unknown
Sourcefire Research Team
Judy Novak <judy.novak@sourcefire.com>

Additional References

CVE:
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2000-0138

Rule References

cve: 2000-0138

Appendix C (Snort's SID 230-DDOS shaft client login to handler)

Snort Signature Database

GEN:SID

1:230

Message

DDOS shaft client login to handler

Rule

alert tcp $HOME_NET 20432 -> $EXTERNAL_NET any (msg:"DDOS shaft client login to handler"; flow:from_server,established; content:"login|3A|"; reference:arachnids,254; reference:url,security.royans.net/info/posts/bugtraq_ddos3.shtml; classtype:attempted-dos; sid:230; rev:5;)

Summary

This event is generated when a DDoS Shaft client communicates with a Shaft handler. It is also possible that this event may be generated when any host attempts to discover or detect a Shaft handler.

Impact

Attempted DDoS. If the listed source IP is in your network, it may be a Shaft client or a host attempting to discover Shaft handlers. If the listed destination IP is in your network, it may be a Shaft handler.

Detailed Information

The Shaft DDoS uses a tiered structure of compromised hosts to coordinate and participate in a distributed denial of service attack. At the highest level, clients communicate with handlers to direct them to launch attacks. A client may communicate with a handler via TCP destination port 20432.

Affected Systems

Any Shaft compromised host.

Attack Scenarios

A Shaft client needs to communicate with handlers to direct attacks.

Ease of Attack

Simple. Shaft code is freely available.

False Positives

A legitimate server port of 20432 will cause this rule to fire. It may also create a false positive if port 20432 is selected as an FTP data port.
If you think this rule has a false positives, please help fill it out.

False Negatives

None Known.
If you think this rule has a false negatives, please help fill it out.

Corrective Action

Perform proper forensic analysis on the suspected compromised host to discover the means of compromise.

Rebuild a confirmed compromised host.

Use a packet-filtering firewall to block inappropriate traffic to the network to prevent hosts from being compromised.

Appendix D (Source Listing for IPOP3D)

```
/*
 * Program:      IPOP3D - IMAP to POP3 conversion server
 *
 * Author:       Mark Crispin
 *               Networks and Distributed Computing
 *               Computing & Communications
 *               University of Washington
 *               Administration Building, AG-44
 *               Seattle, WA  98195
 *               Internet: MRC@CAC.Washington.EDU
 *
 * Date: 1 November 1990
 * Last Edited:  17 January 2003
 *
 * The IMAP toolkit provided in this Distribution is
 * Copyright 1988-2003 University of Washington.
 * The full text of our legal notices is contained in the file called
 * CPYRIGHT, included with this Distribution.
 */

/* Parameter files */

#include <stdio.h>
#include <ctype.h>
#include <errno.h>
extern int errno;         /* just in case */
#include <signal.h>
#include <time.h>
#include "c-client.h"


#define CRLF PSOUT ("\015\012")    /* primary output terpri */


/* Autologout timer */
#define KODTIMEOUT 60*5
#define LOGINTIMEOUT 60*3
#define TIMEOUT 60*10


/* Size of temporary buffers */
#define TMPLEN 1024


/* Server states */

#define AUTHORIZATION 0
#define TRANSACTION 1
#define UPDATE 2
#define LOGOUT 3

/* Eudora food */

#define STATUS "Status: %s%s\015\012"
#define SLEN (sizeof (STATUS)-3)


/* Global storage */
```

```
char *version = "2003.83";          /* server version */
short state = AUTHORIZATION;        /* server state */
short critical = NIL;               /* non-zero if in critical code */
MAILSTREAM *stream = NIL; /* mailbox stream */
long idletime = 0;                  /* time we went idle */
unsigned long nmsgs = 0;  /* current number of messages */
unsigned long ndele = 0;  /* number of deletes */
unsigned long last = 0;             /* highest message accessed */
unsigned long il = 0;               /* initial last message */
char challenge[128];                /* challenge */
char *host = NIL;          /* remote host name */
char *user = NIL;          /* user name */
char *pass = NIL;          /* password */
char *initial = NIL;                /* initial response */
long *msg = NIL;           /* message translation vector */
char *sayonara = "+OK Sayonara\015\012";


/* Function prototypes */

int main (int argc,char *argv[]);
void clkint ();
void kodint ();
void hupint ();
void trmint ();
int pass_login (char *t,int argc,char *argv[]);
char *apop_login (char *chal,char *user,char *md5,int argc,char *argv[]);
char *responder (void *challenge,unsigned long clen,unsigned long *rlen);
int mbxopen (char *mailbox);
long blat (char *text,long lines,unsigned long size);
void rset ();

/* Main program */

int main (int argc,char *argv[])
{
  unsigned long i,j,k;
  char *s,*t;
  char tmp[TMPLEN];
  time_t autologouttime;
  char *pgmname = (argc && argv[0]) ?
    (((s = strrchr (argv[0],'/')) || (s = strrchr (argv[0],'\\'))) ?
     s+1 : argv[0]) : "ipop3d";
                                    /* set service name before linkage */
  mail_parameters (NIL,SET_SERVICENAME,(void *) "pop");
#include "linkage.c"
                                    /* initialize server */
  server_init (pgmname,"pop3","pop3s",clkint,kodint,hupint,trmint);
  challenge[0] = '\0';              /* find the CRAM-MD5 authenticator */
  if (i = mail_lookup_auth_name ("CRAM-MD5",NIL)) {
    AUTHENTICATOR *a = mail_lookup_auth (i);
    if (a->server) {                /* have an MD5 enable file? */
                                    /* build challenge -- less than 128 chars */
      sprintf (challenge,"<%lx.%lx@%.64s>",(unsigned long) getpid (),
               (unsigned long) time (0),tcp_serverhost ());
    }
  }
  /* There are reports of POP3 clients which get upset if anything appears
   * between the "+OK" and the "POP3" in the greeting.
   */
  PSOUT ("+OK POP3");
  if (!challenge[0]) {              /* if no MD5 enable, output host name */
    PBOUT (' ');
    PSOUT (tcp_serverhost ());
```

```
}
PSOUT (" v");
PSOUT (version);
PSOUT (" server ready");
if (challenge[0]) {             /* if MD5 enable, output challenge here */
  PBOUT (' ');
  PSOUT (challenge);
}
CRLF;
PFLUSH ();                      /* dump output buffer */
autologouttime = time (0) + LOGINTIMEOUT;
                                /* command processing loop */
while ((state != UPDATE) && (state != LOGOUT)) {
  idletime = time (0);  /* get a command under timeout */
  alarm ((state == TRANSACTION) ? TIMEOUT : LOGINTIMEOUT);
  clearerr (stdin);             /* clear stdin errors */
  while (!PSIN (tmp,TMPLEN)){   /* read command line */
                                /* ignore if some interrupt */
    if (ferror (stdin) && (errno == EINTR)) clearerr (stdin);
    else {
      char *e = ferror (stdin) ?
        strerror (errno) : "Command stream end of file";
      alarm (0);                /* disable all interrupts */
      syslog (LOG_INFO,"%s while reading line user=%.80s host=%.80s",
              e,user ? user : "???",tcp_clienthost ());
      rset ();          /* try to gracefully close the stream */
      if (state == TRANSACTION) mail_close (stream);
      stream = NIL;
      state = LOGOUT;
      _exit (1);
    }
  }
  alarm (0);                    /* make sure timeout disabled */
  idletime = 0;         /* no longer idle */

  if (!strchr (tmp,'\012'))     /* find end of line */
    PSOUT ("-ERR Command line too long\015\012");
  else if (!(s = strtok (tmp," \015\012")))
    PSOUT ("-ERR Null command\015\012");
  else {                        /* dispatch based on command */
    ucase (s);          /* canonicalize case */
                                /* snarf argument */
    t = strtok (NIL,"\015\012");
                                /* QUIT command always valid */
    if (!strcmp (s,"QUIT")) state = UPDATE;
    else if (!strcmp (s,"CAPA")) {
      AUTHENTICATOR *auth;
      PSOUT ("+OK Capability list follows:\015\012");
      PSOUT ("TOP\015\012LOGIN-DELAY 180\015\012UIDL\015\012");
      if (s = ssl_start_tls (NIL)) fs_give ((void *) &s);
      else PSOUT ("STLS\015\012");
      if (mail_parameters (NIL,GET_DISABLEPLAINTEXT,NIL)) {
        PSOUT ("SASL");/* display secure server authenticators */
        for (auth = mail_lookup_auth (1); auth; auth = auth->next)
          if (auth->server) {
            if (auth->flags & AU_SECURE) {
              PBOUT (' ');
              PSOUT (auth->name);
            }
          }
      }
      else {                    /* display all authentication means */
        PSOUT ("USER\015\012SASL");
        for (auth = mail_lookup_auth (1); auth; auth = auth->next)
```

```
        if (auth->server) {
          PBOUT (' ');
          PSOUT (auth->name);
        }
    }
    CRLF;
    PSOUT (".\015\012");
}

else switch (state) {        /* else dispatch based on state */
case AUTHORIZATION: /* waiting to get logged in */
    if (!strcmp (s,"AUTH")) {
      if (t && *t) {  /* mechanism given? */
        if (host) fs_give ((void **) &host);
        if (user) fs_give ((void **) &user);
        if (pass) fs_give ((void **) &pass);
        s = strtok (t," ");   /* get mechanism name */
                              /* get initial response */
        initial = strtok (NIL,"\015\012");
        if (!(user = cpystr (mail_auth (s,responder,argc,argv)))) {
          PSOUT ("-ERR Bad authentication\015\012");
          syslog (LOG_INFO,"AUTHENTICATE %s failure host=%.80s",s,
                  tcp_clienthost ());
        }
        else if ((state = mbxopen ("INBOX")) == TRANSACTION)
          syslog (LOG_INFO,"Auth user=%.80s host=%.80s nmsgs=%ld/%ld",
                  user,tcp_clienthost (),nmsgs,stream->nmsgs);
        else syslog (LOG_INFO,"Auth user=%.80s host=%.80s no mailbox",
                     user,tcp_clienthost ());
      }
      else {
        AUTHENTICATOR *auth;
        PSOUT ("+OK Supported authentication mechanisms:\015\012");
        if (mail_parameters (NIL,GET_DISABLEPLAINTEXT,NIL)) {
          for (auth = mail_lookup_auth (1); auth; auth = auth->next)
            if (auth->server) {
              if (auth->flags & AU_SECURE) {
                PSOUT (auth->name);
                CRLF;
              }
            }
        }
                              /* display all authentication means */
        else for (auth = mail_lookup_auth (1); auth; auth = auth->next)
          if (auth->server) {
            PSOUT (auth->name);
            CRLF;
          }
        PBOUT ('.');
        CRLF;
      }
    }

    else if (!strcmp (s,"APOP")) {
      if (challenge[0]) {     /* can do it if have an MD5 challenge */
        if (host) fs_give ((void **) &host);
        if (user) fs_give ((void **) &user);
        if (pass) fs_give ((void **) &pass);
                              /* get user name */
        if (!(t && *t && (s = strtok (t," ")) && (t = strtok(NIL,"\012"))))
          PSOUT ("-ERR Missing APOP argument\015\012");
        else if (!(user = apop_login (challenge,s,t,argc,argv)))
          PSOUT ("-ERR Bad APOP\015\012");
        else if ((state = mbxopen ("INBOX")) == TRANSACTION)
```

```
          syslog (LOG_INFO,"APOP user=%.80s host=%.80s nmsgs=%ld/%ld",
                  user,tcp_clienthost (),nmsgs,stream->nmsgs);
        else syslog (LOG_INFO,"APOP user=%.80s host=%.80s no mailbox",
                     user,tcp_clienthost ());
      }
      else PSOUT ("-ERR Not supported\015\012");
    }
                              /* (chuckle) */
    else if (!strcmp (s,"RPOP"))
      PSOUT ("-ERR Nice try, bunkie\015\012");
    else if (!strcmp (s,"STLS")) {
      if (t = ssl_start_tls (pgmname)) {
        PSOUT ("-ERR STLS failed: ");
        PSOUT (t);
        CRLF;
      }
      else PSOUT ("+OK STLS completed\015\012");
    }
    else if (!mail_parameters (NIL,GET_DISABLEPLAINTEXT,NIL) &&
             !strcmp (s,"USER")) {
      if (host) fs_give ((void **) &host);
      if (user) fs_give ((void **) &user);
      if (pass) fs_give ((void **) &pass);
      if (t && *t) {  /* if user name given */
                              /* skip leading whitespace (bogus clients!) */
        while (*t == ' ') ++t;
                              /* remote user name? */
        if (s = strchr (t,':')) {
          *s++ = '\0';        /* tie off host name */
          host = cpystr (t);/* copy host name */
          user = cpystr (s);/* copy user name */
        }
                              /* local user name */
        else user = cpystr (t);
        PSOUT ("+OK User name accepted, password please\015\012");
      }
      else PSOUT ("-ERR Missing username argument\015\012");
    }
    else if (!mail_parameters (NIL,GET_DISABLEPLAINTEXT,NIL) &&
             user && *user && !strcmp (s,"PASS"))
      state = pass_login (t,argc,argv);
    else PSOUT ("-ERR Unknown AUTHORIZATION state command\015\012");
    break;

  case TRANSACTION:           /* logged in */
    if (!strcmp (s,"STAT")) {
      for (i = 1,j = 0,k = 0; i <= nmsgs; i++)
        if (msg[i] > 0) {     /* message still exists? */
          j++;                /* count one more undeleted message */
          k += mail_elt (stream,msg[i])->rfc822_size + SLEN;
        }
      sprintf (tmp,"+OK %lu %lu\015\012",j,k);
      PSOUT (tmp);
    }
    else if (!strcmp (s,"LIST")) {
      if (t && *t) {  /* argument do single message */
        if ((i = strtoul (t,NIL,10)) && (i <= nmsgs) && (msg[i] > 0)) {
          sprintf (tmp,"+OK %lu %lu\015\012",i,
                   mail_elt(stream,msg[i])->rfc822_size + SLEN);
          PSOUT (tmp);
        }
        else PSOUT ("-ERR No such message\015\012");
      }
      else {          /* entire mailbox */
```

```
      PSOUT ("+OK Mailbox scan listing follows\015\012");
      for (i = 1,j = 0,k = 0; i <= nmsgs; i++) if (msg[i] > 0) {
        sprintf (tmp,"%lu %lu\015\012",i,
                 mail_elt (stream,msg[i])->rfc822_size + SLEN);
        PSOUT (tmp);
      }
      PBOUT ('.'); /* end of list */
      CRLF;
    }
  }
  else if (!strcmp (s,"UIDL")) {
    if (t && *t) { /* argument do single message */
      if ((i = strtoul (t,NIL,10)) && (i <= nmsgs) && (msg[i] > 0)) {
        sprintf (tmp,"+OK %lu %08lx%08lx\015\012",i,stream->uid_validity,
                 mail_uid (stream,msg[i]));
        PSOUT (tmp);
      }
      else PSOUT ("-ERR No such message\015\012");
    }
    else {          /* entire mailbox */
      PSOUT ("+OK Unique-ID listing follows\015\012");
      for (i = 1,j = 0,k = 0; i <= nmsgs; i++) if (msg[i] > 0) {
        sprintf (tmp,"%lu %08lx%08lx\015\012",i,stream->uid_validity,
                 mail_uid (stream,msg[i]));
        PSOUT (tmp);
      }
      PBOUT ('.'); /* end of list */
      CRLF;
    }
  }

  else if (!strcmp (s,"RETR")) {
    if (t && *t) { /* must have an argument */
      if ((i = strtoul (t,NIL,10)) && (i <= nmsgs) && (msg[i] > 0)) {
        MESSAGECACHE *elt;
                        /* update highest message accessed */
        if (i > last) last = i;
        sprintf (tmp,"+OK %lu octets\015\012",
                 (elt = mail_elt (stream,msg[i]))->rfc822_size + SLEN);
        PSOUT (tmp);
                        /* output header */
        t = mail_fetch_header (stream,msg[i],NIL,NIL,&k,FT_PEEK);
        blat (t,-1,k);
                        /* output status */
        sprintf (tmp,STATUS,elt->seen ? "R" : " ",
                 elt->recent ? " " : "O");
        PSOUT (tmp);
        CRLF;           /* delimit header and text */
                        /* output text */
        t = mail_fetch_text (stream,msg[i],NIL,&k,NIL);
        blat (t,-1,k);
        CRLF;           /* end of list */
        PBOUT ('.');
        CRLF;
      }
      else PSOUT ("-ERR No such message\015\012");
    }
    else PSOUT ("-ERR Missing message number argument\015\012");
  }
  else if (!strcmp (s,"DELE")) {
    if (t && *t) { /* must have an argument */
      if ((i = strtoul (t,NIL,10)) && (i <= nmsgs) && (msg[i] > 0)) {
                        /* update highest message accessed */
        if (i > last) last = i;
```

```
                                /* delete message */
            sprintf (tmp,"%ld",msg[i]);
            mail_setflag (stream,tmp,"\\Deleted");
            msg[i] = -msg[i];   /* note that we deleted this message */
            PSOUT ("+OK Message deleted\015\012");
            ndele++;            /* one more message deleted */
          }
          else PSOUT ("-ERR No such message\015\012");
        }
        else PSOUT ("-ERR Missing message number argument\015\012");
      }

      else if (!strcmp (s,"NOOP"))
        PSOUT ("+OK No-op to you too!\015\012");
      else if (!strcmp (s,"LAST")) {
        sprintf (tmp,"+OK %lu\015\012",last);
        PSOUT (tmp);
      }
      else if (!strcmp (s,"RSET")) {
        rset ();                /* reset the mailbox */
        PSOUT ("+OK Reset state\015\012");
      }
      else if (!strcmp (s,"TOP")) {
        if (t && *t && (i =strtoul (t,&s,10)) && (i <= nmsgs) &&
            (msg[i] > 0)) {
                                /* skip whitespace */
          while (isspace (*s)) s++;
          if (isdigit (*s)) {   /* make sure line count argument good */
            MESSAGECACHE *elt = mail_elt (stream,msg[i]);
            j = strtoul (s,NIL,10);
                                /* update highest message accessed */
            if (i > last) last = i;
            PSOUT ("+OK Top of message follows\015\012");
                                /* output header */
            t = mail_fetch_header (stream,msg[i],NIL,NIL,&k,FT_PEEK);
            blat (t,-1,k);
                                /* output status */
            sprintf (tmp,STATUS,elt->seen ? "R" : " ",
                     elt->recent ? " " : "O");
            PSOUT (tmp);
            CRLF;               /* delimit header and text */
            if (j) {            /* want any text lines? */
                                /* output text */
               t = mail_fetch_text (stream,msg[i],NIL,&k,FT_PEEK);
                                /* tie off final line if full text output */
               if (j -= blat (t,j,k)) CRLF;
            }
            PBOUT ('.');        /* end of list */
            CRLF;
          }
          else PSOUT ("-ERR Bad line count argument\015\012");
        }
        else PSOUT ("-ERR Bad message number argument\015\012");
      }
      else if (!strcmp (s,"XTND"))
        PSOUT ("-ERR Sorry I can't do that\015\012");
      else PSOUT ("-ERR Unknown TRANSACTION state command\015\012");
      break;
    default:
      PSOUT ("-ERR Server in unknown state\015\012");
      break;
    }
  }
  PFLUSH ();                    /* make sure output finished */
```

```
    if (autologouttime) { /* have an autologout in effect? */
                                /* cancel if no longer waiting for login */
      if (state != AUTHORIZATION) autologouttime = 0;
                                /* took too long to login */
      else if (autologouttime < time (0)) {
         PSOUT ("-ERR Autologout\015\012");
         syslog (LOG_INFO,"Autologout host=%.80s",tcp_clienthost ());
         PFLUSH ();             /* make sure output blatted */
         state = LOGOUT;        /* sayonara */
      }
    }
  }
  if (stream && (state == UPDATE)) {
    mail_expunge (stream);
    syslog (LOG_INFO,"Logout user=%.80s host=%.80s nmsgs=%ld ndele=%ld",
            user,tcp_clienthost (),stream->nmsgs,ndele);
    mail_close (stream);
  }
  else syslog (LOG_INFO,"Logout user=%.80s host=%.80s",user ? user : "???",
               tcp_clienthost ());
  PSOUT (sayonara);             /* "now it's time to say sayonara..." */
  PFLUSH ();                    /* make sure output finished */
  exit (0);                     /* all done */
  return 0;                     /* stupid compilers */
}

/* Clock interrupt
 */

void clkint ()
{
  PSOUT ("-ERR Autologout; idle for too long\015\012");
  syslog (LOG_INFO,"Autologout user=%.80s host=%.80s",user ? user : "???",
          tcp_clienthost ());
  PFLUSH ();                    /* make sure output blatted */
  if (critical) state = LOGOUT; /* badly hosed if in critical code */
  else {                  /* try to gracefully close the stream */
    if ((state == TRANSACTION) && !stream->lock) {
      rset ();
      mail_close (stream);
    }
    state = LOGOUT;
    stream = NIL;
    _exit (1);                  /* die die die */
  }
}


/* Kiss Of Death interrupt
 */

void kodint ()
{
                                /* only if idle */
  if (idletime && ((time (0) - idletime) > KODTIMEOUT)) {
    alarm (0);                  /* disable all interrupts */
    server_init (NIL,NIL,NIL,SIG_IGN,SIG_IGN,SIG_IGN,SIG_IGN);
    PSOUT ("-ERR Received Kiss of Death\015\012");
    syslog (LOG_INFO,"Killed (lost mailbox lock) user=%.80s host=%.80s",
            user ? user : "???",tcp_clienthost ());
    if (critical) state =LOGOUT;/* must defer if in critical code */
    else {                      /* try to gracefully close the stream */
      if ((state == TRANSACTION) && !stream->lock) {
        rset ();
```

```
        mail_close (stream);
      }
      state = LOGOUT;
      stream = NIL;
      _exit (1);          /* die die die */
    }
  }
}


/* Hangup interrupt
 */

void hupint ()
{
  alarm (0);                        /* disable all interrupts */
  server_init (NIL,NIL,NIL,SIG_IGN,SIG_IGN,SIG_IGN,SIG_IGN);
  syslog (LOG_INFO,"Hangup user=%.80s host=%.80s",user ? user : "???",
          tcp_clienthost ());
  if (critical) state = LOGOUT;    /* must defer if in critical code */
  else {                    /* try to gracefully close the stream */
    if ((state == TRANSACTION) && !stream->lock) {
      rset ();
      mail_close (stream);
    }
    state = LOGOUT;
    stream = NIL;
    _exit (1);                      /* die die die */
  }
}


/* Termination interrupt
 */

void trmint ()
{
  alarm (0);                        /* disable all interrupts */
  server_init (NIL,NIL,NIL,SIG_IGN,SIG_IGN,SIG_IGN,SIG_IGN);
  PSOUT ("-ERR Killed\015\012");
  syslog (LOG_INFO,"Killed user=%.80s host=%.80s",user ? user : "???",
          tcp_clienthost ());
  if (critical) state = LOGOUT;    /* must defer if in critical code */
  else {                    /* try to gracefully close the stream */
    if ((state == TRANSACTION) && !stream->lock) {
      rset ();
      mail_close (stream);
    }
    state = LOGOUT;
    stream = NIL;
    _exit (1);                      /* die die die */
  }
}

/* Parse PASS command
 * Accepts: pointer to command argument
 * Returns: new state
 */

int pass_login (char *t,int argc,char *argv[])
{
  char tmp[TMPLEN];
                                    /* flush old passowrd */
  if (pass) fs_give ((void **) &pass);
```

```
  if (!(t && *t)) {               /* if no password given */
    PSOUT ("-ERR Missing password argument\015\012");
    return AUTHORIZATION;
  }
  pass = cpystr (t);              /* copy password argument */
  if (!host) {                    /* want remote mailbox? */
                                  /* no, delimit user from possible admin */
    if (t = strchr (user,'*')) *t++ ='\0';
                                  /* attempt the login */
    if (server_login (user,pass,t,argc,argv)) {
      int ret = mbxopen ("INBOX");
      if (ret == TRANSACTION)     /* mailbox opened OK? */
        syslog (LOG_INFO,"%sLogin user=%.80s host=%.80s nmsgs=%ld/%ld",
                t ? "Admin " : "",user,tcp_clienthost (),nmsgs,stream->nmsgs);
      else syslog (LOG_INFO,"%sLogin user=%.80s host=%.80s no mailbox",
                   t ? "Admin " : "",user,tcp_clienthost ());
      return ret;
    }
  }
#ifndef DISABLE_POP_PROXY
                                  /* remote; build remote INBOX */
  else if (anonymous_login (argc,argv)) {
    syslog (LOG_INFO,"IMAP login to host=%.80s user=%.80s host=%.80s",host,
            user,tcp_clienthost ());
    sprintf (tmp,"{%.128s/user=%.128s}INBOX",host,user);
                                  /* disable rimap just in case */
    mail_parameters (NIL,SET_RSHTIMEOUT,0);
    return mbxopen (tmp);
  }
#endif
                                  /* vague error message to confuse crackers */
  PSOUT ("-ERR Bad login\015\012");
  return AUTHORIZATION;
}

/* Authentication responder
 * Accepts: challenge
 *          length of challenge
 *          pointer to response length return location if non-NIL
 * Returns: response
 */

#define RESPBUFLEN 8*MAILTMPLEN

char *responder (void *challenge,unsigned long clen,unsigned long *rlen)
{
  unsigned long i,j;
  unsigned char *t,resp[RESPBUFLEN];
  if (initial) {          /* initial response given? */
    if (clen) return NIL; /* not permitted */
                                  /* set up response */
    t = (unsigned char *) initial;
    initial = NIL;                /* no more initial response */
    return (char *) rfc822_base64 (t,strlen ((char *) t),rlen ? rlen : &i);
  }
  PSOUT ("+ ");
  for (t = rfc822_binary (challenge,clen,&i),j = 0; j < i; j++)
    if (t[j] > ' ') PBOUT (t[j]);
  fs_give ((void **) &t);
  CRLF;
  PFLUSH ();                      /* dump output buffer */
  resp[RESPBUFLEN-1] = '\0';      /* last buffer character is guaranteed NUL */
  alarm (LOGINTIMEOUT);           /* get a response under timeout */
  clearerr (stdin);               /* clear stdin errors */
```

```
                                /* read buffer */
  while (!PSIN ((char *) resp,RESPBUFLEN)) {
                                /* ignore if some interrupt */
    if (ferror (stdin) && (errno == EINTR)) clearerr (stdin);
    else {
      char *e = ferror (stdin) ?
         strerror (errno) : "Command stream end of file";
      alarm (0);          /* disable all interrupts */
      server_init (NIL,NIL,NIL,SIG_IGN,SIG_IGN,SIG_IGN,SIG_IGN);
      syslog (LOG_INFO,"%s, while reading authentication host=%.80s",
              e,tcp_clienthost ());
      state = UPDATE;
      _exit (1);
    }
  }
  if (!(t = (unsigned char *) strchr ((char *) resp,'\012'))) {
    int c;
    while ((c = PBIN ()) != '\012') if (c == EOF) {
                                /* ignore if some interrupt */
      if (ferror (stdin) && (errno == EINTR)) clearerr (stdin);
      else {
         char *e = ferror (stdin) ?
           strerror (errno) : "Command stream end of file";
         alarm (0);             /* disable all interrupts */
         server_init (NIL,NIL,NIL,SIG_IGN,SIG_IGN,SIG_IGN,SIG_IGN);
         syslog (LOG_INFO,"%s, while reading auth char user=%.80s host=%.80s",
                 e,user ? user : "???",tcp_clienthost ());
         state = UPDATE;
         _exit (1);
      }
    }
    return NIL;
  }
  alarm (0);                    /* make sure timeout disabled */
  if (t[-1] == '\015') --t;     /* remove CR */
  *t = '\0';                    /* tie off buffer */
  return (resp[0] != '*') ?
    (char *) rfc822_base64 (resp,t-resp,rlen ? rlen : &i) : NIL;
}

/* Select mailbox
 * Accepts: mailbox name
 * Returns: new state
 */

int mbxopen (char *mailbox)
{
  unsigned long i,j;
  char tmp[TMPLEN];
  MESSAGECACHE *elt;
  nmsgs = 0;                    /* no messages yet */
  if (msg) fs_give ((void **) &msg);
                                /* open mailbox */
  if (stream = mail_open (stream,mailbox,NIL)) {
    if (!stream->rdonly) {      /* make sure not readonly */
      if (j = stream->nmsgs) {  /* if mailbox non-empty */
         sprintf (tmp,"1:%lu",j);/* fetch fast information for all messages */
         mail_fetch_fast (stream,tmp,NIL);
         msg = (long *) fs_get ((stream->nmsgs + 1) * sizeof (long));
         for (i = 1; i <= j; i++) if (!(elt = mail_elt (stream,i))->deleted) {
           msg[++nmsgs] = i;    /* note the presence of this message */
           if (elt->seen) il = last = nmsgs;
         }
      }
```

```
      sprintf (tmp,"+OK Mailbox open, %lu messages\015\012",nmsgs);
      PSOUT (tmp);
      return TRANSACTION;
    }
    else sayonara = "-ERR Can't get lock.  Mailbox in use\015\012";
  }
  else sayonara = "-ERR Unable to open user's INBOX\015\012";
  syslog (LOG_INFO,"Error opening or locking INBOX user=%.80s host=%.80s",
          user,tcp_clienthost ());
  return UPDATE;
}

/* Blat a string with dot checking
 * Accepts: string
 *          maximum number of lines if greater than zero
 *          maximum number of bytes to output
 * Returns: number of lines output
 *
 * This routine is uglier and kludgier than it should be, just to be robust
 * in the case of a message which doesn't end in a newline.  Yes, this routine
 * does truncate the last two bytes from the text.  Since it is normally a
 * newline and the main routine adds it back, it usually does not make a
 * difference.  But if it isn't, since the newline is required and the octet
 * counts have to match, there's no choice but to truncate.
 */

long blat (char *text,long lines,unsigned long size)
{
  char c,d,e;
  long ret = 0;
                                /* no-op if zero lines or empty string */
  if (!(lines && (size-- > 2))) return 0;
  c = *text++; d = *text++;     /* collect first two bytes */
  if (c == '.') PBOUT ('.');    /* double string-leading dot if necessary */
  while (lines && --size) {     /* copy loop */
    e = *text++;          /* get next byte */
    PBOUT (c);                  /* output character */
    if (c == '\012') {          /* end of line? */
      ret++; --lines;           /* count another line */
                                /* double leading dot as necessary */
      if (lines && size && (d == '.')) PBOUT ('.');
    }
    c = d; d = e;       /* move to next character */
  }
  return ret;
}


/* Reset mailbox
 */

void rset ()
{
  unsigned long i;
  char tmp[20];
  if (nmsgs) {                  /* undelete and unmark all of our messages */
    for (i = 1; i <= nmsgs; i++) { /*  */
      if (msg[i] < 0) {         /* ugly and inefficient, but trustworthy */
        sprintf (tmp,"%ld",msg[i] = -msg[i]);
        mail_clearflag (stream,tmp,i <= il ? "\\Deleted" : "\\Deleted \\Seen");
      }
      else if (i > il) {
        sprintf (tmp,"%ld",msg[i]);
        mail_clearflag (stream,tmp,"\\Seen");
```

```
      }
    }
    last = il;
  }
  ndele = 0;                        /* no more deleted messages */
}

/* Co-routines from MAIL library */


/* Message matches a search
 * Accepts: MAIL stream
 *          message number
 */

void mm_searched (MAILSTREAM *stream,unsigned long msgno)
{
  /* Never called */
}


/* Message exists (i.e. there are that many messages in the mailbox)
 * Accepts: MAIL stream
 *          message number
 */

void mm_exists (MAILSTREAM *stream,unsigned long number)
{
  /* Can't use this mechanism.  POP has no means of notifying the client of
     new mail during the session. */
}


/* Message expunged
 * Accepts: MAIL stream
 *          message number
 */

void mm_expunged (MAILSTREAM *stream,unsigned long number)
{
  unsigned long i = number + 1;
  msg[number] = 0;                  /* I bet that this will annoy someone */
  while (i <= nmsgs) --msg[i++];
}


/* Message flag status change
 * Accepts: MAIL stream
 *          message number
 */

void mm_flags (MAILSTREAM *stream,unsigned long number)
{
  /* This isn't used */
}


/* Mailbox found
 * Accepts: MAIL stream
 *          hierarchy delimiter
 *          mailbox name
 *          mailbox attributes
 */
```

```
void mm_list (MAILSTREAM *stream,int delimiter,char *name,long attributes)
{
  /* This isn't used */
}


/* Subscribe mailbox found
 * Accepts: MAIL stream
 *          hierarchy delimiter
 *          mailbox name
 *          mailbox attributes
 */

void mm_lsub (MAILSTREAM *stream,int delimiter,char *name,long attributes)
{
  /* This isn't used */
}


/* Mailbox status
 * Accepts: MAIL stream
 *          mailbox name
 *          mailbox status
 */

void mm_status (MAILSTREAM *stream,char *mailbox,MAILSTATUS *status)
{
  /* This isn't used */
}

/* Notification event
 * Accepts: MAIL stream
 *          string to log
 *          error flag
 */

void mm_notify (MAILSTREAM *stream,char *string,long errflg)
{
  mm_log (string,errflg); /* just do mm_log action */
}


/* Log an event for the user to see
 * Accepts: string to log
 *          error flag
 */

void mm_log (char *string,long errflg)
{
  switch (errflg) {
  case NIL:                       /* information message */
  case PARSE:                     /* parse glitch */
    break;                        /* too many of these to log */
  case WARN:                      /* warning */
    syslog (LOG_DEBUG,"%s",string);
    break;
  case ERROR:                     /* error that broke command */
  default:                        /* default should never happen */
    syslog (LOG_NOTICE,"%s",string);
    break;
  }
}
```

```
/* Log an event to debugging telemetry
 * Accepts: string to log
 */

void mm_dlog (char *string)
{
  /* Not doing anything here for now */
}


/* Get user name and password for this host
 * Accepts: parse of network mailbox name
 *          where to return user name
 *          where to return password
 *          trial count
 */

void mm_login (NETMBX *mb,char *username,char *password,long trial)
{
                                  /* set user name */
  strncpy (username,*mb->user ? mb->user : user,NETMAXUSER-1);
  strncpy (password,pass,255);    /* and password */
  username[NETMAXUSER] = password[255] = '\0';
}

/* About to enter critical code
 * Accepts: stream
 */

void mm_critical (MAILSTREAM *stream)
{
  ++critical;
}


/* About to exit critical code
 * Accepts: stream
 */

void mm_nocritical (MAILSTREAM *stream)
{
  --critical;
}


/* Disk error found
 * Accepts: stream
 *          system error code
 *          flag indicating that mailbox may be clobbered
 * Returns: abort flag
 */

long mm_diskerror (MAILSTREAM *stream,long errcode,long serious)
{
  if (serious) {          /* try your damnest if clobberage likely */
    syslog (LOG_ALERT,
            "Retrying after disk error user=%.80s host=%.80s mbx=%.80s: %.80s",
            user,tcp_clienthost (),
            (stream && stream->mailbox) ? stream->mailbox : "???",
            strerror (errcode));
    alarm (0);                    /* make damn sure timeout disabled */
    sleep (60);                   /* give it some time to clear up */
    return NIL;
  }
```

```
  syslog (LOG_ALERT,"Fatal disk error user=%.80s host=%.80s mbx=%.80s: %.80s",
          user,tcp_clienthost (),
          (stream && stream->mailbox) ? stream->mailbox : "???",
          strerror (errcode));
  return T;
}


/* Log a fatal error event
 * Accepts: string to log
 */

void mm_fatal (char *string)
{
  mm_log (string,ERROR);  /* shouldn't happen normally */
}
```

Appendix E (SQL – Table Creation)

```
CREATE TABLE `sans`.`alert` (
      adate varchar(5),
      atime varchar(8),
      attack varchar(100),
      misc varchar(100),
      srcip varchar(20),
      srcport varchar(5),
      dstip varchar(20),
      dstport varchar(5),
      INDEX `idxsrcip`(`srcip`)
)
TYPE = MYISAM;

CREATE TABLE `sans`.`scans` (
      adate varchar(5),
      atime varchar(8),
      scans varchar(100),
      srcip varchar(20),
      srcport varchar(5),
      dstip varchar(20),
      dstport varchar(5),
      INDEX `idxsrcip`(`srcip`)
)
TYPE = MYISAM;


create table `sans`.`oos` (
      adate varchar(5),
      atime varchar(8),
      srcip varchar(20),
      srcport varchar(5),
      dstip varchar(20),
      dstport varchar(5),
      protocol varchar(5),
      ttl smallint,
      tos varchar(5),
      id varchar(5),
      iplen smallint,
      dglen smallint,
      flags varchar(8),
      sequence varchar(10),
      ack varchar(5),
      win varchar(1),
      tcplen smallint,
      mss varchar(4),
      ts varchar(10),
      ws smallint,
      sackok varchar(1),
      extradata varchar(1),
      INDEX `idxsrcip`(`srcip`)
)
TYPE = MYISAM;

CREATE TABLE `sans`.`event` (
```

```
      srcip varchar(20),
      srcport varchar(5),
      adate varchar(5),
      atime varchar(8),
      event varchar(100),
      dstip varchar(20),
      dstport varchar(5),
      source varchar(6),
      INDEX `idxsrcip`(`srcip`,`adate`,`atime`)
)
TYPE = MYISAM;


CREATE TABLE `sans`.`stumble` (
      srcip varchar(20),
      source varchar(6),
      INDEX `idxsrcip`(`srcip`)
)
TYPE = MYISAM;
```

Appendix F (PERL Scripts for parsing)

PERL Script to parse the alert file:

```
# Declare the subroutine
sub trimspaces($);

if (open (ATTACK, "alert")) {
         }
         else {
                  die ("Cannot open input file!");
         }

while ($line = <ATTACK>)
{        $dateend = index($line,'[**]');
         if ($dateend > -1)
         {        $msgstart = $dateend + 4;
                  $msgend = index($line,'[**]',$msgstart);
                  if ($msgend > -1)
                  {        $datetime = substr($line,0,$dateend - 1);
                           @datetimeparts = split('-',$datetime);
                           $date = @datetimeparts[0];

                           @timeparts = split('\.',@datetimeparts[1]);
                           $time = @timeparts[0];

                           $msg = substr($line,$msgstart,$msgend - $msgstart);
                           $msg = trimspaces($msg);

                           $detailstart = $msgend + 4;
                           $detail = substr($line,$detailstart,length($line)-$detailstart);
                           $detail = trimspaces($detail);

                           $srcip = '';
                           $srcport ='';

                           $dstip = '';
                           $dstport = '';

                           $attack = '';

                           @msgparts = split(':',$msg);

                           if(@msgparts[0] =~ /spp_portscan/)
                           {        if(@msgparts[1] =~ /portscan status/)
                                    {        $misc = trimspaces(@msgparts[2]);

                                             @attackparts = split(' ',@msgparts[1]);
                                             $attack = @attackparts[0] . " " . @attackparts[1];

                                             @srcipparts = split(':',@attackparts[3]);
                                             $srcip = @srcipparts[0];
                                    }

                                    if(@msgparts[1] =~ /End of portscan/)
                                    {        $misc = trimspaces(@msgparts[2]);

                                             @attackparts = split(' ',@msgparts[1]);
                                             $attack = @attackparts[0] . " " . @attackparts[1] . " " .
@attackparts[2];


                                             @srcipparts = split(':',@attackparts[4]);
                                             $srcip = @srcipparts[0];
                                    }

                                    if(@msgparts[1] =~ /PORTSCAN DETECTED/)
                                    {        $miscstart = index(@msgparts[1],'(') + 1;
                                             $miscend = index(@msgparts[1],')');

                                             $misc = trimspaces(substr(@msgparts[1],$miscstart,$miscend-
$miscstart));

                                             @attackparts = split(' ',@msgparts[1]);
                                             $attack = @attackparts[0] . " " . @attackparts[1];

                                             @srcipparts = split(':',@attackparts[3]);
                                             $srcip = @srcipparts[0];
                                    }
```

```
                        }

                        if(@msgparts[0] =~ /External FTP/)
                        {       @attackparts = split(' ',$msg);
                                $misc = '';
                                $attack = @attackparts[0] . ' ' . @attackparts[1] . ' ' . @attackparts[2] .
' ' . @attackparts[3];

                                @detailparts = split(' ',$detail);

                                @srcipparts = split(':',@detailparts[0]);
                                $srcip = @srcipparts[0];
                                $srcport =  @srcipparts[1];

                                @dstipparts = split(':',@detailparts[2]);
                                $dstip = @dstipparts[0];
                                $dstport =  @dstipparts[1];
                        }

                        if(@msgparts[0] =~ /activity/)
                        {       @attackparts = split(' ',$msg);
                                $misc = @attackparts[0];
                                $attack = @attackparts[1];

                                @detailparts = split(' ',$detail);

                                @srcipparts = split(':',@detailparts[0]);
                                $srcip = @srcipparts[0];
                                $srcport =  @srcipparts[1];

                                @dstipparts = split(':',@detailparts[2]);
                                $dstip = @dstipparts[0];
                                $dstport =  @dstipparts[1];
                        }

                        if(length($attack) < 1)
                        {       $misc = '';
                                $attack = $msg;

                                @detailparts = split(' ',$detail);

                                @srcipparts = split(':',@detailparts[0]);
                                $srcip = @srcipparts[0];
                                $srcport =  @srcipparts[1];

                                @dstipparts = split(':',@detailparts[2]);

                                $dstip = @dstipparts[0];
                                $dstport =  @dstipparts[1];

                        }

                        print "$date" . '%';
                        print "$time" . '%';
                        print "$attack" . '%';
                        print "$misc" . '%';
                        print "$srcip" . '%';
                        print "$srcport" . '%';
                        print "$dstip" . '%';
                        print "$dstport";
                        print "\n";
                }
        }
}

#Remove whitespaces from the start and end of the string
sub trimspaces($)
{       my $string = shift;
        $string =~ s/^\s+//;
        $string =~ s/\s+$//;
        return $string;
}
```

PERL Script to parse the scans file:

```
if (open (ATTACK, "scans")) {
        }
```

```
        else {
                die ("Cannot open input file!");
        }

#open (OUTFILE, "> fin.scan");

while ($line = <ATTACK>)  {
#$line = <ATTACK>;
        @words1 = split (' ',$line);
        $month = @words1[0];
        $day = @words1[1];
        $time = @words1[2];

        if($month =~ /Apr/) { $date="04/$day"};

        @src = split(':',@words1[3]);
        $src_ip = @src[0];
        $src_port = @src[1];

        @dst = split(':',@words1[5]);
        $dst_ip = @dst[0];
        $dst_port = @dst[1];


        $msg = @words1[6];
        for($i=7;(length(@words1[$i]) and !(@words1[$i] =~ /Apr/)) > 0;$i++)
        {       $msg = $msg . " " . @words1[$i];
        }


        print "$date" . '%';
        print "$time" . '%';
        print "$msg" . '%';
        print "$src_ip" . '%';
        print "$src_port" . '%';
        print "$dst_ip" . '%';
        print "$dst_port";
        print "\n";
}
```

PERL Script to parse the oos file:

```
open (OOS, "oos");

$firsttime = 1;
while ($line = <OOS>)
{
        if(($line =~ /\=\+\=\+\=\+\=\+\=\+\=\+\=\+\=\+\=\+\=\+\=\+\=\+\=\+\=\+\=\+\=\+\=\+\=\+\=\+/) or ($firsttime == 1))
        {
                if($firsttime == 0)
                {
                        print "$date" . '%';
                        print "$time" . '%';
                        print "$srcip" . '%';
                        print "$srcport" . '%';
                        print "$dstip" . '%';
                        print "$dstport" . '%';

                        print "$protocol" . '%';
                        print "$ttl" . '%';
                        print "$tos" . '%';
                        print "$id" . '%';
                        print "$iplen" . '%';
                        print "$dglen" . '%';

                        print "$flags" . '%';
                        print "$sequence" . '%';
                        print "$ack" . '%';
                        print "$win" . '%';
                        print "$tcplen" . '%';

                        print "$mss" . '%';
                        print "$ts" . '%';
                        print "$ws" . '%';
                        print "$sackok" . '%';

                        if($extralines > 2)
                        {
                                $extradata = 1;
                        }
```

```
        else
        {
                $extradata = 0;
        }

        print "$extradata";
        print "\n";

        $skipline = <OOS>;
}
$extralines = 0;

# ******************************** Line 1 ********************************
if($firsttime == 1)
{
        $line1 = $line;
}
else
{
        $line1 = <OOS>;
}

@line1parts = split(' ',$line1);
$datetime = @line1parts[0];
@datetimeparts = split('-',$datetime);
$date = @datetimeparts[0];

@timeparts = split('\.',@datetimeparts[1]);
$time = @timeparts[0];

@srcipparts = split(':',@line1parts[1]);
$srcip = @srcipparts[0];
$srcport =  @srcipparts[1];

@dstipparts = split(':',@line1parts[3]);
$dstip = @dstipparts[0];
$dstport =  @dstipparts[1];

# ******************************** Line 2 ********************************
$line2 = <OOS>;

@line2parts = split(' ',$line2);
$protocol = @line2parts[0];

@ttlparts = split(':',@line2parts[1]);
$ttl = @ttlparts[1];

@tosparts = split(':',@line2parts[2]);
$tos = @tosparts[1];

@idparts = split(':',@line2parts[3]);
$id = @idparts[1];

@iplenparts = split(':',@line2parts[4]);
$iplen = @iplenparts[1];

@dglenparts = split(':',@line2parts[5]);
$dglen = @dglenparts[1];

$fragflag = @line2parts[6];



# ******************************** Line 3 ********************************
$line3 = <OOS>;

@line3parts = split(' ',$line3);

$flags = @line3parts[0];
$sequence = @line3parts[2];
$ack = @line3parts[4];
$win = @line3parts[6];
$tcplen = @line3parts[8];



$line4 = <OOS>;
$optionstart = index($line4,'=>') + 2;
$options = trimspaces(substr($line4,$optionstart,length($line4) - $optionstart));

$mss = '';
$mssstart = index($options,'MSS:');
if($mssstart > -1)
{
```

```
                        $mssstart = $mssstart + 4;
                        @mssparts = split(' ',substr($options,$mssstart,length($options) - $mssstart));
                        $mss = @mssparts[0];
                }

                $ts = '';
                $tsstart = index($options,'TS:');
                if($tsstart > -1)
                {
                        $tsstart = $tsstart + 4;
                        @tsparts = split(' ',substr($options,$tsstart,length($options) - $tsstart));
                        $ts = @tsparts[0];
                }

                $ws = '';
                $wsstart = index($options,'WS:');
                if($wsstart > -1)
                {
                        $wsstart = $wsstart + 4;
                        @wsparts = split(' ',substr($options,$wsstart,length($options) - $wsstart));
                        $ws = @wsparts[0];
                }


                $sackok = 0;
                $wsstart = index($options,'SackOk');
                if($wsstart > -1)
                {       $sackok = 1
                }
        }
        $firsttime = 0;
        $extralines++;

}

#Remove whitespaces from the start and end of the string
sub trimspaces($)
{       my $string = shift;
        $string =~ s/^\s+//;
        $string =~ s/\s+$//;
        return $string;
}
```

Appendix G (SQL Script to load database tables)

Script to load parsed file into the alert database table:

```
delete from alert;
load data local infile "/sans/ciapractical/work/alert.ld" into table
alert fields terminated by "%";
```

Script to load parsed file into the scans database table:

```
delete from scans;
load data local infile "/sans/ciapractical/work/scans.ld" into table
scans fields terminated by "%";
```

Script to load parsed file into the oos database table:

```
delete from oos;
load data local infile "/sans/ciapractical/work/oos.ld" into table
oos fields terminated by "%";
```

Appendix H (SQL Script to generate the events table)

```
insert into event select
srcip,srcport,adate,atime,attack,dstip,dstport,'alert' from alert;
insert into event select
srcip,srcport,adate,atime,scans,dstip,dstport,'scans' from scans;
insert into event select
srcip,srcport,adate,atime,'oos',dstip,dstport,'oos' from oos;
insert into stumble select srcip,source from event;
```