Global Information Assurance Certification Paper

Raj Bhatt
GCIA Practical
Sans CDI West
Dec 2001
San Francisco, CA

Feb. 18, 2002
Version: 3.0

ASSIGNMENT 1: The search for a common security analysis environment 3
Introduction 3
Comparative Analysis Of Current Market Products 4
Requirement For Security Environments 7
Conclusion 8
References 9

ASSIGNMENT 2: Detects

Detect #1: Data On A Syn/Ack 10

Detect #2: FormMail Activity 14

Detect #3: Administratively Prohibited ICMP 22

Detect #4: ScanSSH connection attempt 29

Detect #5: pcAnywhere Scan 34

ASSIGNMENT 3: ANALYZE THIS

Introduction 38

Executive Summary 38

Alert Files Analysis - Top Signatures 39

Alert Files Analysis - Top Talkers 44

OOS Files Analysis 46

Scan Files Analysis 54

Selected Registration Information 57

Link Graph 61

Defensive Recommendations 62

Analysis Process 63

References 64

ASSIGNMENT #1: White Paper

The search for a common security analysis environment

Introduction

There are two sides to the security data analysis equation. Raw data is voluminous, time dependant, complex, and comes with problems of scale. On the other side of the equation we have the analyst who comes with all levels of personal skills, education, intuition, and tastes. Security analysis environments attempt to solve this equation by organizing, condensing, manipulating and summarizing the raw security data for presentation to the analyst, in a form that is much more easy to comprehend.

Lets look at word processors in the market for example. All word processors have certain features that, over time, users have taken for granted. Things like spell checking, text alignment, export/import to other formats etc. are available in all word processors in the market today. Contrasting this with security analysis environments we see that the products that are available today vary a lot in features and functionality. The environments seem to have different objectives in mind with respect to the features they provide. If we were to categorize these environments, we could come up with four broad categories.

First we have the IDSes, that simply show events or alarms as they occur. These are simple working environments that simply put a front end on products, products whose main purpose is achieved by the software in the back end.

Next we have the dump analyzers that are environments to allow forensic examination on logged data, most often tcpdump logs. These are usually standalone analysis environments operating on individual or collections of log files.

The third category consists of the log gatherers that collect logs from different devices, and provide an environment to analyze these logs. Log consolidators are often called Meta IDSes because of the higher level cross device analysis they offer.

The fourth category is that of the visualization. Traditionally, security data is usually presented in a grid or tabular fashion. In this last category we have environments that incorporate different data visualization metaphors to help the analyst examine security data. Not many products in the market today have alternative visuals. Most of the work in this area is still at in the hands of research teams.

This paper will examine representative security environments in the market today, and will recommend a checklist of features that all security analysis environments should strive to have.

Comparative Analysis Of Current Market Products

In this section I've selected a few representative products, both commercial and open source, and comment on their security environments.

ISS Real Secure: ISS RealSecure [4] is one of the most widely used commercial IDS in the industry and makes a good representative environment for the first category. ISS has a single real time monitoring and management console, the *RealSecure Manager*. The console is primarily built upon on a grid metaphor that shows alerts as they come in. The console is a windows only program that is not customizable. It incorporates a simple severity based layout that isn't cluttered. Security event data is stored in an external database, upon which third party reporting tools can be used to query and generate reports. The console also comes with a fairly large set of built in reports. Real time events are shown in the grid but the console does have session playback ability. Exporting data off the console isn't provided.

Snort: Snort[5] is the Open Source equivalent, and it offers no built in visualization. Due to its open source origins, many third party open source add-ons have been built around snort. Marty Roesch, the author of snort, is taking on the commercial world as well and his company, Source Fire, has created a console for managing snort. This console is web based. Features of this console include, data aggregation and event analysis tools. Snort can be configured to store its event data into an external database, thereby allowing third party reporting tools to be used using standard ODBC/JDBC driver. Snort Reports is an add-on from Circuits Maxis that generates web-based reports. Snort Snarf is another add-on that reads snort logs and generated HTML reports.

DEMARC: DEMARC [6] is a relatively new product in the market. Available from Demarc Security, this product offers one of best Web Interfaces in the industry today. The interface relies mostly on the grid visual allowing both real time monitoring and forensics search ability. A nice feature is available in the form of the "quick stats" panel. This allows the analyst to see a summary of what's going on in the enterprise at a glance in near real time. A powerful search interface allows slice and dice of the security data. Small colorful web images are very effectively used to show status and severity of events. DEMARC is an excellent example of where web based security environments are heading. The web site has many sample screen shots of the product in action.

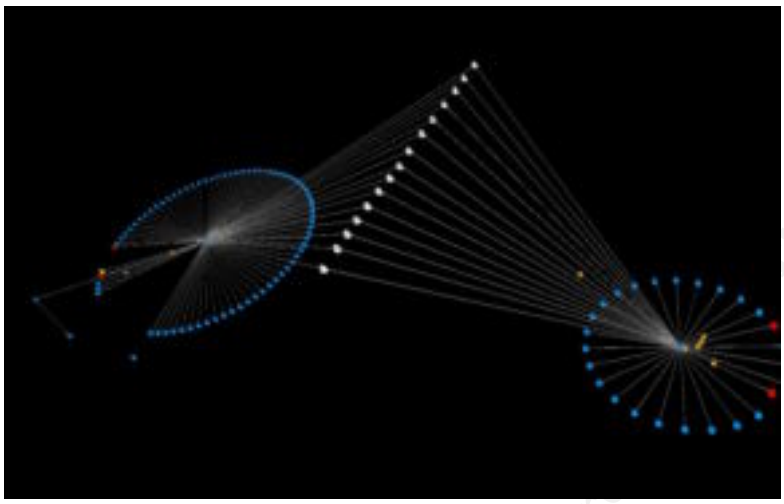
ACID: From Carnegie Mellon University comes the Analysis Console for Intrusion Databases (ACID [7]). Acid is probably the best representative of the dump analyzers category. It is a PHP-based analysis engine to search, and process a database of incidents generated by security software such as Snort and ipchains. ACID is known for its slice and dice ability. It has a powerful query-builder and search interface allowing specification of queries at both the packet headers and payloads. Besides the standard grid metaphor for search results, ACID offers built in charts and statistics generation. A nice feature of the program is incident management, which means that events can be grouped into incidents and treated as one entity amongst several analysts.

Intrusion Vision: Intrusion Vision [8] from General Dynamics (recently acquired the technology from Motorola) takes a highly graphical approach to reporting security events. It has a single visualization that borrows the concept from airline flight traffic controllers. The centerpiece of the console looks like a dartboard or bulls eye, with concentric rings corresponding to severity of events. The rings are further sliced into wedges, with each wedge corresponding to a category of alert. As events are received, they are displayed as small dots on the rings. Multiple dartboards can be used to support an event classification hierarchy, and the view naturally allows for drill down and drills up. Different shades of colors are used when multiple alerts are to be displayed in the same "wedge". Specific alert details are available in windows surrounding the ring display reducing clutter, and allowing the analyst to focus on the center. Perhaps a little counter-intuitive, more severe alerts are displayed in the outer rings. The following picture is a screenshot, showing the dartboard visual.

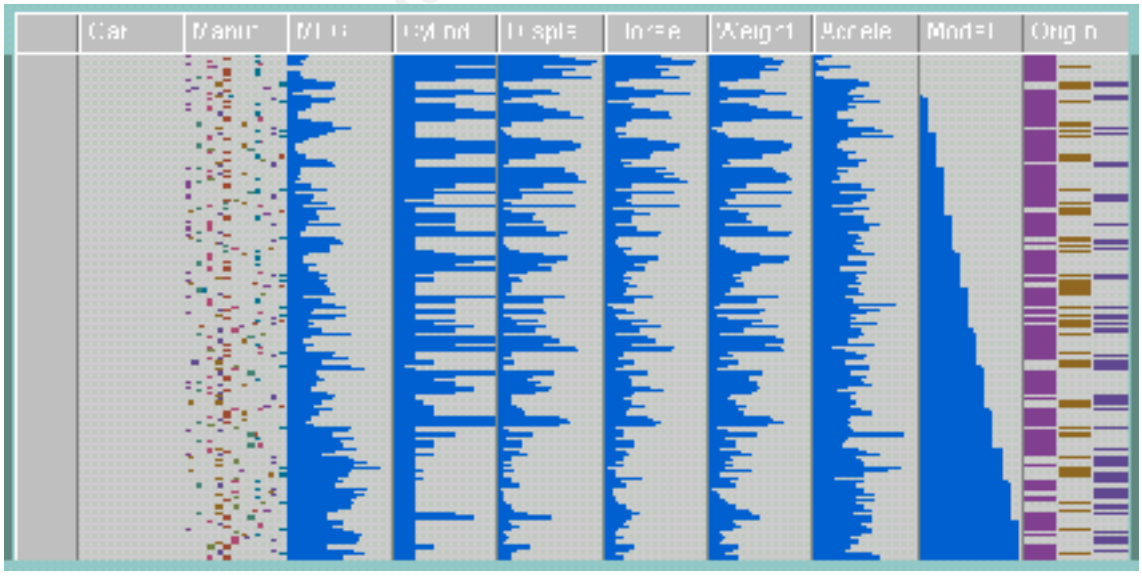
Netforensics: Netforensics [9] is a log consolidator and serves as a good example of the category of Meta-IDSes. It is known in the industry for the strength of its reporting feature. The product has a lot of built in reports over the consolidated security events. The primary interface is HTML based using a web browser. The product allows for historical/forensics analysis, and offers “multidimensional drilldown” using their reporting architecture. By building in parameterization of their reports, and allowing for report result filtering, the product allows the analyst to extract subsections of larger data sets for further analysis. The product also has the ability to export data from a report.

E-Sentinal*:* E-Sentinal [10] is the log consolidation product from E-Security. It has a UNIX based console that is built upon the standard grid visual. Lately, they have been demonstrating a new visual where they can take an image, and super-impose objects onto it. These objects represent real life security entities such as firewalls, routers, etc. When an event matching an entity is encountered, its corresponding superimposed object is made to visually blink.

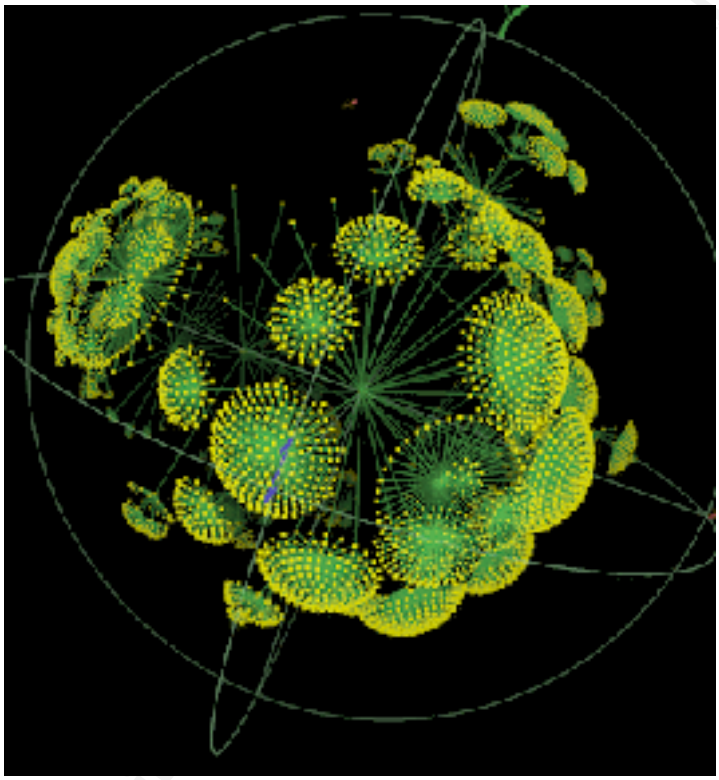
Silent Runner: One of the products in the market that offers alternate visualizations is Silent Runner [11], which is not really an IDS, but rather a network traffic analysis tool with advanced monitoring capabilities. It is known for a very unique visualization. It incorporates a 3D visualization capability allowing an analyst to "fly" through the virtual network looking for anomalous behavior.



InXight: InXight [12] is a commercial venture that is commercializing research from Xerox Parc's User interface group. This group has come up with some amazing new visualizations. Other verticals have already adapted visuals from InXight. In particular the Hyperbolic Tree (also known as Star Tree), Cone Tree, Table Lens and Perspective Walls, if adapted to visualizing security information, can be very powerful analytic tools. Qualys, a firm that specializes in scanning for vulnerabilities, is one of the players in the security industry that has adapted the hyperbolic tree in its product feature. This hyperbolic tree can be an extremely powerful concept if applied to security data analysis. The InXight web site has many examples of it in use, and the reader is encouraged to see it in action. The image below shows the table lens visualization, which is another popular visualization from InXight.



CAIDA: CAIDA is the Cooperative Association for Internet Data Analysis. It is a research institute that is really collaboration between the commercial, government and research institutes. Its goal is to provide tools and analyses software for the purpose of maintain the global Internet infrastructure. CAIDA is a source of many graphing, plotting, and mapping tools, and in particular a source of some amazing data visualizations. Plankton [13] is one of their tools for visualizing web traffic. This can very easily be adapted to visualizing security related network traffic. Walrus [14] is another tool for interactive 3-D visualization. An image taken from walrus is shown below.



Requirement For Security Environments

We have looked at a few representative products in the market today. They are all heterogeneous, with few common features. To effectively solve the equation of data analysis, the security environment must exhibit some common requirements. Some of the requirement criteria are listed below.

- The software should provide multiple visual metaphors show selected security events in different views. The most common view today is the Grid or Spread Sheet view. Charts, blinking light views and other 2D/3D views should supplement the grid. Collaboration between commercial, open source and the research community should be fostered, and the great new ideas coming out of universities today can be put to use.

- Ability to Slice and Dice event data according to any criteria the analyst chooses. Here the ability to extract subsections of the event data for detailed inspection is paramount. Filters play an important role in defining exactly what the analyst wishes to see. There should be common UI metaphors to allow the analyst to create and express these filters.
- Speed of queries and ability to control results is also very important. The environment should foster quick querying of the data by incorporating things like maximum rows returned, or retrieve counts only. The analyst most often doesn't know that he is looking for and this allows him to mine the information at a higher level. Speed is important for the "chain of thought". Slow queries and response times, can easily result in the analyst looking for alternative ways to get his job done.
- Power users must be accommodated. Most analysts come from a systems administration environment, where the command line is the way things are done. Security environments consisting of point and click user interfaces, will only be accepted if they accommodate the command line like behavior of power users.
- The security environment should be able to condense the data and present summaries. This is usually implemented in the form of Reports. A large set of reports provided to analyze typical common scenarios is essential, as is the ability to define ad-hoc reports.
- The software should be able to provide both real time and forensics views of the data with the ability to seamlessly query what has happened in the past, and at the same time show what is going on currently.
- The ability to provide data hierarchy drill down where the events from a base layer of a hierarchy into finer and finer layers of the hierarchy, with the purpose of narrowing in to one small area or item. This is where we connect different views of the data, by linking them in orders or hierarchies that make sense to the environment being analyzed.
- The software should be able to import from and export to common data exchange formats. Comma Separated Values (CSV) is a typically used to import data into Spreadsheets. The XML based Intrusion Detection Message Exchange Format (IDMEF) is another common data exchange format.
- Customizability of the user interface is another criterion. Present day visual display systems can be overly cluttered and can lead to unnecessary information overload. Ability to choose and arrange components of the GUI helps the analyst focus on the data analysis at hand.
- The software's supported OS platforms are another criterion. Web based GUI's have seemed to solve this by providing platform independence. Sun's Java cross-platform programming language allows for rich applets and standalone clients.

Conclusion

Comparing the products in the market today, it is obvious that most products use the grid as the default visual for analysis. This dependency on grid, has many advantages, but also has a few flaws. Grid visuals when implemented using the web browser, may be quite frustrating to use, because the user is forced to view the events page by page. On the

other hand, rich client implementations, such as ISS Real Secure suffer from the problem of too many events. When too many events come in, the older events scroll off, and usually the grid isn't able to temporarily pause the scroll.
Quite a few of the products, simply dump their output to a database, and depend on third party reporting as a catch all solution for visualization. Reporting is usually slow, and shows a snapshot of the data in a given time interval. It is excellent for summarization and higher level condensed views, but doesn't fare well when the analyst would like to see a continuously updating view. With reporting, drill down is possible, but has to be built it.
Lots of research is being done in data mining and information visualization. New visuals are constantly being developed. Web and Network traffic analysis visualizations have recently done pretty well, and they can be adapted to the field of security quite easily. Specialized Hardware can also be used for computing intensive visualizations. This approach is quite common used in CAD/CAM software. Lets keep looking for different visualization and data presentation techniques, because it will enable us to detect more patterns and perhaps develop algorithms that prevent security incidents. A quote off the Mitre [3] website perhaps says it all:

"A picture is worth a thousand words; moving pictures, maybe a few billion!"

References

1. Communications of the ACM. Visualizing Everything. August 2001 Vol. 44, Number 8
2. Keller, Peter and Mary. Visual Cues, Practical Data Visualization. IEEE Computer Society Press, 1993
3. Gershon, Nahum. Insights on information visualizations. Jan 2002 URL: http://www.mitre.org/jobs/features/storytelling/
4. Product Users Guide. ISS Real Secure. URL: http://documents.iss.net/literature/RealSecure/RS_WGM_UG_6.0.pdf
5. Product Home Page. Snort. URL: http://www.snort.org
6. Product Home Page. DEMARC. URL: http://www.demarc.com/
7. Product Home Page. ACID. URL: http://www.cert.org/kb/acid/
8. Product Home Page: Intrusion Vision. URL: http://www.gd-decisionsystems.com/intrusionvision/
9. Product Home Page. NetForensics. URL: http://www.netforensics.com/
10. Product Home Page. E-Sentinal. URL: http://www.esecurityinc.com/products/oesp.asp
11. Product Home Page. Silent Runner. URL: http://www.silentrunner.com/
12. Products Page. InXight. URL: http://www.inxight.com/products
13. Plankton Visualization. CAIDA URL: http://www.caida.org/tools/visualization/plankton
14. Walrus Visualization. CAIDA URL: http://www.caida.org/tools/visualization/ walrus

Detect #1: Data On A Syn/Ack

Source of Trace

This detect was submitted to the intrusions mailing list that is being maintained by the folks at incidents.org. Brent Erickson submitted it to the mailing list on January 17th, 2002.

Detect was generated by

Mr. Erickson provided information from two Snort deployments, both at version 1.8.3, build #88. In addition, he provided a description of the snort rule that that triggered these alerts. Both deployments

Probability the source address was spoofed

Low. The packet payload contains data that resemble an attack to a windows HTTP server, and this would mean that the attacker was expecting a response from the server.

Description of the Attack

This was a rouge packet that was seen on Mr. Erickson's Network. He has two implementations of the snort IDS. One is placed on the outside, which he calls Snort System A, and one on the inside, which he calls Snort B. I have pasted the information he provided from both systems, the DNS resolution, a whois lookup, and the snort rule below. In addition to all this, Mr. Erickson states in his email, *"The address and the subnet on our network do not exist"*.

Information from Snort A:
[1:0:0] Suspicious Probe SYN-ACK and Data {TCP} 4.3.52.149:36977 ->
yyy.xxx.201.251:3204

Information from Snort B:
[**] Suspicious Probe SYN-ACK and Data [**]
01/16-21:09:06.668401 4.3.52.149:36977 -> yyy.xxx.201.251:3204
TCP TTL:225 TOS:0x0 ID:48556 IpLen:20 DgmLen:54 DF
***A**S* Seq: 0xF466ABE Ack: 0x657A87D3 Win: 0x17DE TcpLen: 20
63 6D 64 2E 65 78 65 3F 2F 63 2B 64 69 72 cmd.exe?/c+dir

Snort Rule:
alert tcp $EXTERNAL_NET any -> $HOME_NET any
(msg:"Suspicious Probe SYN-ACK and Data";flags:
SA;dsize: > 1;tag:session,6,packets;)

DNS Resolution:
iplsin1-52-149.biz.dsl.gtei.net (4.3.52.149)

ARIN WHOIS:
GENUITY (NET-GNTY-4-0)
3 Van de Graaff Dr.
Burlington, MA 01803 US
Netname: GNTY-4-0
Netblock: 4.0.0.0 - 4.255.255.255
Maintainer: GNTY

The two snort alerts triggered, because the rule is looking for packets with the Syn and Ack flags set, which also have a payload size greater than 1.

Attack mechanism

This is an attempted stimulus in the hope of triggering a response. It is targeting port 3204, which is assigned by IANA to be the "Network Watcher DB Access" port. Examining the payload, one notices that the packet is probably targeting an HTTP service. "cmd.exe?/c+dir" is a common signature for attacks against HTTP services running on windows. In fact, snort has two rules that explicitly look for this signature, "WEB-IIS cmd.exe access" and "WEB-MISC rcmd attempt".
The attempted stimulus is to trigger a response from a listening HTTP service. It hopes to get a directory listing, by issuing the "dir" command. We also can deduce from the Snort rule, that only one packet was captured from this source because the tag directive in the Snort rule would have kicked in and more packets from 4.3.52.149 would have been collected.

The Syn/Ack with Data triggered these alerts. Normally Syn/Ack packets don't have payloads. Some situations when a packet could have data on Syn/Ack:

- Packet Crafting: The attacker was using a packet generator/injector such as hping2 or winject.
- Bug: There was a bug in the code that generated the packet, resulting in the invalid Flag combinations.
- Packet was corrupted from the source on its way to the destination.

Correlations

I did a search on the Internet for the source address, 4.3.52.149. I found two very interesting matches.

- http://216.120.82.53/blocked.htm
  - This site claims to have blocked 4.3.52.149 because it is worm infected.
- http://msgs.securepoint.com/cgi-bin/get/ids-0112/3.html
  - This is an anonymous message in the secure point IDS mailing list archive dated Dec 4th. The anonymous poster presents IIS logs that show evidence of attempts to access his IIS machine from 4.3.52.149. I've pasted a few entries below.
    19:35:17 **4.3.52.149** GET /scripts/root.exe 404
    19:35:17 **4.3.52.149** GET /MSADC/root.exe 404

19:35:18 **4.3.52.149** GET /c/winnt/system32/cmd.exe 404
19:35:18 **4.3.52.149** GET /d/winnt/system32/cmd.exe 404
19:35:20 **4.3.52.149** GET
/scripts/..%5c../winnt/system32/cmd.exe 200
19:35:20 **4.3.52.149** GET
/scripts/..%5c../winnt/system32/cmd.exe 502
19:35:21 **4.3.52.149** GET
/scripts/..%5c../winnt/system32/cmd.exe 502
19:36:09 **4.3.52.149** GET
/scripts/..%5c../winnt/system32/cmd.exe 502
19:36:12 **4.3.52.149** GET /scripts/..%5c../Admin.dll 500

- http://www.oriole.net/report.html
  This site shows up in the search, as a site that keeps track of code red attacks. 4.3.52.149 doesn't show up on the page, but since the search engine triggered a match, it must have been on the page, at the time the search engine indexed the page's content.

URLs to Cert advisories and to IIS vulnerabilities
http://www.cert.org/advisories/CA-2001-26.html
http://www.cert.org/advisories/CA-2001-12.html
http://www.kb.cert.org/vuls/id/111677

Evidence of active targeting

There is no evidence of targeting within Mr. Erickson's email. I searched the mailing list archives, and found a lot of posting by Mr. Erickson, but didn't see anything with 4.3.52.149. On one hand we a likely a "wrong number" and on the other, we have a strong possibility of the source being infected with a worm, and the worm would be targeting every possible web server.

Severity

Severity = (Criticality + Lethality) –
(System Countermeasures - Network Countermeasures)

Criticality:	1 (The destination doesn't even exist)
Lethality:	3 (Attack is a recon, but can have serious effects)
System Countermeasures:	5 (Assume target has patches applied)
Network Countermeasures:	1 (Assume no protection from Firewall)
Severity:	2

Defensive recommendations

- Simply block all traffic from the source IP, 4.3.52.149 . It is most likely an infected host.
- Ensure you have the latest patches for IIS servers that address the directory traversal vulnerability.

- Get a cleaner tool from Microsoft and user to on machines running IIS servers
  http://www.microsoft.com/Downloads/Release.asp?ReleaseID=31878

Multiple choice test question

In which of the following stages of a Three Way TCP Handshake would you expect to find data within the packet?

- Stage 1, Syn
- Stage 2, Syn/Ack
- Stage 3, Ack
- None of the above

Answer: D

Detect #2: FormMail Activity

Anders Reed Mohn

2002-01-10 20:54:42 4.4.50.27 GET /cgi-bin/formmail.pl **recipient=mangroin51@aol.com**&**subject=http://www.itcompagniet.no/cgi-bin/form mail.pl**& email=iov@mail.chorus.net&=http://www.itcompagniet.no/cgi-bin/formmail.pl 404 335 10 Microsoft+URL+Control+-+6.00.8862 -
2002-01-14 01:35:46 63.253.250.95 POST /cgi-bin/formmail.pl - 405 133 10 - -
2002-01-14 11:16:35 66.19.176.51 GET /cgi-bin/formmail.pl recipient=mangroin51@aol.com&subject=http://www.itcompagniet.no/cgi-bin/form mail.pl&email=iov@mail.chorus.net&=http://www.itcompagniet.no/cgi-bin/formmail.pl 404 335 20 Microsoft+URL+Control+- +6.00.8862 -

Jan

24.181.109.55 [26/Nov/2001:23:29:42 +0900] "GET /cgi-bin/formmail.pl?email=f2%40aol%2Ecom&**subject=www%2Eskwea%2Eco%2Ejp%2Fcgi%2Dbin%2Fformmail%2Epl&recipient=kanbud%40aol%2Ecom&msg=w00t**
172.138.24.237 [08/Dec/2001:21:39:16 +0900] "GET /cgi-bin/formmail.pl?email=f2%40aol%2Ecom&subject=www%2Eskwea%2Eco%2Ejp%2Fcgi%2Dbin%2Fformmail%2Epl&recipient=ciphernotcyphr%40aol%2Ecom&msg=w00t
172.158.127.144 [01/Jan/2002:07:47:13 +0900] "GET /cgi-bin/formmail.pl?email=f2%40aol%2Ecom&subject=www%2Eskwea%2Eco%2Ejp%2Fcgi%2Dbin%2Fformmail%2Epl&recipient=pyrex%40mrearl%2Ecom&msg=w00t
24.167.2.130 [06/Jan/2002:05:22:01 +0900] "GET /cgi-bin/formmail.pl?email=f2%40aol%2Ecom&subject=www%2Eskwea%2Eco%2Ejp%2Fcgi%2Dbin%2Fformmail%2Epl&recipient=consults%40aol%2Ecom&msg=w00t
207.172.11.148 [07/Jan/2002:00:51:13 +0900] "GET /cgi-bin/formmail.pl?email=f2%40aol%2Ecom&subject=www%2Eskwea%2Eco%2Ejp%2Fcgi%2Dbin%2Fformmail%2Epl&recipient=enveemysoul%40email%2Ecom&msg=w00t
63.253.92.110 [10/Jan/2002:09:53:16 +0900] "GET /cgi-bin/formmail.pl?email=f2%40aol%2Ecom&subject=www%2Eskwea%2Eco%2Ejp%2Fcgi%2Dbin%2Fformmail%2Epl&recipient=dre%40email%2Ecom&msg=w00t

Stephen Shephard

(Only first entry shown in detail. URL shown for the rest in the interest of brevity)

```
#(1 - 45137) [Jan 7 2002  0:06] [arachNIDS/226]
IDS226/web-cgi_http-cgi-formmail
IPv4: 209.86.191.62 -> 205.169.91.194
      hlen=5 TOS=0 dlen=368 ID=24237 flags=0 offset=0 TTL=117
chksum=60377
TCP:  port=3804 -> dport: 80  flags=***AP*** seq=3720939
      ack=3442730288 off=5 res=0 win=5840 urp=0 chksum=64066
Payload:  length = 328

000 : 47 45 54 20 2F 63 67 69 2D 62 69 6E 2F 66 6F 72   GET /cgi-bin/for
010 : 6D 6D 61 69 6C 2E 70 6C 3F 72 65 63 69 70 69 65   mmail.pl?recipie
020 : 6E 74 3D 62 61 72 73 73 6F 6D 35 31 40 61 6F 6C   nt=barssom51@aol
030 : 2E 63 6F 6D 26 73 75 62 6A 65 63 74 3D 68 74 74   .com&subject=htt
040 : 70 3A 2F 2F 77 77 77 2E 74 61 63 2D 64 65 6E 76   p://www.tac-denv
050 : 65 72 2E 63 6F 6D 2F 63 67 69 2D 62 69 6E 2F 66   er.com/cgi-bin/f
060 : 6F 72 6D 6D 61 69 6C 2E 70 6C 26 65 6D 61 69 6C   ormmail.pl&email
070 : 3D 6C 61 73 64 67 72 40 61 63 6E 65 74 2E 6E 65   =lasdgr@acnet.ne
080 : 74 26 3D 68 74 74 70 3A 2F 2F 77 77 77 2E 74 61   t&=http://www.ta
090 : 63 2D 64 65 6E 76 65 72 2E 63 6F 6D 2F 63 67 69   c-denver.com/cgi
0a0 : 2D 62 69 6E 2F 66 6F 72 6D 6D 61 69 6C 2E 70 6C   -bin/formmail.pl
0b0 : 20 48 54 54 50 2F 31 2E 31 0D 0A 41 63 63 65 70    HTTP/1.1..Accep
0c0 : 74 3A 20 69 6D 61 67 65 2F 67 69 66 2C 20 69 6D   t: image/gif, im
0d0 : 61 67 65 2F 78 2D 78 62 69 74 6D 61 70 2C 20 69   age/x-xbitmap, i
0e0 : 6D 61 67 65 2F 6A 70 65 67 2C 20 69 6D 61 67 65   mage/jpeg, image
0f0 : 2F 70 6A 70 65 67 2C 20 2A 2F 2A 0D 0A 55 73 65   /pjpeg, */*..Use
100 : 72 2D 41 67 65 6E 74 3A 20 4D 69 63 72 6F 73 6F   r-Agent: Microso
110 : 66 74 20 55 52 4C 20 43 6F 6E 74 72 6F 6C 20 2D   ft URL Control -
120 : 20 36 2E 30 30 2E 38 38 36 32 0D 0A 48 6F 73 74    6.00.8862..Host
130 : 3A 20 77 77 77 2E 74 61 63 2D 64 65 6E 76 65 72   : www.tac-denver
140 : 2E 63 6F 6D 0D 0A 0D 0A                           .com....
```

```
GET /cgi-
bin/formmail.pl?recipient=chewmama69@aol.com&subject=http://www.tac-
denver.com/cgi-bin/formmail.pl&email=octh@visi.com&=http://www.tac-
denver.com/cgi-bin/formmail.pl HTTP/1.1..Accept:
```

```
GET /cgi-
bin/formmail.pl?recipient=jkjdsf7894fask@aol.com&subject=Jill call
me&email=skdjfj84fsdk43@aol.com&=http://www.tac-denver.com/cgi-
bin/formmail.pl
```

```
GET /cgi-
bin/formmail.pl?recipient=mangroin51@aol.com&subject=http://www.tac-
denver.com/cgi-
bin/formmail.pl&email=ukawer@timeworld.com&=http://www.tac-
denver.com/cgi-bin/formmail.pl
```

```
GET /cgi-
bin/formmail.pl?recipient=barssom51@aol.com&subject=http://www.tac-
denver.com/cgi-
bin/formmail.pl&email=lasdgr@acnet.net&=http://www.tac-
denver.com/cgi-bin/formmail.pl
```

```
GET /cgi-
bin/formmail.pl?recipient=chewmama69@aol.com&subject=http://www.tac-
denver.com/cgi-bin/formmail.pl&email=octh@visi.com&=http://www.tac-
denver.com/cgi-bin/formmail.pl
```

```
GET /cgi-
bin/formmail.pl?recipient=jkjdsf7894fask@aol.com&subject=Jill call
me&email=skdjfj84fsdk43@aol.com&=http://www.tac-denver.com/cgi-
bin/formmail.pl .pl
```

```
GET /cgi-
bin/formmail.pl?recipient=mangroin51@aol.com&subject=http://www.tac-
denver.com/cgi-
bin/formmail.pl&email=ukawer@timeworld.com&=http://www.tac-
denver.com/cgi-bin/formmail.pl
```

Source of Trace

This detect was submitted to the intrusions mailing list that is being maintained by the folks at incidents.org. Anders Reed Mohn from Norway submitted it to the mailing list on January 15th, 2002. Four members Jan (from Japan, no last name), Stephen Shepherd, Donna MacLeod and Rich Parker responded to the list with snippets of their logs. Bill Scherr also participated in the thread. Mr. Mohn detecting the sudden upsurge in FormMail alerts brought up the topic to see if anyone else was seeing the same.

Detect was generated by

Detects from each submitter are outlined below:

Anders Mohn	Web Server Log (likely IIS)
Jan	Web Server Log (likely Apache)
Stephen Shepherd	ACID query results, Detected by arachNIDS
Donna MacLeod	Snort
Rich Parker	Web Server Log (likely Apache)

Probability the source address was spoofed

The probability that the source IP is spoofed is low. For this type of attack, the attacker is sending an HTTP request to a CGI program, and is expecting a response back. To send an HTTP request, a TCP session must be established first. On the other hand, the probability that the source Email address is spoofed is quite high.

Description of the Attack

This is an attacked aimed at a common public domain CGI script. FormMail is a CGI script that is written in Perl and is used by many web sites to parse the results of any HTTP form, and email them to a specified user.

Anders Reed Mohn

In the case of Anders Reed Mohn, we see three entries. Since the second one is an HTTP post, we don't have information on the contents of the request.
The first, from IP 4.4.50.27 and the second, from IP 66.19.176.51 are very identical. They are targeting the recipient mangroin51@aol.com, with the subject set to the URL of the CGI form itself.

My suspicion is that this is some sort of script that is scanning multiple web servers looking for formmail. If a vulnerable formmail script is found, the request will be honored, and the email will be sent to mangroin51@aol.com. From the point of view of mangroin51@aol.com, email will arrive with the subject line containing the URL to vulnerable servers. So this leads to a premise that the someone with access to the "mangroin" account is running the script. I also notice that the requests end in

"&=http..", implying that the last parameter isn't defined correctly. This could be a script coding error, and I'd guess that the last parameter would be "&msg=", so that the email sent back would have the URL to the vulnerable formail in both the subject and content.

Jan
Jan from Japan posts a similar trace. In his case, the recipient keeps changing, but again the subject is the URL to the formmail script, lending to a similar, email me back successful detects effect. Since the recipient isn't always the same, I suspect that this is a group of individuals looking for vulnerable servers, and notifying each other. Also the message field is constant across all alerts and is set to implying a code word that a group of individuals would understand.

Stephen Shepherd
Stephen Shepherd sent in a posting of twelve very detailed alerts from his system. Four of them were HTTP POST requests, and didn't show the POST request's content. These have been left out of the traces above.

Here I noticed two things. The recipient kept changing, but the subject wasn't always the URL. When the recipient was jkjdsf7894fask@aol.com, the subject was "Jill Call Me". These observations lead me to suspect that those events are actual spam being sent out exploiting the formmail. I also noticed that in Mr. Shepherd's logs, the following four recipients were being used in the same order:
barssom51@aol, chewmama69@aol.com, jkjdsf7894fask@aol.com, mangroin51@aol.com

Attack mechanism

As shown in the packet traces, the attack mechanism was simply using the HTTP protocol to send form data to the vulnerable CGI script. The FormMail script written by Matt Wright, had a vulnerability in version 1.6 that allow anonymous users to use the script to spam anonymously. Versions 1.7 and 1.8 were released to fix this problem, but didn't address all possible cases of the vulnerability, and the latest version, Version 1.9, was released on August 3rd, 2001. The problem with the script was that it failed to properly handle the IP address of the sender. It would send out email using the IP address of the web server as the sender's address, as opposed to the IP address of the user who submitted the form.

The attack then is a simple form submission to the FormMail CGI script. The email address to spam is used as the value for the "recipient" parameter. Simply fill in the address to spam, and an anonymous email will be sent.
Here is an example that was posted on BugTraq:

*http://www.hum.auc.dk/cgibin/FormMail.pl?**recipient=email@address-to-spam.com**&message=Proof%20that%20FormMail.pl%20can%20be%20used%20to%20send%20anonymous%20spam*

A description of the problem code from BugTraq states that the subroutine check_url is the problem:

```
sub check_url {

  # Localize the check_referer flag which determines if user is valid.    #
  local($check_referer) = 0;

  # If a referring URL was specified, for each valid referer, make sure   #
  # that a valid referring URL was passed to FormMail.                    #

  if ($ENV{'HTTP_REFERER'}) {
    foreach $referer (@referers) {
      if ($ENV{'HTTP_REFERER'} =~ m|https?://([^/]*)$referer|i) {
        $check_referer = 1;
        last;
      }
    }
  }
  else {
    $check_referer = 1;
  }

  # If the HTTP_REFERER was invalid, send back an error.                  #
  if ($check_referer != 1) { &error('bad_referer') }
}
```

An attack can advantage of this code, in multiple ways.

- If the request sent by the attack doesn't contain the 'HTTP_REFERER' header field, then the $check_referer will be set to 1, and the request will be validated.
- The HTTP_REFERER field can easily be spoofed. HTTP requests can be crafted very easily with a spoofed referrer value. For example, telnet to your favorite web server and type in GET / HTTP/1.0<cr> followed by Referer: xxx<cr>, and the referrer will be xxx
- The logic in the line
  if ($ENV{'HTTP_REFERER'} =~ m|https?://([^/]*)$referer|i) {
  can be taken advantage of, by crafting referrer values that will match the regular expression.

The fix to this vulnerability was supplied by Parmeshwar Babu, and consists of an additional check that is performed by the script. This additional code, checks for valid recipients, and doesn't allow email to any arbitrary recipient. The list of valid recipients is configurable. His patch can be found at http://www.mailvalley.com/formmail/

Correlations

Three of the five traces posted, were detailed enough to point to the vulnerability. Formmail has had many variations over time, and it's hard to tell if everything refers to the same version. From the traces provided we notice that there seems to be a small spike in formmail exploits at the beginning of January 2002. The earliest trace, provided by Jan, was dated 26th Nov 2001. We also know that Version 1.9 was released in Aug of 2001.

Looking at the list of source IPs provided, we notice that a fair amount came from mindspring.net, aol.com, and splitrock.net. The mangroin51@aol.com recipient was reported by both Anders Reed Mohn and Stephen Shepherd.

There is a CVE Candidate under consideration, CAN-2001-0357. Here is the URL to the candidate, and to BugTraq URLs from the candidate:

http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0357
http://www.securityfocus.com/archive/1/168177
http://www.securityfocus.com/archive/1/168292
http://www.securityfocus.com/archive/1/168366
http://www.securityfocus.com/archive/1/168345
http://www.securityfocus.com/archive/1/168302
http://www.securityfocus.com/archive/1/168360
http://www.securityfocus.com/archive/1/168633

The BugTraq ID is 2469, and the URL to the page is

http://www.securityfocus.com/bid/2469

Other information found on the Internet:

http://securitytracker.com/alerts/2001/Mar/1001108.html
http://www.nwfusion.com/newsletters/bug/2001/00556960.html
http://www.info-sec.com/internet/01/internet_032701b_j.shtml

Lastly, the following URL shows exactly why this is still a problem today

http://www.monkeys.com/formmailer/

Evidence of active targeting

The detects presented above seem to all point to general scanning of HTTP servers. The scans are looking for formmail scripts, and if one is found, it will be exploited at a later time.

Severity

Severity = (Criticality + Lethality) –
(System Countermeasures - Network Countermeasures)

Criticality: 1

This is random targeting. Assume the script isn't even installed, which can back up by seeing 404 return codes and "script not found" messages in the traces.

Lethality: 2

Attack is a spam, generally an annoyance, unless used as a denial of Service.

System Countermeasures: 3

Lets assume target has the patches applied, but the patches still don't resolve the vulnerability all the way. The script is still open to Regular Expression crafting.

Network Countermeasures: 1

Not much a Firewall can do, but restricting incoming HTTP requests to the script can help.

Severity: -1

Defensive recommendations

Here is a list of recommendations I would make, in no order of priority:

- From the description above, we note that the email sent out by the web server, contains the IP address of the web server, However the web server logs contain the IP address of the spammer. I recommend that the IP address of the spammer be put into the email being sent out, as an extra Email directive.
- To make it a wee bit harder on the spammers, use HTTP POST when designing the HTML form that sends the request to the script, and modify the script to only accept HTTP POST requests.
- Use the patch provided. It uses a static set of referrers so that email will be sent out, only if the HTTP request came from a known address.
- Develop your own in-house script. Stop using these public domain general-purpose scripts.
- Instead of using HTTP to send out email, use an HTML "mailto" link. This will bring up the user's email client, and use that to send the email.
- If you use the formmail script, rename it to something else, so automated scripts don't find it.

Multiple choice test question

You are asked to design an HTML form to send a username and password to a CGI program. To do so you will create an HTML form. Which of the following is the best choice for declaring the form.

A <form name=”foo” method=”GET” action=”bar.cgi”>
B <form name=”foo” method=”POST” action=”bar.cgi”>
C <form name=”foo” method=”GET” action=”/cgi-bin/bar.cgi”>
D <form name=”foo” method=”POST” action=”/cgi-bin/bar.cgi”>

Answer: D (Doesn’t use relative paths, and uses a POST, so you don’t see the password on the URL)

Detect #3: Administratively Prohibited ICMP

Activity #1

1	02/05-22:16:52.864602 210.120.57.11:42304 -> aaa.bbb.ccc.ddd:111 TCP TTL:245 TOS:0x0 ID:49777 IpLen:20 DgmLen:44 DF ******S* Seq: 0x50C8CD8 Ack: 0x0 Win: 0x2238 TcpLen: 24 TCP Options (1) => MSS: 1460
2	02/05-22:16:52.884602 aaa.bbb.ccc.ddd:111 -> 210.120.57.11:42304 TCP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:44 DF ***A**S* Seq: 0xB2F75827 Ack: 0x50C8CD9 Win: 0x16D0 TcpLen: 24 TCP Options (1) => MSS: 1460
3	02/05-22:16:53.034602 210.120.57.11:42304 -> aaa.bbb.ccc.ddd:111 TCP TTL:245 TOS:0x0 ID:49778 IpLen:20 DgmLen:40 DF ***A**** Seq: 0x50C8CD9 Ack: 0xB2F75828 Win: 0x2238 TcpLen: 20
4	02/05-22:16:57.064602 210.120.57.11:42304 -> aaa.bbb.ccc.ddd:111 TCP TTL:245 TOS:0x0 ID:49779 IpLen:20 DgmLen:40 DF ***A***F Seq: 0x50C8CD9 Ack: 0xB2F75828 Win: 0x2238 TcpLen: 20
5	02/05-22:16:57.064602 aaa.bbb.ccc.ddd:111 -> 210.120.57.11:42304 TCP TTL:64 TOS:0x0 ID:62304 IpLen:20 DgmLen:40 DF ***A***F Seq: 0xB2F75828 Ack: 0x50C8CDA Win: 0x16D0 TcpLen: 20
6	02/05-22:16:57.214602 210.120.57.11:42304 -> aaa.bbb.ccc.ddd:111 TCP TTL:245 TOS:0x0 ID:49780 IpLen:20 DgmLen:40 DF ***A**** Seq: 0x50C8CDA Ack: 0xB2F75829 Win: 0x2238 TcpLen: 20

Activity #2

1	02/06-05:19:37.064602 210.120.57.11:957 -> aaa.bbb.ccc.ddd:111 TCP TTL:245 TOS:0x0 ID:51675 IpLen:20 DgmLen:44 DF ******S* Seq: 0x8739627D Ack: 0x0 Win: 0x2238 TcpLen: 24 TCP Options (1) => MSS: 1460
2	02/06-05:19:37.084602 aaa.bbb.ccc.ddd:111 -> 210.120.57.11:957 TCP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:44 DF ***A**S* Seq: 0xEE459CE1 Ack: 0x8739627E Win: 0x16D0 TcpLen: 24 TCP Options (1) => MSS: 1460
3	02/06-05:19:37.234602 210.120.57.32 -> aaa.bbb.ccc.ddd ICMP TTL:246 TOS:0x0 ID:30345 IpLen:20 DgmLen:56 Type:3 Code:13 DESTINATION UNREACHABLE: PACKET FILTERED ** ORIGINAL DATAGRAM DUMP: aaa.bbb.ccc.ddd:111 -> 210.120.57.11:957 TCP TTL:55 TOS:0x0 ID:0 IpLen:20 DgmLen:44 Seq: 0xEE459CE1 ** END OF DUMP 45 00 00 2C 00 00 40 00 37 06 0D E6 3F C5 EA 9D E..,..@.7...?... D2 78 39 0B 00 6F 03 BD EE 45 9C E1 .x9..o...E..
4	02/06-05:19:40.554602 210.120.57.11:957 -> aaa.bbb.ccc.ddd:111 TCP TTL:245 TOS:0x0 ID:51676 IpLen:20 DgmLen:44 DF ******S* Seq: 0x8739627D Ack: 0x0 Win: 0x2238 TcpLen: 24 TCP Options (1) => MSS: 1460

5	02/06-05:19:40.554602 aaa.bbb.ccc.ddd:111 -> 210.120.57.11:957 TCP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:44 DF ***A**S* Seq: 0xEE459CE1 Ack: 0x8739627E Win: 0x16D0 TcpLen: 24 TCP Options (1) => MSS: 1460
6	02/06-05:19:40.704602 210.120.57.32 -> aaa.bbb.ccc.ddd ICMP TTL:246 TOS:0x0 ID:30361 IpLen:20 DgmLen:56 Type:3 Code:13 DESTINATION UNREACHABLE: PACKET FILTERED ** ORIGINAL DATAGRAM DUMP: aaa.bbb.ccc.ddd:111 -> 210.120.57.11:957 TCP TTL:55 TOS:0x0 ID:0 IpLen:20 DgmLen:44 Seq: 0xEE459CE1 ** END OF DUMP 45 00 00 2C 00 00 40 00 37 06 0D E6 3F C5 EA 9D E..,..@.7...?... D2 78 39 0B 00 6F 03 BD EE 45 9C E1 .x9..o...E..
7	02/06-05:19:40.864602 aaa.bbb.ccc.ddd:111 -> 210.120.57.11:957 TCP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:44 DF ***A**S* Seq: 0xEE459CE1 Ack: 0x8739627E Win: 0x16D0 TcpLen: 24 TCP Options (1) => MSS: 1460
8	02/06-05:19:46.864602 aaa.bbb.ccc.ddd:111 -> 210.120.57.11:957 TCP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:44 DF ***A**S* Seq: 0xEE459CE1 Ack: 0x8739627E Win: 0x16D0 TcpLen: 24 TCP Options (1) => MSS: 1460
9	02/06-05:19:46.954602 210.120.57.11:957 -> aaa.bbb.ccc.ddd:111 TCP TTL:245 TOS:0x0 ID:51677 IpLen:20 DgmLen:44 DF ******S* Seq: 0x8739627D Ack: 0x0 Win: 0x2238 TcpLen: 24 TCP Options (1) => MSS: 1460
10	02/06-05:19:46.954602 aaa.bbb.ccc.ddd:111 -> 210.120.57.11:957 TCP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:44 DF ***A**S* Seq: 0xEE459CE1 Ack: 0x8739627E Win: 0x16D0 TcpLen: 24 TCP Options (1) => MSS: 1460
11	02/06-05:19:47.014602 210.120.57.32 -> aaa.bbb.ccc.ddd ICMP TTL:246 TOS:0x0 ID:30364 IpLen:20 DgmLen:56 Type:3 Code:13 DESTINATION UNREACHABLE: PACKET FILTERED ** ORIGINAL DATAGRAM DUMP: aaa.bbb.ccc.ddd:111 -> 210.120.57.11:957 TCP TTL:55 TOS:0x0 ID:0 IpLen:20 DgmLen:44 Seq: 0xEE459CE1 ** END OF DUMP 45 00 00 2C 00 00 40 00 37 06 0D E6 3F C5 EA 9D E..,..@.7...?... D2 78 39 0B 00 6F 03 BD EE 45 9C E1 .x9..o...E..
12	02/06-05:19:58.864602 aaa.bbb.ccc.ddd:111 -> 210.120.57.11:957 TCP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:44 DF ***A**S* Seq: 0xEE459CE1 Ack: 0x8739627E Win: 0x16D0 TcpLen: 24 TCP Options (1) => MSS: 1460
13	02/06-05:19:59.004602 210.120.57.32 -> aaa.bbb.ccc.ddd ICMP TTL:246 TOS:0x0 ID:30367 IpLen:20 DgmLen:56 Type:3 Code:13 DESTINATION UNREACHABLE: PACKET FILTERED ** ORIGINAL DATAGRAM DUMP: aaa.bbb.ccc.ddd:111 -> 210.120.57.11:957 TCP TTL:55 TOS:0x0 ID:0 IpLen:20 DgmLen:44 Seq: 0xEE459CE1 ** END OF DUMP 45 00 00 2C 00 00 40 00 37 06 0D E6 3F C5 EA 9D E..,..@.7...?... D2 78 39 0B 00 6F 03 BD EE 45 9C E1 .x9..o...E..

Source of Trace

This detect was found monitoring traffic to and from a home based Linux system running RedHat 7.2. It has minimal services installed, and is connected to the Internet via a DSL (static IP).
Note: Target address obfuscated. Renamed to: "aaa.bbb.ccc.ddd"

Detect was generated by

The traffic was collected using tcpdump, and the dump files were run through Snort, using the default rule set that comes with version 1.8.

Probability the source address was spoofed

The first set of packets, established a full three-way TCP connection to the destination, and in this case the source IP wasn't spoofed. For the second set of packets, the probability is high that the source IP was spoofed. Whois information for the source address reveled that the source originates from an ISP in Korea. From the Korean Registry, I got the following:

```
# ENGLISH
IP Address        : 210.120.57.0-210.120.57.255
Network Name      : BORANET-LLINE-ASIA38299D
Connect ISP Name  : BORANET
Connect Date      : 20020125
Registration Date : 20020128

[ Organization Information ]
Orgnization ID    : ORG236218
Org Name          : Asia Sinyong Infomation
State             : SEOUL
Address           : 57-10 Seosomun-Dong Chung-Gu
Zip Code          : 100-110
```

Description of the Attack

This attack can be divided into two activities. One was reconnaissance, to get information on the target, and the other was to attack the target on port 111.

Running Snort on the tcpdump files, generated nine "ICMP Destination Unreachable (Communication Administratively Prohibited)" alerts, all coming from one source address. I've pasted one of the alerts below.

```
[**] [1:485:2] ICMP Destination Unreachable (Communication
Administratively Prohibited) [**]
[Classification: Misc activity] [Priority: 3]
02/06-05:19:40.704602 210.120.57.32 -> aaa.bbb.ccc.ddd
```

```
ICMP TTL:246 TOS:0x0 ID:30361 IpLen:20 DgmLen:56
Type:3  Code:13  DESTINATION UNREACHABLE: PACKET FILTERED
** ORIGINAL DATAGRAM DUMP:
aaa.bbb.ccc.ddd:111 -> 210.120.57.11:957
TCP TTL:55 TOS:0x0 ID:0 IpLen:20 DgmLen:44
Seq: 0xEE459CE1
** END OF DUMP
```

Investigating this source address a bit further, I noticed that this source address had made a successful connection to the destination earlier. To show this, I've broken up the detect into two sets of activity.

- Activity One: 3 Way handshake, and quick exit. Here we see the source connecting to the target, and after a successful connection, immediately terminating the connection.
- Activity Two: Troubled Connection Attempt. The detect table for this activity shows twenty-six packets describing a subsequent connection attempt, about six hours later. I show the first thirteen packets, and the pattern established continues with thirteen more packets

Attack mechanism

Activity #1:
The first set of packets was a simple establishment of a TCP connection using the standard 3-Way handshake. The source came in, established a connection, and quickly disconnected. From the traffic of this activity he was able to determine that the target machine was running a service in port 111 and possibly using OS fingerprinting tools, could tell that the target was a Linux box. Chances are that this attack was part of a larger scan, and that destination was just one of the successful hits. Looking at the packets the sequence numbers and the corresponding ACK numbers are paired correctly and this implies that the source wasn't spoofed.

Activity #2:
The second activity is a lot more interesting. Here is an outline of what is going on, packet by packet, for the first few, and there is a pattern established that continues for the rest of the activity:

1. First SYN packet from source. Trying to establish connection.
2. Target responds with SYN/ACK
3. SYN/ACK from target reaches an intermediate router close to target, and the router rejects the packet, sending an ICMP Destination Unreachable packet back. This ICMP destination Unreachable packet has packet #2 embedded in it. The sequence numbers of packet #2 match that of the embedded packet. This ICMP packet from the router has a TTL of one less than that found in

packet #1, which implies that the rejecting router is one hop away from the source.

4. Source sends another SYN Packet. The sequence number on this packet is the same as that of packet number 1, which implies that this is a retry to #1
5. Target responds again with SYN/ACK.
6. Once again the intermediate router rejects that packet and sends back an ICMP Destination Unreachable packet with packet #5 embedded.
7. Target sends SYN/ACK. This is a retry to packet #2, and it is 3 seconds after packet #2 was sent.
8. Target sends SYN/ACK. This is another retry to packet #2, and it is 6 seconds after #2 was sent.
9. Source sends another retry SYN Packet to packet #1
10. Target responds to #9 with SYN/ACK.
11. Once again the intermediate router rejects that packet and sends back an ICMP Destination Unreachable packet with packet #9 embedded
12. Target sends SYN/ACK. This is another retry to packet #2, and it is 12 seconds after #2 was sent.
13. Once again the intermediate router rejects that packet and sends back an ICMP Destination Unreachable packet with packet #9 embedded.

This pattern of SYN, SYN/ACK, Router Reject, and retry continues until both sides reach their retry limit.

From the second activity, we observe that the traffic from the source isn't getting through to the router. The router is rejecting the traffic, with an ICMP Destination Unreachable packet. This means that the destination doesn't exist from the routers point of view, implying that the source address is spoofed. If we look at the TTL values, we notice that in activity one, the TTL was 245, and that the first packet's TTL was 245. The ICMP rejection's packet has a TTL of 246. This means that the source is one hop away from the router, and that the originator of the spoof is either one hop away as well, or has crafted the TTL value so that it matches as the spoofed packet passes by the router. Looking at the embedded packet in the ICMP rejection (#3), we see that the TTL is 55. The outgoing TTL was 64 (#2), which says that the router is 9 hops away from the source. Mac addresses don't apply to the analysis because the source only sees the Mac address of the last router that the packet went through.

Port 111 is reserved for the portmapper. The portmapper is basically a small registry program, where programs that wish to offer services register themselves and clients can ask the portmapper for programs by program number to use those services.

Correlations

Over the years there have been many vulnerabilities in RPC and the portmapper implementation, and the latest one that closely applies to this detect is the rpc.statd vulnerability. Below are some URLs to this vulnerability, and other pertinent links.

- Excellent Paper on RpcBind and PortMapper by David Reese
  - http://www.sans.org/newlook/resources/IDFAQ/blocking.htm
- BugTraq ID 1480
  - http://www.securityfocus.com/bid/1480
- CVE
  - http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2000-0800
  - http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2000-0666
- CERT Advisory on rpc.statd
  - http://www.cert.org/advisories/CA-2000-17.html
- DShield lists 111 as one of the hot ports.
  - http://www1.dshield.org/ports/port111.html
  - http://www1.dshield.org/port_report.php?port=111

Evidence of active targeting

There was no prior evidence of activity from this host towards the destination. After these two incidents, no further activity was seen. I suppose a case can be made for "selective" targeting. With the first round of packets, the source established the validity of the target port and that the target matched the profile of a host he wanted to attack

Severity

Severity = (Criticality + Lethality) –
(System Countermeasures - Network Countermeasures)

Criticality: 4 (The destination hosts a personal web site)
Lethality: 5 (Attack is extremely dangerous if successful.
Target can be rooted)

System Countermeasures: 3
(This vulnerability is in RedHat 6.2, but the rpc service itself has had much vulnerability historically)
Network Countermeasures: 2
(Firewall in place, but port not blocked)

Severity: 4

Defensive recommendations

- Evaluate the need for this service. If it isn't needed, don't even start it
- Block at firewall if for internal use only
- Install latest patches
- Egress Filtering, if an outbound packet has a source address that isn't on your network, drop it.

Multiple choice test question

You are investigating a TCP packet. You suspect that the packet is from a spoofed source address. Which of the following fields would be most useful for your analysis?

a) Time To Live and Source Address
b) Time To Live and Source Mac Address
c) Source Address and Source Mac Address
d) Source Address and Source Port

Answer: A

Detect #4: ScanSSH connection attempt

```
02/11-01:41:03.003534 211.42.142.67:3390 -> 63.197.234.157:22
TCP TTL:49 TOS:0x0 ID:18153 IpLen:20 DgmLen:60 DF
******S* Seq: 0xADA9E43A  Ack: 0x0  Win: 0x7D78  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 37671701 0 NOP WS: 0
```

```
02/11-01:41:03.033534 63.197.234.157:22 -> 211.42.142.67:3390
TCP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:60 DF
***A**S* Seq: 0xEABB7722  Ack: 0xADA9E43B  Win: 0x16A0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 3686399 37671701 NOP
TCP Options => WS: 0
```

```
02/11-01:41:03.253534 211.42.142.67:3390 -> 63.197.234.157:22
TCP TTL:49 TOS:0x0 ID:18458 IpLen:20 DgmLen:52 DF
***A**** Seq: 0xADA9E43B  Ack: 0xEABB7723  Win: 0x7D78  TcpLen: 32
TCP Options (3) => NOP NOP TS: 37671729 3686399
```

```
02/11-01:41:03.373534 63.197.234.157:22 -> 211.42.142.67:3390
TCP TTL:64 TOS:0x0 ID:14132 IpLen:20 DgmLen:75 DF
***AP*** Seq: 0xEABB7723  Ack: 0xADA9E43B  Win: 0x16A0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 3686435 37671729
53 53 48 2D 31 2E 39 39 2D 4F 70 65 6E 53 53 48  SSH-1.99-OpenSSH
5F 32 2E 39 70 32 0A                              _2.9p2.
```

```
02/11-01:41:03.603534 211.42.142.67:3390 -> 63.197.234.157:22
TCP TTL:49 TOS:0x0 ID:18462 IpLen:20 DgmLen:52 DF
***A**** Seq: 0xADA9E43B  Ack: 0xEABB773A  Win: 0x7D78  TcpLen: 32
TCP Options (3) => NOP NOP TS: 37671764 3686435
```

```
02/11-01:41:03.833534 211.42.142.67:3692 -> 63.197.234.157:22
TCP TTL:49 TOS:0x0 ID:18469 IpLen:20 DgmLen:60 DF
******S* Seq: 0xADA10F6E  Ack: 0x0  Win: 0x7D78  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 37671787 0 NOP WS: 0
```

```
02/11-01:41:03.833534 63.197.234.157:22 -> 211.42.142.67:3692
TCP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:60 DF
***A**S* Seq: 0xEB100A30  Ack: 0xADA10F6F  Win: 0x16A0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 3686481 37671787 NOP
TCP Options => WS: 0
```

```
02/11-01:41:04.053534 211.42.142.67:3692 -> 63.197.234.157:22
TCP TTL:49 TOS:0x0 ID:18486 IpLen:20 DgmLen:52 DF
***A**** Seq: 0xADA10F6F  Ack: 0xEB100A31  Win: 0x7D78  TcpLen: 32
TCP Options (3) => NOP NOP TS: 37671809 3686481
```

```
02/11-01:41:04.053534 63.197.234.157:22 -> 211.42.142.67:3692
TCP TTL:64 TOS:0x0 ID:19030 IpLen:20 DgmLen:75 DF
***AP*** Seq: 0xEB100A31  Ack: 0xADA10F6F  Win: 0x16A0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 3686503 37671809
53 53 48 2D 31 2E 39 39 2D 4F 70 65 6E 53 53 48  SSH-1.99-OpenSSH
5F 32 2E 39 70 32 0A                              _2.9p2.
```

```
02/11-01:41:04.283534 211.42.142.67:3692 -> 63.197.234.157:22
TCP TTL:49 TOS:0x0 ID:18491 IpLen:20 DgmLen:52 DF
***A**** Seq: 0xADA10F6F  Ack: 0xEB100A48  Win: 0x7D78  TcpLen: 32
TCP Options (3) => NOP NOP TS: 37671832 3686503
```

```
02/11-01:41:04.283534 211.42.142.67:3692 -> 63.197.234.157:22
TCP TTL:49 TOS:0x0 ID:18492 IpLen:20 DgmLen:80 DF
***AP*** Seq: 0xADA10F6F  Ack: 0xEB100A48  Win: 0x7D78  TcpLen: 32
TCP Options (3) => NOP NOP TS: 37671832 3686503
53 53 48 2D 31 2E 30 2D 53 53 48 5F 56 65 72 73  SSH-1.0-SSH_Vers
69 6F 6E 5F 4D 61 70 70 65 72 0A 00              ion_Mapper..
02/11-01:41:04.283534 63.197.234.157:22 -> 211.42.142.67:3692
TCP TTL:64 TOS:0x0 ID:19031 IpLen:20 DgmLen:52 DF
***A**** Seq: 0xEB100A48  Ack: 0xADA10F8B  Win: 0x16A0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 3686526 37671832
02/11-01:41:04.283534 211.42.142.67:3692 -> 63.197.234.157:22
TCP TTL:49 TOS:0x0 ID:18493 IpLen:20 DgmLen:52 DF
***A***F Seq: 0xADA10F8B  Ack: 0xEB100A48  Win: 0x7D78  TcpLen: 32
TCP Options (3) => NOP NOP TS: 37671832 3686503
02/11-01:41:04.293534 63.197.234.157:22 -> 211.42.142.67:3692
TCP TTL:64 TOS:0x0 ID:19032 IpLen:20 DgmLen:52 DF
***A***F Seq: 0xEB100A48  Ack: 0xADA10F8C  Win: 0x16A0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 3686527 37671832
02/11-01:41:04.293534 211.42.142.67:3390 -> 63.197.234.157:22
TCP TTL:49 TOS:0x0 ID:18494 IpLen:20 DgmLen:52 DF
***A*R** Seq: 0xADA9E43B  Ack: 0xEABB773A  Win: 0x7D78  TcpLen: 32
TCP Options (3) => NOP NOP TS: 37671833 3686435
02/11-01:41:04.293534 63.197.234.157:22 -> 211.42.142.67:3692
TCP TTL:64 TOS:0x0 ID:19033 IpLen:20 DgmLen:52 DF
***A*R** Seq: 0xEB100A49  Ack: 0xADA10F8C  Win: 0x16A0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 3686527 37671832
02/11-01:41:04.513534 211.42.142.67:3692 -> 63.197.234.157:22
TCP TTL:49 TOS:0x0 ID:18507 IpLen:20 DgmLen:52 DF
***A**** Seq: 0xADA10F8C  Ack: 0xEB100A49  Win: 0x7D78  TcpLen: 32
TCP Options (3) => NOP NOP TS: 37671855 3686527
02/11-01:41:04.513534 63.197.234.157:22 -> 211.42.142.67:3692
TCP TTL:255 TOS:0x0 ID:0 IpLen:20 DgmLen:40 DF
*****R** Seq: 0xEB100A49  Ack: 0x0  Win: 0x0  TcpLen: 20
```

Source of Trace

This detect was found monitoring traffic to and from a home based Linux system running RedHat 7.2. The system has a log rotation script that rotates logs every hour. This detect was found in one of the hourly log files. Running Snort on the tcpdump file didn't generate any alerts.

Detect was generated by

The traffic was collected using tcpdump, and the dumpfiles were run through Snort, using the default rule set that comes with version 1.8.

Probability the source address was spoofed

Low. There were two TCP connections made, and each used a full three-way handshake. The source IP originates from an ISP in Korea. From the Korean Registry, I got the following:

ENGLISH
IP Address : 211.42.142.64-211.42.142.127
Network Name : NSB
Connect ISP Name : KOLNET
Connect Date : 20000223
Registration Date : 20000224

[Organization Information]
Orgnization ID : ORG100881
Org Name : N.S.B. CO., LTD.
State : SEOUL
Address : 3F #70-10 MUNJUNG-DONG SONGPA-GU
Zip Code : 138-200

Description of the Attack

The first set of packets was simply establishing a TCP connection using the standard 3-Way handshake. After the connection was established, the target (sshd) pushed its version information out to the source. The source acknowledged it, and quickly turned out an established a second TCP connection.
Once again the sshd pushed its version information to the source. This source then turned around and pushes a packet, surprisingly containing a payload with the string "SSH-1.0-SSH_Version_Mapper.". The destination acknowledged this version mapper packet next. The source didn't continue, and sent a Fin packet to terminate the session, which terminated the second session. Lastly, the source sends another Fin packet to terminate the first session.

To allow cross version portability, the SSH protocol mandates both sides to announce their versions on startup. This explains the payload data we see. The version information is in the format
SSH-<protocol version>-<comment>

Attack mechanism

This attack was generated using a tool that because of the protocol specification has to identify itself. We were able to capture a portion of this identification "SSH-1.0-SSH_Version_Mapper". A search on the Internet pointed to a public domain script called scanssh. This is a scanner that scans IP addresses for SSH servers. From the tool's homepage, typical output looks like this:

```
scanssh -E 10.0.0.0/24 10.1.0.0/25
[...]
10.1.0.124 <timeout>
10.0.0.83 SSH-1.99-OpenSSH_2.3.0p1
```

10.0.0.68 SSH-1.5-OpenSSH_2.3.0
10.0.0.57 <timeout>
10.1.0.9 SSH-1.5-1.2.27
[...]

scanssh turns out to be a simple reconnaissance tool that is used in scanning for open SSH servers. A paper written by the author of the script claims the following:

If we do not receive a reply in a certain period, the TCP SYN packet is resent. This process continues until the retry limit is reached.

This implies that the send connection was made because the first connection's reply wasn't received in time. Looking at the packets, the first connection's was established within one second!! So retry isn't a very good explanation on why there were two connections. I can only guess at this point that it is as software bug, and that two connections are always made for each IP being scanned.

Historically, sshd has had many vulnerabilities on different platforms. The latest vulnerability at the time of this writing is the CRC32 overflow which is an overflow bug, that could allow attackers to write to arbitrary locations of memory.

Correlations

SSH scanning is a very common activity reported. DShield shows SSH as the third most probed port at the time of this writing. The intrusions mailing list from the folks at incidents.org has a ton of SSH probes submitted. For example see Laurie Zirkle's posts from January 23 to January 30, 2002 when she posts an SSH probe almost every day.

From my detect #5, I see that port 22, was also used by older versions of pcAnywhere, and that there may be tools our there that are scanning for port 22 which are looking for pcAnywhere installs.

- Bugtraq:
  - http://www.securityfocus.com/bid/2347
- CVE:
  - http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0144
- Scanssh home page
  - http://www.monkey.org/~provos/scanssh

Evidence of active targeting

Looking at prior logs, there were a lot of SSH connection attempts (probes), but there was no evidence of activity from this particular host.

Severity

Severity = (Criticality + Lethality) –
(System Countermeasures - Network Countermeasures)

Criticality: 4 (The destination hosts a web site and ssh is used for administration)
Lethality: 5 (CRC-32 attack is extremely dangerous if successful. Overflow can be exploited to root the target.)

System Countermeasures: 1
(This system is vulnerable, and hadn't been patched at the time of the detect. It's a new install.)
Network Countermeasures: 1
(Port 22 is opened up in the firewall)

Severity: 7

Defensive recommendations

- Block port 22 at firewall. Allow only sources that are authorized to connect to sshd through the firewall.
- Keep up with the latest versions and security advisories for sshd. Here is the information for RedHad linux 7.2:
  - http://www.redhat.com/mailing-lists/redhat-watch-list/msg00298.html
- If you want to block this tool, write a snort rule that checks for "SSH-1.0-SSH_Version_Mapper", and simply drops the connection using snort's flexresp plugin. The rule could look something like this:

```
alert tcp !$HOME_NET any -> $HOME_NET 22
    (msg: "scanssh zapped!";
    flags: A;
    content: "SSH-1.0-SSH_Version_Mapper ";
    resp:rst_snd)
```

Multiple choice test question

You are examining outgoing traffic from your network and notice an interesting TCP packet. The packet's payload has the string "SSH-1.99-OpenSSH_2.9" embedded within it. Which of the following ports is this packet most likely originating from?

e) Port 80
f) Port 21
g) Port 22
h) Port 23

Answer: C

Detect #5: pcAnywhere Scan

```
01/11/2002 14:14:31.656 - UDP packet dropped - Source:aaa.bbb.ccc.DDD,
1034,
WAN - Destination:aaa.bbb.ccc.EEE, 5632, LAN - -
01/11/2002 14:16:12.048 - UDP packet dropped - Source:aaa.bbb.ccc.DDD,
1040,
WAN - Destination:aaa.bbb.ccc.EEE, 5632, LAN - -
01/11/2002 14:20:31.400 - UDP packet dropped - Source:aaa.bbb.ccc.DDD,
1049,
WAN - Destination:aaa.bbb.ccc.EEE, 5632, LAN - -
01/11/2002 14:26:19.384 - UDP packet dropped - Source:aaa.bbb.ccc.DDD,
1053,
WAN - Destination:aaa.bbb.ccc.EEE, 5632, LAN - -
```

Source of Trace

This detect was found off a personal firewall protecting a small home network.

Note: the source is obfuscated to "aaa.bbb.ccc.DDD" and the destination is obfuscated to "aaa.bbb.ccc.EEE". The destination was obfuscated because it is on the same subnet as the source, which is an observation that is used in the analysis for this detect.

Detect was generated by

This detect was generated by the firewall, which is a SonicWall Internet Appliance, that emails its logs nightly.

Probability the source address was spoofed

Low. The source is on the same subnet at the target, and it is sending UDP packets leading me to believe that this is a scan searching for other pcAnyWhere installations. The source IP originates from the Pacific Bell ISP in the San Francisco Bay Area. The WHOIS information is as follows:

Pac Bell Internet Services (NETBLK-PBI-NET-7) PBI-NET-7
63.192.0.0 - 63.207.255.255
SNFC21 RBACK15 BASIC 63.197.232.0 (NETBLK-SBCIS55741) SBCIS55741
63.197.232.0 - 63.197.235.255

To single out one record, look it up with "!xxx", where xxx is the handle, shown in parenthesis following the name, which comes first.

The ARIN Registration Services Host contains ONLY Internet

Network Information: Networks, ASN's, and related POC's.
Please use the whois server at rs.internic.net for DOMAIN related
Information and whois.nic.mil for NIPRNET Information.

Description of the Attack

This was a UDP scan from where the source was on the same subnet as the destination. The target port was 5632, which is commonly used by the product pcAnywhere from Symantec. The newer versions of the product have a host-browsing feature, that sends out UDP packets to find hosts that have implementations of the software installed. From the pcAnywhere Knowledgebase, the following paragraphs provide a good explanation:

> *When a pcAnywhere remote attempts to connect to a host, it sends User Datagram Protocol (UDP) packets to the host through its "TCPIPStatusPort" port. pcAnywhere uses these packets to determine the name of the host and its status. UDP packets are faster than Transmission Control Protocol (TCP) packets, but their delivery is not guaranteed. If these packets are not returned within a specific time for whatever reason, pcAnywhere generates a "Timeout looking for connection" error message. If you are browsing for hosts, then the browse list will show just the address instead of the host's status, name, and address.*

Attack mechanism

Besides the host-browsing feature, there have been other reported vulnerabilities for this product. I found the following CVE entries:

- http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2000-0273
- http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-1028
- http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2000-0300
- http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2000-0324

Looking at them, the vulnerabilities have been with the TCP connections aspect of the product, and not the UDP based host browsing.

Correlations

- From the pcAnywhere Knowledge Base, older versions of the software used port 22.
- From the sans web site, the following is a pertinent article
  http://www.sans.org/y2k/123199-1220.htm
  In Number 7 (at the bottom of the article), Lenny Zeltser, theorizes that there is a tool out there that is looking for pcAnywhere installations, by alternating

connection attempts to port 22 and port 5063. He notes that in his scan, the source port remained constant, and the target ports alternated.

- "Paranoid Pc Anywhere" by Kris Kistler, is another relevant article on the sans site: http://rr.sans.org/win/paranoid.php.

Evidence of active targeting

Looking at historical logs over the last six months, there was no evidence of activity from this particular host. I didn't find any activity to port 5632 either.

Severity

Severity = (Criticality + Lethality) –
(System Countermeasures - Network Countermeasures)

Criticality: 4
The destination is a workstation with a lot of personal information
Lethality: 1
This was a simple scan looking for open pcAnyWhere installations

System Countermeasures: 5
Target system doesn't run pcAnyWhere
Network Countermeasures: 5
Firewall blocks port 5632

Severity: -5

Defensive recommendations

- Block port 5632 at firewall.
- Avoid using these kinds of sharing software. Use SSH which is comparatively more secure, and is also available for windows.
- Joanne Ashland gives an excellent checklist for DSL and Computer Security Issues:
  - http://rr.sans.org/homeoffice/DSL.php
- From pcAnywhere - security features overview -- Knowledge Base article
  You can prevent a pcAnywhere host from answering a remote scan by creating and setting this Registry DWORD value to 0 HKEY_LOCAL_MACHINE\SOFTWARE\Symantec\pcANYWHERE\CurrentVersion\System\DisplayHostInList

Multiple choice test question

pcAnywhere is a common "remote control" product for windows. Which of the following software products from Microsoft closely resembles the functionality of pcAnywhere?

i) Microsoft Proxy Server
j) Microsoft Terminal Server
k) Microsoft Internet Information Server
l) Microsoft Exchange Server

Answer: B

ASSIGNMENT #3: Analyze This

Introduction

This is a security audit for a university covering five consecutive days of events. The data was broken up into three different kinds of information generated using the popular Snort IDS. Alert files consisting of snort alert information were supplemented by Scan files, which were Snort scan reports. In addition the university provided information regarding Out of Spec packets that were detected on their network.

The following table lists the files I selected for analysis:

alert.011223.txt	oos_Dec.23.2001.txt	scans.011223.txt
alert.011224.txt	oos_Dec.24.2001.txt	scans.011224.txt
alert.011225.txt	oos_Dec.25.2001.txt	scans.011225.txt
alert.011226.txt	oos_Dec.26.2001.txt	scans.011226.txt
alert.011227.txt	oos_Dec.27.2001.txt	scans.011227.txt

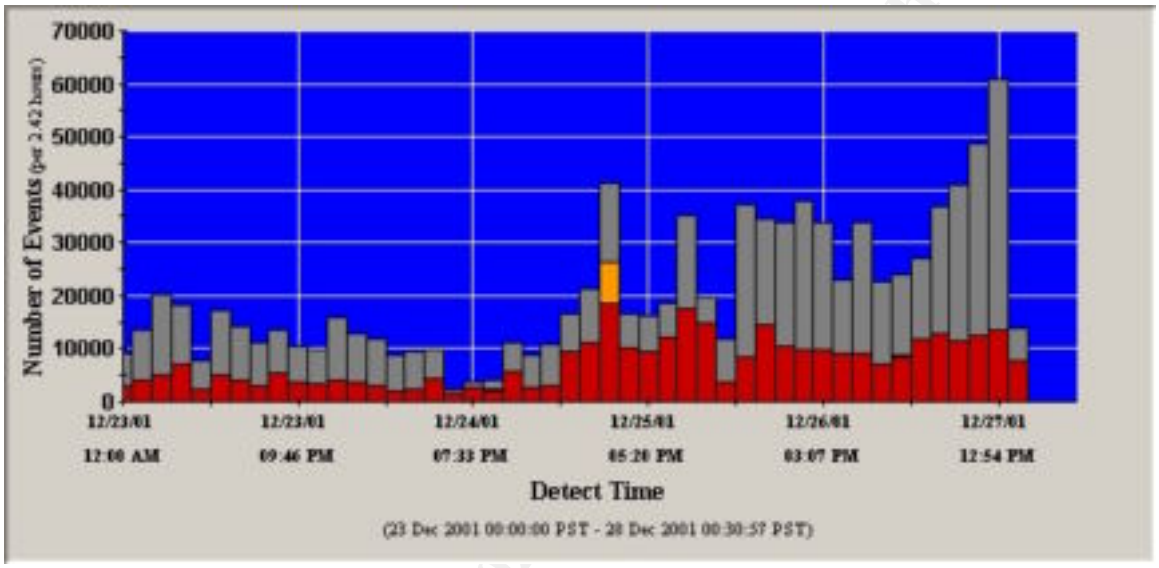
Note*:*
The university's network nodes were presented using the "MY.NET" subnet. For easier data manipulation, I decided to change the MY.NET prefix to a 10.0 prefix and in this report I use MY.NET and 10.0 interchangeably.

Executive Summary

The university has a pretty large internal network and has a liberal policy that allows a lot of access to and from the outside world. For the analysis, I have tried to focus on the top talkers of each category and you will see my report arranged pretty much along the top 10 of each analytical category. This is just one of the many ways to slice and dice the huge amount of information given for analysis.

The total number of alerts analyzed was 9,900,596, which can be broken up as follows. 360,190 came from the alerts files, 8,280 came from the OOS files, and 622,126 came from the scan files. Analysis of the data proved that the university was constantly being scanned. I will show some cases, where I believe the university was attacked, and some cases where university computers were even compromised. As I went through the analysis, I created a "Hotlist". This is a list of offenders that I suspect engaged in malicious activity and that need to be watched. Through out the analysis, the entries in the holist are marked in red and I present registration information for some of these offenders. I later used the hotlist to come up with defensive recommendations for the university.

The following stacked bar chart shows how the data was spread out over the five days I analyzed. The red bar represents data from the Alert files, the yellow represents data from the OOS files, and the gray bar represents the scan data. Right off the bat, we can see that the OOS data was miniscule in comparison to the alerts and scans, and that we had a major spike in OOS data on Christmas day.



Alert Files Analysis - Top Signatures

The first set of files I tackled was the alert files. This would give me an overall picture of the activity over the five days giving me an idea of the talkers, and the kinds of alerts being generated. I noticed that a lot of the alerts were from the spp_portscan signature. This signature kicks in when the portscan plugin of snort detects scan activity. My plan was to analyze scan activity separately, so the analysis results of the alert files that I present below don't include spp_portscan alerts.

The table below presents the top fifteen unique signatures I found in the data set. It shows a count of the number of occurrences of each signature, and also a count of the number of unique source and destination addresses that triggered that signature.

Signature	Occurrences	Sources	Targets
Watchlist 000220 IL-ISDNNET-990517	62330	26	19
MISC traceroute	38927	73	7
CS WEBSERVER - external web traffic	26184	4495	1
MISC source port 53 to <1024	22663	5133	10
ICMP Echo Request BSDtype	13742	25	15
WEB-MISC prefix-get //	13202	669	4
INFO MSN IM Chat data	11931	148	204
ICMP Source Quench	9411	27	94

MISC Large UDP Packet	8528	40	7
ICMP Destination Unreachable (Communication Administratively Prohibited)	5813	63	55
SCAN Proxy attempt	5669	74	4681
Queso fingerprint	5146	43	29
SYN-FIN scan!	5026	1	5026
ICMP Destination Unreachable (Host Unreachable)	4292	334	33
BACKDOOR NetMetro File List	3586	1	1

Top 15 Alerts: Watchlist 000220 IL-ISDNNET-990517:
This signature triggered by far the highest number of alerts. It is a rule that seems to be monitoring incoming traffic from the 212.179 subnet. Outgoing traffic to this subnet doesn't seem to be monitored by this rule, because I didn't find any alerts with target address in the 212.179 subnet. Doing a whois query against RIPE, we see that this is a subnet registered in Israel. Looking at the traffic, most of it seems to be file-sharing traffic, with traffic going to ports 1214 (MP3 and Kazaa), and port 6346 (Gnutella). A large majority of the traffic was just between two nodes, 212.179.35.118:60339 and 10.0.70.70:1214 -- Kazaa (61295 events!).

Top 15 Alerts: MISC traceroute
This signature is a signature that is added to detect traceroute attempts. It usually triggers off a low TTL value in the incoming packet. Looking at the data we see that packets aimed at 10.0.140.9 and 10.0.170.148, with a high target port, triggered the alerts. I also noticed "ICMP Destination Unreachable (Communication Administratively Prohibited)" and "ICMP Destination Unreachable (Host Unreachable)" alerts with these addresses, leading me to suspect that these are routers.

Top 15 Alerts: CS WEBSERVER - external web traffic
This is a simple rule that monitors traffic to the CS (Computer Science?) Web Server, which has address 10.0.100.165. Alerts to the web server seemed to be fairly evenly distributed by source address. The table below lists the top ten external sources triggering this alert.

217.218.2.8	840
210.183.232.26	618
61.129.52.125	595
66.77.74.144	455
64.157.224.117	434
64.157.224.107	376
204.166.111.29	271
66.7.131.157	224
66.7.131.156	194
64.157.224.130	193

Top 15 Alerts: MISC source port 53 to <1024
This signature monitors connection attempts with source port 53 to target ports less than 1024 (non-ephemeral ports). Looking at the time range I selected, I find that all the events are from source port 53 to port 53. Port 53 is used for DNS, and this is most likely zone transfers between the servers. This then implies that we have a lot of false positives. I would recommend changing the rule to trigger when the target port is less than 1024, but not 53. We have four targets for this alert: 10.0.1.2, 10.0.1.3, 10.0.1.4, and 10.0.1.5. These targets also triggered "DNS Zone Transfer" alerts.
10.0.1.2 was also the source in 69 'connect to 515 from inside' alerts. From Mike Bells GCIA practical [6]:

> *This alert is produced by a connection attempt to TCP port 515 originating from within the MY.NET network. There are exploits associated with LPRng, which runs at port 515, allowing execution of arbitrary code or a possible DOS of the printing services.*

Top 15 Alerts: ICMP Echo Request BSDtype
These are simply ICMP ping requests generated by BSD/OS, FreeBSD, NetBSD, OpenBSD 2.5, Linux, or Solaris 2.5-2.7. Their presence usually implies network mapping or other reconnaissance activity. In the selected data range most of the requests were targeted to 10.0.70.148, and coming from 141.213.11.120, 128.223.4.21, and 147.46.59.144. 147.46.59.144 was further logged triggering 41 anonymous FTP requests. I found a correlation on 147.46.59.144. It was a news group posting that was in Korean. The author however had the following English signature that is a lab in the CS university which I think is definitely worth contacting.

```
Cho, KyoungWoon
OS Lab, Dept. of Comp.Sci., Seoul Nat'l Univ.
Tel. 02-880-6571
```
Pag. 012-378-0710

Top 15 Alerts: WEB-MISC prefix-get //
This is a rule that triggers whenever an HTTP request is made without a full URL, but with "//" instead. A high proportion of these alerts were targeting 10.0.253.114 (12797 alerts) and 10.0.253.115 (402 alerts). Thee top three source address were 207.96.37.198, 206.196.188.50, and 208.253.106.26.
Looking further, I notice that 207.96.37.198 the worst offender was hitting the web server quite hard. I suspect some sort of Denial of Service attempt here, because he had 441 alerts on 12/26 from 16:21 to 16:54 (approx ½ hr). Alternatively, it could be retry attempts from an automated script. 206.196.188.50's requests, on the other hand, were more evenly distributed across the five days.

Top 15 Alerts: INFO MSN IM Chat data
This is an informational rule that is monitoring Microsoft Instant Messenger Chat data. I don't know what the university policy is on Instant Messenger usage. The top ten internal users that triggered this alert are:

10.0.5.13	9320
10.0.60.11	3696
10.0.87.50	2336
10.0.60.39	1793
10.0.5.75	975
10.0.223.82	698
10.0.98.200	602
10.0.83.16	569
10.0.83.20	553
10.0.97.186	396

My recommendation would be to check these sources for vulnerabilities associated with MSN Instant Messenger. The following is an excellent article on the dangers of Instant Messaging: http://rr.sans.org/threats/IM.php

Top 15 Alerts: ICMP Source Quench
This alert is checking for source quench ICMP packets. Source quench ICMP packets have type=4, and are used for flow control. The message is a request to decrease the traffic rate of data messages sent to a destination. Inspecting the alerts I notice that this alert primarily comes from 10.0.5.13 and is being sent to multiple internal hosts on the 10.0.200.* subnet. The times of the alerts are distributed.

Top 15 Alerts: MISC Large UDP Packet
This is a standard snort rule that triggers on large UDP packets. Usually the site configures the trigger point, with the default being 4000 bytes. Large UDP packets are unusual, because UDP is inherently a non-reliable protocol. For example, DNS will switch to UDP for large data transfers. Of late large UDP packets are used for MTU discovery, which has the possibility of generating false positives. Looking at the data, I see the following top talkers, of which the first one, 61.150.5.19, accounts for a substantially large proportion.

61.150.5.19:3994	-> 10.0.111.145:3739
216.106.172.149:54567	-> 10.0.153.210:1434
209.249.123.125:16226	-> 10.0.70.192:2872
203.74.13.162:30364	-> 10.0.53.120:4517

Top 15 Alerts: ICMP Destination Unreachable (Communication Administratively Prohibited)
This alert is triggered when by a router when its filter settings explicitly prohibited communication to destination. The generated ICMP packet is of type 3 with a code of 13. When we encounter this, it usually points to network mapping attempts for reconnaissance. The top three external hosts generating this alert are shown below. Digging further, these three don't show up in the data.

65.207.94.30	4483

192.80.43.21	420
141.161.184.45	414

Top 15 Alerts: SCAN Proxy attempt

This is a standard snort signature that triggers on Syn packets to ports 1080 and 8080, which are proxy ports for Wingate/Socks and HTTP. If not properly configured, these proxies can allow anonymous use (misuse actually), and can be used as launching pads for attacks. Analysis shows that a significantly high percentage of these alerts were generated by traffic from 61.155.250.75:1576 to 10.0.253.105:8080. In addition to this, on 12/26 from 6:46 to 7:06 AM (20 Minutes), 65.165.14.43 did a big scan on the 10.0.1 subnet, looking for open 1080 ports.

These two ports are proxy ports, and it's worth looking at the outbound traffic from these ports. For port 8080, there was no outbound traffic in the analysis time frame. For port 1080, the following table lists the internal hosts that show outbound traffic on port 1080.

10.0.84.185	10.0.98.108	10.0.98.179
10.0.97.170	10.0.98.124	10.0.98.181
10.0.97.203	10.0.98.132	10.0.98.189
10.0.97.213	10.0.98.138	10.0.98.230
10.0.97.214	10.0.98.149	10.0.98.242

A quick look at this outbound traffic shows a lot of scan activity, which could imply that attackers are using these proxy ports to launch scans. A correlation from the internet shows that 65.165.14.43 is on the list of code-red offenders. The list can be found here http://www.walkah.net/files/code-red-offenders.txt

Top 15 Alerts: Queso fingerprint

Queso is a program that probes a remote machine with a certain sequence of TCP packets. By analyzing the response packets it can determine the type of operating system that runs on the remote machine, the version of that OS and sometimes it can even give information about the configuration of that machine.

The pattern for Queso scan is typically the SYN bit set, plus the two reserved bits set. The Queso fingerprinting signature is also know for producing many false positives, since the reserved bits may be used by routers for congestion control.

Looking at the data set, we see that this alert was generated by 43 unique sources. Of these sources, by far most alerts were triggered by traffic from 206.65.191.129 (4895 alerts out of 5146). The alerts were targeting 10.0.98.197 and 10.0.98.177. We don't see any outbound traffic to this address. A lookup on 206.65.191.129 shows that it is monitor.dslreports.com, which implies that someone from the university used the dslreports scanning service to scan the two target hosts mentioned above. I say someone from the university, because dslreports will need explicit permission with valid email addresses to launch a scan. A lot of correlations on 206.65.191.129 can be seen on the internet, and explanations that they are from a scanning service.

Top 15 Alerts: SYN-FIN scan!
This signature triggers on packers that have both the Syn and the Fin TCP flags set. Surprisingly, only one source address was responsible for all the 5026 alerts that are in the data set. This source, 24.0.28.234 was responsible for the large Christmas day scan from 21:50 to 22:06, when he scanned the 10.0 subnet. He used port 22 for the source port, and was scanning for open port 22 as well. Port 22 as a source was probably selected because it is likely that firewalls would not block that port.

Top 15 Alerts: ICMP Destination Unreachable (Host Unreachable)
This is a standard snort signature that triggers when ICMP packets of type 3 with code set to 1 are seen. Routers usually send these types of packets when destination isn't reachable. Hosts may also send this packet as well if a port isn't reachable (they really should use type=2, Protocol unreachable)
Looking at the alerts we see external routers telling internal routers of host unreachable outside the network. We don't see the internal host that initiated the packet, but only responses to the internal router that the packet went through.

Most frequent internal routers are
10.0.70.70 1608
10.0.140.9 1191
10.0.137.7 703

Most frequent external routers are
63.146.1.33 - wdc-edge-05.inet.qwest.net
160.36.56.17 - matrix1.cs.utk.edu
209.226.51.33 – New-Liskeard-33.nt.net

Top 15 Alerts: BACKDOOR NetMetro File List
This is a standard snort signature looking for packets that have evidence of the Net Metropolitan Trojan. This Trojan uses ports 5031 or 5032. This signature is also know for generating lots of false positives, when a program uses ports 5031 or 5032 ephemeral ports to connect to a service. Looking at the data, we see that all 3586 alerts with this signature were from 10.0.60.11:20 to 209.49.12.32:5032.

Alert Files Analysis - Top Talkers

To satisfy the top-talkers requirement, I looked at the top fifteen source and destination addresses. The table below lists the top talkers from the alert files.

	Source Addresses		Target Addresses	
	Address	**Count**	**Address**	**Count**
1	212.179.35.118	61327	10.0.70.70	63386
2	10.0.5.13	9320	10.0.140.9	40542
3	24.0.28.234	5027	10.0.100.165	27052

4	206.65.191.129	4908	10.0.253.114	13721
5	61.150.5.19	4690	10.0.70.148	12523
6	65.165.14.43	4668	10.0.1.3	9270
7	65.207.94.30	4483	10.0.1.5	6563
8	141.213.11.120	4272	10.0.1.4	5754
9	128.223.4.21	4130	10.0.98.177	4608
10	147.46.59.144	3893	209.49.12.32	3586
11	10.0.60.11	3696	10.0.153.210	3519
12	216.106.172.149	3518	24.180.204.24	1757
13	10.0.87.50	2336	10.0.130.123	1417
14	63.146.1.33	2133	10.0.98.177	4608
15	10.0.60.39	1793	209.49.12.32	3586

As expected most of these show up in the top signature analysis. The ones that stood out are discussed below:

206.65.191.129: We saw this address in the analysis for Queso Fingerprint. It further stands out because it triggered 11 "Null Scan" events. Null scans are connections attempts to destination port 0. What stands out is that the Null Scans were not within the time span of the large Queso Scan. They were about 10 minutes earlier, and they targeted 10.0.98.187 and 10.0.98.177 only. Null Scans coming from this host are probably another sort of reconnaissance activity. A great network countermeasure is the firewall; because firewalls can be configured to block null scans.

216.106.172.149: We saw this address in the analysis for "Misc Large UDP Packet". We further see that this host was responsible for 116 "Incomplete Packet Fragments Discarded" alerts. Incomplete packet fragments can point to packet crafting or firewall denial of service attacks. However in this case we have a relatively small number of alerts, well distributed through out the alert data, and this host was sending large UDP packets. This can lead us to conclude that the incomplete packet fragments were part of the scan, and not a separate activity.

63.146.1.33: We didn't see this host in the signature analysis. It is responsible for a fairly large scan of the university network as well. The scan it was involved with was a slow ICMP scan that covered all five days. The packets it sent may have been large, because it triggered two "Misc Large ICMP Packet" alerts.

10.0.60.39: We saw this host in the analysis for "INFO MSN IM Chat data". In addition to that, this host was logged for generating 26 "Telnet login incorrect" alerts. I noticed that no more than three alerts were generated for any individual source, implying that a '3 failed attempts and you are out rule' is in place and seems to be working.

10.0.130.123: We haven't seen this host in the signature analysis. This host is running an FTP server. It accounts for 1032 log entries with the "Backdoor NetMetro Incoming traffic" signature. This is quite a large number, and it all came from one

host, 193.252.200.136. It turns out that this host was using port 5031 to connect to the ftp server, and this was done in two sessions one on 12/23 and one on 12/24. No other alerts were seen from 193.252.200.136. This one could go either way. It could be a genuine alert, or could be that on both days, the source just happened to pick port 5031 thereby triggering false positive. This address also triggered some HTTP alerts (61 of them) by sources on the Net-NCFC watchlist.

OOS Files Analysis

The second set of files I tackled was the Out Of Specification files. The tables below presents the top ten source and destination addresses I found in the OOS data. It shows a count of the number of occurrences for each address.

	Source Addresses		Target Addresses	
	Address	Count	Address	Count
1	24.0.28.234	7931	10.0.253.43	104
2	199.183.24.194	104	10.0.6.7	34
3	24.219.121.208	37	10.0.253.114	21
4	65.105.159.22	24	10.0.100.165	19
5	141.157.92.22	17	10.0.1.6	17
6	24.36.185.188	15	10.0.253.125	16
7	202.168.254.178	7	10.0.70.49	16
8	217.226.42.119	7	10.0.6.40	14
9	213.84.157.192	7	10.0.253.41	11
10	204.228.228.145	7	10.0.253.24	10

OOS Source Address: 24.0.28.234

This address by far accounts for most of the OOS alerts found. Out of a total of 8280, 7931 came from this IP, which is greater than 95%. This address was scanning from TCP port 22 targeting port 22, targeting subnet 10.0 The scan was on Christmas day, from 21:50 to 22:12, which maps to the yellow spike in the distribution graph I presented at the beginning of the analysis.

The following table shows sample entries from this scan:

```
12/25-21:50:46.405655 24.0.28.234:22 -> MY.NET.1.2:22
TCP TTL:25 TOS:0x0 ID:39426
**SF**** Seq: 0x7863007   Ack: 0x6D563A98   Win: 0x404
00 00 00 00 00 00                          ......

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
12/25-21:50:46.415952 24.0.28.234:22 -> MY.NET.1.3:22
TCP TTL:25 TOS:0x0 ID:39426
**SF**** Seq: 0x7863007   Ack: 0x6D563A98   Win: 0x404
00 00 00 00 00 00                          ......
```

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
12/25-21:50:46.521709 24.0.28.234:22 -> MY.NET.1.8:22
TCP TTL:25 TOS:0x0 ID:39426
**SF**** Seq: 0x7863007   Ack: 0x6D563A98   Win: 0x404
00 00 00 00 00 00                              ......

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
12/25-21:50:46.526144 24.0.28.234:22 -> MY.NET.1.9:22
TCP TTL:25 TOS:0x0 ID:39426
**SF**** Seq: 0x7863007   Ack: 0x6D563A98   Win: 0x404
00 00 00 00 00 00                              ......

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
12/25-21:50:46.568282 24.0.28.234:22 -> MY.NET.1.12:22
TCP TTL:25 TOS:0x0 ID:39426
**SF**** Seq: 0x7863007   Ack: 0x6D563A98   Win: 0x404
00 00 00 00 00 00                              ......
```

From the scan data, we can see that the Protocol, TTL, TOS, and ID fields remained the same over the period. The Syn and Fin flags were set implying that this was a Syn/Fin scan. I also noticed that the sequence numbers were the same for batches of about 10 to 15 packets. This observation along with the fixed ID fields implies that the packets were crafted. The batches of 10 to 15 could mean that there was one program that was being run in a script. There doesn't seem to be any evidence of attempts to evade the IDS, since the IDS did detect this scan, alerting on the "Syn-Fin Scan!" signature. No outgoing traffic to this address was found in the data I analyzed. I found a very nice correlation by Teri Bidwell [8] called "mystery SF scan tool = Idlescan correlation"[7] where she explains that this kind of scan could have come from the Idlescan tool.

OOS Source Address: 199.183.24.194 104

Traffic from this address, was always to the same destination, 10.0.253.43 on port 25. It was fairly evenly distributed over the time period with a few packets ever hour or so. The Ids and Sequence numbers changed for each packet. Below is a set of sample events

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
12/27-02:18:39.494002 199.183.24.194:36303 -> MY.NET.253.43:25
TCP TTL:52 TOS:0x0 ID:47734  DF
21S***** Seq: 0xB4259067   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 236136977 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=++
12/27-03:42:19.792019 199.183.24.194:47978 -> MY.NET.253.43:25
TCP TTL:52 TOS:0x0 ID:45873  DF
21S***** Seq: 0xEF5E61EA   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 236638939 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
12/27-05:10:01.280108 199.183.24.194:41363 -> MY.NET.253.43:25
TCP TTL:52 TOS:0x0 ID:14106  DF
21S***** Seq: 0x3B63EEC5   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 237165023 0 EOL EOL EOL EOL
```

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
12/27-06:00:41.316855 199.183.24.194:51405 -> MY.NET.253.43:25
TCP TTL:52 TOS:0x0 ID:32161  DF
21S***** Seq: 0xFA30009F   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 237468990 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
12/27-06:10:34.130375 199.183.24.194:57050 -> MY.NET.253.43:25
TCP TTL:52 TOS:0x0 ID:63516  DF
21S***** Seq: 0x1F1D9CD8   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 237528260 0 EOL EOL EOL EOL
```

Observations include that fact that these packets were considered out of spec because the reserve bits are set. The source port was dynamic and ephemeral. The reserved bits are usually set by routers for explicit congestion notification (ECN). Looking at the destination, 10.0.253.43 seems to be a mailserver. Looking at what the IDS caught, we see that it caught the usual portscan activity and also triggered on Queso finger printing. As noted in the Alert File Analysis of Queso, this is one of the reasons why Queso Finger Printing can have false positives. One more interesting alert that caught my eye was that the IDS reported "SMTP chameleon overflow" for 10.0.253.43, which I feel needs more investigation.

199.183.24.194 is a host registered to Red Hat, and from Scott Shinberg's [11] Practical, it is the Linux developers group email server. Given all this, I come to conclude that this is an actual case of congestion, probably kicking in when large email messages are exchanged.

OOS Source Address: 24.219.121.208
OOS Source Address: 65.105.159.22
OOS Source Address: 141.157.92.22
OOS Source Address: 202.168.254.178
OOS Source Address: 217.226.42.119
OOS Source Address: 213.84.157.192
OOS Source Address: 204.228.228.145

These addresses are grouped together for analysis because they exhibit the same behavior. Traffic from the 24.219.121.208 address was always from a high port targeting port 80, 65.105.159.22 was targeting port 25, and 141.157.92.22 was targeting 563. Traffic was in small bursts, and in all cases, the ids and the sequences numbers were not fixed. The bursts may be an attempt to elicit a response from the target system. No other traffic was detected outbound to this IP addresses. Given these three observations, I would come to a conclusion that these were either crafted packets sent to the above targets to incite some sort of response, or like in the case above, these were genuine cases of congestion. Traffic from the 202.168.254.178 and 217.226.42.119 was always from a high port targeting port 80, 213.84.157.192 was Kazaa traffic targeting port 1214, and

204.228.228.145 was targeting 25. I also conclude that traffic to these four source address. In the table below, I show alerts from each one of these addresses.

12/27-19:10:46.961562 **24.219.121.208**:3215 -> MY.NET.6.7:80 TCP TTL:45 TOS:0x0 ID:51118 DF **21S******* Seq: 0xA1A63627   Ack: 0x0   Win: 0x16D0 TCP Options => MSS: 1460 SackOK TS: 221114761 0 EOL EOL EOL EOL
12/27-19:16:14.155597 **65.105.159.22**:41996 -> MY.NET.6.35:25 TCP TTL:44 TOS:0x0 ID:10840 DF **21S******* Seq: 0x90EAEF3D   Ack: 0x0   Win: 0x16D0 TCP Options => MSS: 1460 SackOK TS: 510248293 0 EOL EOL EOL EOL
12/25-12:27:04.631763 **141.157.92.22**:64975 -> MY.NET.1.6:563 TCP TTL:53 TOS:0x0 ID:31549 DF **21S******* Seq: 0x32040EA3   Ack: 0x0   Win: 0x16D0 TCP Options => MSS: 1412 SackOK TS: 299660702 0 EOL EOL EOL EOL
12/23-23:24:11.898487 **202.168.254.178**:52793 -> MY.NET.253.125:80 TCP TTL:39 TOS:0x0 ID:49008 DF **21S******* Seq: 0x56C2ABFB   Ack: 0x0   Win: 0x16D0 TCP Options => MSS: 1460 SackOK TS: 216732811 0 EOL EOL EOL EOL
12/27-16:48:00.761271 **217.226.42.119**:64137 -> MY.NET.100.165:80 TCP TTL:52 TOS:0x0 ID:33366 DF **21S******* Seq: 0x898ACA5D   Ack: 0x0   Win: 0x16D0 TCP Options => MSS: 1412 SackOK TS: 2050059 0 EOL EOL EOL EOL
12/25-22:21:33.723260 213.84.157.192:45672 -> MY.NET.100.236:1214 TCP TTL:51 TOS:0x0 ID:18211 DF **21S******* Seq: 0x100A2A38   Ack: 0x0   Win: 0x16D0 TCP Options => MSS: 1460 SackOK TS: 14970438 0 EOL EOL EOL EOL
12/27-20:45:54.297035 **204.228.228.1**45:40729 -> MY.NET.253.41:25 TCP TTL:50 TOS:0x0 ID:16566 DF **21S******* Seq: 0x9F819C   Ack: 0x0   Win: 0x16D0 TCP Options => MSS: 1460 SackOK TS: 218480938 0 EOL EOL EOL EOL

OOS Source Address: 24.36.185.188

OK, here come the interesting events. A total of fifteen out of spec events were detected from this host. The table below shows all fifteen events.

12/23-20:10:12.508677 24.36.185.188:1621 -> MY.NET.70.49:1214 TCP TTL:116 TOS:0x0 ID:26171 DF **21S**P*U** Seq: 0x27   Ack: 0x903D186A   Win: 0x5010 90 3D 18 6A 2C EA 50 10 00 00 34 A2 00 00 00 00 .=.j,.P...4..... 00 00   .. =+ 12/23-20:17:12.438062 24.36.185.188:1690 -> MY.NET.70.49:1214 TCP TTL:116 TOS:0x0 ID:18068 DF ****SFR*AU** Seq: 0x150030   Ack: 0xFE631868   Win: 0x5010 TCP Options => EOL EOL EOL EOL EOL EOL =+ 12/23-20:17:29.123824 24.36.185.188:1690 -> MY.NET.70.49:1214 TCP TTL:116 TOS:0x0 ID:65173 DF **21SFRP**** Seq: 0x30   Ack: 0xFE63186F   Win: 0x5010

```
TCP Options => EOL EOL EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
12/23-20:19:39.192504 24.36.185.188:1690 -> MY.NET.70.49:1214
TCP TTL:116 TOS:0x0 ID:14499  DF
**SFR*** Seq: 0x30FE63   Ack: 0x21898   Win: 0x5010
06 9A 04 BE 00 30 FE 63 00 02 18 98 03 07 50 10  .....0.c......P.
22 38 CD AA 00 00 00 00 00 00                    "8........

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
12/23-20:26:04.467907 24.36.185.188:1738 -> MY.NET.70.49:1214
TCP TTL:116 TOS:0x0 ID:25029  DF
21**RPA* Seq: 0x39AE11   Ack: 0x17B2   Win: 0x5010
00 39 AE 11 00 00 17 B2 19 DC 50 10 00 00 2A 0D  .9........P...*.
00 00 00 00 00 00                                ......

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
12/23-20:27:16.790582 24.36.185.188:1770 -> MY.NET.70.49:1214
TCP TTL:116 TOS:0x0 ID:53962  DF
*1SF**** Seq: 0x1003E   Ack: 0xDB011770   Win: 0x5010
TCP Options => EOL EOL EOL EOL EOL EOL SackOK EOL Opt 80 (7): AE0A 8700 0000 EOL EOL
EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
12/23-20:30:24.028306 24.36.185.188:1770 -> MY.NET.70.49:1214
TCP TTL:116 TOS:0x0 ID:19422  DF
**SFR*A* Seq: 0x3EDB01   Ack: 0x17F7   Win: 0x8010
36 17 80 10 22 38 CF 87 00 00 01 01 05 0A 17 F7  6..."8..........
3B CB 17 F7                                      ;...

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
12/23-20:31:16.617947 24.36.185.188:1770 -> MY.NET.70.49:1214
TCP TTL:116 TOS:0x0 ID:63456  DF
21SFR*A* Seq: 0x3EDB01   Ack: 0x11809   Win: 0x5010
TCP Options => EOL EOL EOL EOL EOL EOL SackOK

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
12/23-20:33:50.820711 24.36.185.188:0 -> MY.NET.70.49:1770
TCP TTL:116 TOS:0x0 ID:18412  DF
**SF***U Seq: 0x4BE003E   Ack: 0xDB011852   Win: 0x5010
TCP Options => EOL EOL EOL EOL EOL EOL SackOK

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
12/23-20:37:46.889101 24.36.185.188:1770 -> MY.NET.70.49:1214
TCP TTL:116 TOS:0x0 ID:61691  DF
21SF*P*U Seq: 0x3EDB01   Ack: 0x418A2   Win: 0x8010
TCP Options => EOL EOL NOP NOP Sack: 6306@47263 EOL EOL EOL EOL EOL EOL EOL EOL
EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
12/23-20:40:08.084414 24.36.185.188:1770 -> MY.NET.70.49:1214
TCP TTL:116 TOS:0x0 ID:28934  DF
*1SF**A* Seq: 0x4003E   Ack: 0xDB0118D5   Win: 0x5010
00 04 00 3E DB 01 18 D5 19 93 50 10 0B 68 F0 B5  ...>......P..h..
00 00 00 00 00 00                                ......
```

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
12/23-20:46:55.529667 24.36.185.188:1770 -> MY.NET.70.49:1214
TCP TTL:116 TOS:0x0 ID:36152  DF
21SF*P*U Seq: 0x3EDB01   Ack: 0x196D   Win: 0x5010
TCP Options => EOL EOL EOL EOL EOL EOL TS: 148559443 0 EOL EOL EOL EOL EOL EOL EOL
EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
12/23-20:48:32.542712 24.36.185.188:1770 -> MY.NET.70.49:1214
TCP TTL:116 TOS:0x0 ID:54595  DF
*1SF*P** Seq: 0x3EDB01   Ack: 0xEA1983   Win: 0x8010
TCP Options => EOL EOL NOP NOP Sack: 6531@60403 EOL EOL EOL EOL EOL EOL EOL EOL
EOL EOL EOL EOL EOL EOL EOL EOL EOL EOL EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
12/23-21:03:13.367231 24.36.185.188:1770 -> MY.NET.70.49:1214
TCP TTL:116 TOS:0x0 ID:23929  DF
2*SF***U Seq: 0x4003E   Ack: 0xDB011A1A   Win: 0x8010
30 63 80 10 22 38 DA 3A 00 00 01 01 05 0A 1A 1A  0c.."8.:.......
36 17 1A 1A                                      6...

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
12/23-21:14:48.524390 24.36.185.188:4 -> MY.NET.70.49:1770
TCP TTL:116 TOS:0x0 ID:49312  DF
*1SFRP*U Seq: 0x4BE003E   Ack: 0xDB011A9C   Win: 0x8010
DB 01 1A 9C 2A AF 80 10 16 D0 F5 38 00 00 01 01  ....*......8....
05 0A 1A 9C 30 63 1A 9C                          ....0c..

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

My observations for this interesting scan are as follows. The target is 10.0.70.49, which is the help Desk, as proved by the presence of "External FTP to Helpdesk MY.NET.70.49" alerts in the data set. Eight of the alerts are from source port 1770, and two of them are targeting source port 1770. 13 of the 15 alerts are to target port 1214, which is Kazaa/MP3 technology. Looking at the ID field, it is not constant and is changing, but since the packets are not too close together, I couldn't tell if it was incrementing. There is data on the packets, and I noticed the window size alternates between 0x5010 and 0x8010, and also that I don't see the MSS TCP option being used. I can't tell much from the data that is seen in the payload of the packets. The TTL is constant at 116, implying that the source is less than (128-12) 12 hops away.

There is some packet crafting going on here. First, I noticed one of the events is from source port 0. We also see incorrect usage of the TCP Option EOL. EOL is used to pad the final byte of the TCP options to a four byte boundary, and here we see EOL inline and before other options. If the packet wanted to align options inline, a NOP should have been used. Also the fact that

With these observations, I'm lead to conclude some type of network mapping or OS Fingerprinting attempt looking to exploit Kazaa ports with crafted packets.

OOS Target Address: 10.0.253.43
All traffic to this node came from 199.183.24.194. See the source analysis for 199.183.24.194.

OOS Target Address: 10.0.6.7
Most traffic to this node came from 24.219.121.208. See the source analysis for 24.219.121.208. A couple of events came from 193.251.49.8, which resolves to ATuileries-101-1-1-8.abo.wanadoo.fr. Wanadoo has a pretty bad reputation on the intrusions mailing list.

OOS Target Address: 10.0.253.114
Traffic to this node came mostly from 65.129.*.*:18245 to port 21536.
There were 15 events over 3 days. The table below shows a few sample events.

```
12/23-11:09:15.974118 65.129.32.4:18245 -> MY.NET.253.114:21536
TCP TTL:117 TOS:0x0 ID:449  DF
2*SF***U Seq: 0x2F686F6D   Ack: 0x6573756E   Win: 0x7373
65 73 75 6E 2E 63 73 73 20 48 54 54 50 2F 31 2E  esun.css HTTP/1.
31 0D 0A 41 63 63 65 70 74 3A                    1..Accept:
```

```
12/23-11:51:29.193186 65.129.41.99:18245 -> MY.NET.253.114:21536
TCP TTL:120 TOS:0x0 ID:47984  DF
2*SF***U Seq: 0x2F686F6D   Ack: 0x6573756E   Win: 0x7373
65 73 75 6E 2E 63 73 73 20 48 54 54 50 2F 31 2E  esun.css HTTP/1.
31 0D 0A 41 63 63 65 70 74 3A                    1..Accept:
```

```
12/24-15:04:40.755071 65.129.29.16:18245 -> MY.NET.253.114:21536
TCP TTL:117 TOS:0x0 ID:40451  DF
2*SF***U Seq: 0x2F686F6D   Ack: 0x6573756E   Win: 0x7373
65 73 75 6E 2E 63 73 73 20 48 54 54 50 2F 31 2E  esun.css HTTP/1
31 0D 0A 41 63 63 65 70 74 3A                    1..Accept:
```

These packets have hints of crafting, in that the sequence number and ack number remains the same. They events are pretty spread out over time, so cannot be retransmissions. Looking at the payload data, we see hints of HTTP traffic and furthermore, we see that 10.0.253.114 has a web server because of the "WEB-MISC prefix-get //" alerts with it as the destination.

Looking for information on port 21526, I found this correlation that matches pretty well.
http://archives.neohapsis.com/archives/incidents/2000-11/0161.html
A quote from this posting:

> *We have seen it for several months [2] in Poland, these packets are generated by* ***some brain damaged device*** *(I don't know what this is); they would be correct TCP packets if something did not strip TCP header placing HTTP request right after the IP header. Look at the numbers and you'll see that such damaged packet will be resolved to `port 21536 probe' - "GET " resolves to ports 18245 -> 21536.*

OOS Target Address: 10.0.100.165
Once again we see traffic with the reserve bits set. Most came from 202.130.239.149. I do notice that the IP ID was fixed and was set to Zero! This is a source registered to a company called eastern telecom, in China.

OOS Target Address: 10.0.1.6
All came from 141.157.92.22. See the OOS source address analysis for 141.157.92.22.

OOS Target Address: 10.0.253.125
Multiple different source address were found in the traffic to this address. A majority of them exhibited the same behavior as the traffic from 24.219.121.208. The first row in the table below show this. I did see two more unusual packets that exhibit a pattern I have already seen above. These two packets are in row 2 and 3 of the table.

12/23-23:24:16.685997 202.168.254.178:52802 -> MY.NET.253.125:80 TCP TTL:39 TOS:0x0 ID:63496 DF **21S***** Seq: 0x5701EE61 Ack: 0x0 Win: 0x16D0 TCP Options => MSS: 1460 SackOK TS: 216733289 0 EOL EOL EOL EOL
12/23-00:17:15.979376 67.160.235.105:8636 -> MY.NET.253.125:80 TCP TTL:112 TOS:0x0 ID:62322 DF ****SF**** Seq: 0xC98396B1 Ack: 0x93D1EE Win: 0x8010 TCP Options => EOL EOL NOP NOP Sack: 53742@45570 EOL EOL EOL EOL EOL EOL EOL EOL EOL EOL
12/24-22:29:31.510742 65.129.46.147:18245 -> MY.NET.253.125:21536 TCP TTL:120 TOS:0x0 ID:10241 DF ****SFRP*U Seq: 0x2F7E6367 Ack: 0x6568726D Win: 0x7072 31 2F 70 72 65 73 5F 73 69 74 65 2F 70 72 65 73 1/pres_site/pres 63 2E 68 74 6D 6C c.html

OOS Target Address: 10.0.70.49
Alerts to this node were mostly due to traffic from 24.36.185.188. See the OOS source address analysis.

OOS Target Address: 10.0.6.40
All alerts to this node were from high ports on 65.105.159.22. See the OOS source address analysis.

Scan Files Analysis

The last set of files I tackled was the scan files. The tables below presents the top ten source and destination addresses I found in the scan data. It shows a count of the number of occurrences for each address.

	Source Addresses		Target Addresses	
	Address	Count	Address	Count
1	10.0.87.50	401927	24.164.41.210	25847
2	212.95.76.165	20224	10.0.70.148	12213
3	24.138.61.171	16810	24.157.184.117	8206
4	204.152.184.75	11772	24.100.50.113	6032
5	211.248.231.10	9876	24.218.45.138	5848
6	65.165.14.43	9508	24.23.140.185	5574
7	210.58.102.86	7680	67.165.163.5	4355
8	10.0.97.220	6229	24.254.241.95	4055
9	24.44.21.206	5412	64.53.16.160	3997
10	10.0.84.185	5168	213.106.160.152	3655

Scan Source Address: 10.0.87.50

Alerts show mostly UDP traffic from ports 888 and 999. 888 is a port that is associated with CDDB - Music Database access, and 999 is a Trojan port associated with Chat Power, Deep Throat, Foreplay, and WinSatan Trojans. If this is CDDB traffic, why does this IP stand out? Supposedly university students would use CDDB quite a lot, and we would expect a more uniform distribution. We definitely need to look at this one for more Trojan activity.

Scan Source Address: 212.95.76.165

This host did a large scan of the 10.0.subner on 12/27 (over 3Hrs), looking for open 21 ports. I further checked for any outgoing information on this address and as expected I found:

ICMP Destination Unreachable (Protocol Unreachable)
ICMP Destination Unreachable (Communication Administratively Prohibited)

These were triggered when this scan traffic was for unreachable or disallowed targets

Source Address: 24.138.61.171

This host also did a large scan. It scanned subnet 10.0. on 12/27 within 1 Hour.
It was looking for open 21 ports. No outbound traffic except "ICMP Destination Unreachable (Communication Administratively Prohibited)" was found. Note, this host's scan, hit the beetle as well, so "beetle.cs" alerts show up. (See link graph on the beetle)

Scan Source Address: 204.152.184.75
The scan from this address was aimed at only one machine, 10.0.70.148. It was a TCP scan looking for high open ports in the range 1024 to 5000. In the data set I noticed that sometimes hosts are scanned multiple times, implying that the scanner isn't keeping track of previous results. The scan was distributed over all the days if the range I selected for analysis. Looking at outgoing traffic, I got an unexpected surprise. 10.0.70.148 is most likely a compromised machine. I saw a lot of Trojan alerts being triggered. Trojan alerts warrant further investigation, as they can be quite serious. Trojan signatures are typically written using the ports of the packet, and therefore can generate false positives. However, I feel that all Trojan alerts should be considered hostile and definitely need to be looked at for Trojan activity. I've listed the Trojan alerts that were triggered off 10.0.70.148 below, and listed the internal addresses that should be investigated. Anthony Bell [10] wrote a fantastic article on the Adore worm and its variations.

- High port 65535 tcp - possible Red Worm – traffic
  - 10.0.6.35 10.0.6.44 10.0.11.4 10.0.16.42
  - 10.0.17.64 10.0.107.57 10.0.253.42
- IDS50/trojan_trojan-active-subseven
  - 10.0.70.148
  - 10.0.130.123
- Port 55850 tcp - Possible myserver activity - ref. 010313-1
  - 10.0.1.8 10.0.5.29 10.0.6.47 10.0.11.4
  - 10.0.16.42 10.0.60.39 10.0.70.148 10.0.100.165
  - 10.0.111.140 10.0.253.24 10.0.253.114 10.0.253.125

Somewhat related to Trojan activity is virus activity. Virus alerts are triggered when an internal host is sending virus infected email. The virus alerts that I found are

- Virus - Possible pif Worm ,
- Virus - Possible MyRomeo Worm
- Virus - Possible scr Worm

And the hosts that are possibly infected are:

- 10.0.6.7 10.0.6.39 10.0.6.44 10.0.6.59
- 10.0.7.20 10.0.60.17 10.0.100.230

Scan Source Address: 211.248.231.10
This was a slow scan, looking for open port 22 on the 10.0 subnet. No outbound traffic to this address was seen.

Scan Source Address: 65.165.14.43
This was a fast Interleaving scan looking for open ports 21 and 1080 and targeting the 10.0 subnet. No outbound traffic to this address was seen. We note that this is on the hot list for triggering "SCAN Proxy attempt" alerts

Scan Source Address: 210.58.102.86
This was a fast scan, coming from port 21, looking for port 21, which is unusual. Source port 21 was probably selected, because a firewall would let it through. No outbound traffic was seen to this address.

Scan Source Address: 10.0.97.220
This is unique in that it is an internal host that is scanning. The host is seen sending UDP packets to 194.251.249.169 and 216.33.98.254:13 in blocks of 4 to 5 packets. By looking at the UDP packets only, I couldn't tell what was going on. However, I noticed that 10.0.97.220 generated also generated "ICMP Echo Request CyberKit 2.2 Windows" in the alerts file. CyberKit [11] is a collection of network tools for Windows 9x/NT/2000/ME.

Scan Source Address: 24.44.21.206
This was a medium paced scan (relatively speaking!) that was looking for port 21
It scanned the 10.0. subnet. It also triggered some anonymous FTP attempts.

Scan Source Address: 10.0.84.185
This was mostly sending UDP packets to destination port 4665. It also showed up as generating quite a few TCP Syn packets. It doesn't show up in the alert and OOS files. Port 4665 is associated with eDonkey[12], a file-sharing program.

Scan Target Address: 24.164.41.210
For this target address, I noticed all traffic was coming from 10.0.87.50 and directed to port 27500. This is a port used by the Network game Quake. There were no other alerts/oos events with this address. The following URL gives a good overview on this and other network games:
http://www.networkice.com/advice/Exploits/Ports/groups/Quake/default.htm

Scan Target Address: 10.0.70.148
All traffic to this address was from 204.152.184.75 and 62.243.72.50. This host is a compromised host, as detected in the analysis for 204.152.184.75 above.

Scan Target Address: 24.157.184.117
All traffic to this address was from 10.0.87.50, It was from Port 888 to port 27005 which leads me to conclude that this is more network gaming traffic, this time it's the game "Half-Life". Garreth Geremiah's [13] practical offers a correlation. His analysis focused on the scans, and his findings concluded that there was a lot of network gaming going on, and that Half-Life was one of the games. He offers the suggestion that some of the university machines may be running as network game servers. To run these servers, the user needs root privelages, implying that machines may be compromised.

Scan Target Address: 24.100.50.113
Traffic mostly from 10.0.87.50, Again its "Half life" gaming traffic.

Scan Target Address: 24.218.45.138
All the traffic to this address was from 10.0.87.50, UDP from port 888 to port 10122. Suspect another Network game, but unsure.

Scan Target Address: 24.23.140.185
Scan Target Address: 67.165.163.5
Scan Target Address: 24.254.241.95
Scan Target Address: 64.53.16.160
Most or All traffic to this addresses was from 10.0.87.50, It was from Port 888 to port 27005 which leads conclude, "Half-Life" gaming.

Scan Target Address: 213.106.160.152
Most traffic to this address was from 10.0.87.50, using source port 999 to destination port 1025. More gaming, this time it is Network Blackjack.

Selected Registration Information

Address: 212.179.35.118
Reason: Top most talker in the alert files

```
inetnum:     212.179.0.0 - 212.179.255.255
netname:     IL-ISDNNET-990517
descr:       PROVIDER
country:     IL
admin-c:     NP469-RIPE
tech-c:      TP1233-RIPE
tech-c:      ZV140-RIPE
tech-c:      ES4966-RIPE
status:      ALLOCATED PA
mnt-by:      RIPE-NCC-HM-MNT
changed:     hostmaster@ripe.net 19990517
changed:     hostmaster@ripe.net 20000406
changed:     hostmaster@ripe.net 20010402
source:      RIPE
route:       212.179.0.0/17
descr:       ISDN Net Ltd.
origin:      AS8551
notify:      hostmaster@isdn.net.il
mnt-by:      AS8551-MNT
changed:     hostmaster@isdn.net.il 19990610
source:      RIPE
```

person: Nati Pinko
address: Bezeq International
address: 40 Hashacham St.
address: Petach Tikvah Israel
phone: +972 3 9257761
e-mail: hostmaster@isdn.net.il
nic-hdl: NP469-RIPE
changed: registrar@ns.il 19990902
source: RIPE

Address: 147.46.59.144
Reason: Did large ICMP scan, as well as triggered FTP related alerts

Seoul National University (NET-SNU)
Computer Center
56-1 Shinrim-Dong Kwanak-Gu
Seoul 151-742
KR

Netname: NET-SNU
Netblock: 147.46.0.0 - 147.46.255.255

Coordinator:
Kim, Eunkyung (EK49-ARIN) egkim@ERCCW1.SNU.AC.KR
+82.2.880.5365

Domain System inverse mapping provided by:

ERCC.SNU.AC.KR 147.46.80.1
NS.KREN.NM.KR 147.47.1.1

Record last updated on 15-Jun-1995.
Database last updated on 16-Feb-2002 19:55:51 EDT.

Address: 207.96.37.198
Reason: Responsible for large Web scan

Erol's Internet Service (NETBLK-EROLS-CUST-951)
7921 Woodruff Court
Springfield, VA 22151
US

Netname: EROLS-CUST-951
Netblock: 207.96.37.0 - 207.96.37.255

Coordinator:
Erol's Internet Services (EROLS-NOC-ARIN) noc@RCN.COM
703-321-8000

Record last updated on 24-Feb-1998.
Database last updated on 16-Feb-2002 19:55:51 EDT.

Address: 61.150.5.19
Reason: Responsible for large Web scan

inetnum	61.150.0.0 - 61.150.31.255
netname	SNXIAN
descr	xi'an data branch,XIAN CITY SHAANXI PROVINCE
country	CN
admin-c	WWN1-AP, inverse
tech-c	WWN1-AP, inverse
mnt-by	MAINT-CHINANET-SHAANXI, inverse
mnt-lower	MAINT-CN-SNXIAN, inverse
changed	ipadm@public.xa.sn.cn 20010309
source	APNIC

person	WANG WEI NA, inverse
address	Xi Xin street 90# XIAN
country	CN
phone	+8629-724-1554
fax-no	+8629-324-4305
e-mail	xaipadm@public.xa.sn.cn, inverse
nic-hdl	WWN1-AP, inverse
mnt-by	MAINT-CN-SNXIAN, inverse
changed	wwn@public.xa.sn.cn 20001127
source	APNIC

Address: 24.0.28.234
Reason: Top OOS source address that was responsible for the Christmas Day Spike.

@Home Network (NETBLK-ATHOME)
450 Broadway Street
Redwood City, CA 94063
US

Netname: ATHOME
Netblock: 24.0.0.0 - 24.23.255.255
Maintainer: HOME

Coordinator:
Operations, Network (HOME-NOC-ARIN) noc-abuse@noc.home.net
(650) 556-5599

Domain System inverse mapping provided by:

NS1.HOME.NET	24.0.0.27
NS2.HOME.NET	24.2.0.27

ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE

Record last updated on 10-Apr-2000.
Database last updated on 16-Feb-2002 19:55:51 EDT.

Link Graph

For the link graph portion of the assignment, I selected to analyze traffic to and from the beetle! The beetle is a networked machine I found while doing the analysis that has the IP address, 10.0.70.69. The university's snort rules has a special snort signature monitoring traffic to and from this node, and the signature is called "beetle.ucs".

Defensive Recommendations

During the analysis process, I created a hotlist, which is the list of addresses that warrant investigation. Looking at the list we get an idea of the sort of defensive measures that the university should take.

212.179.35.118 and 10.0.70.70 are on the hotlist because of the being the top talkers. These exchanged a lot of Kazaa traffic. Kazaa is a multimedia file sharing and jukebox. The university should examine its policy on file sharing. Here we have proof that this kind of peer-to-peer traffic can consume a lot of network bandwidth.

10.0.100.165 is the CS Web server that is accessible from the outside. There were quite a lot of WEB alerts against this server. My analysis didn't reach too deeply into the web alerts, and I recommend looking at this server's web logs for further analysis. I also recommend that this server has the latest patches, and the OS is hardened. I'd also suggest that the file system where the HTML resides, be made read-only or be constantly refreshed, to avoid defacement attacks, I even heard of sites that serve their html off CD Rom Media. 10.0.253.114, 10.0.253.115 are also running web servers, and these recommendations apply to them as well.

We saw quite a few FTP related alerts in the data. My recommendation is to restrict incoming traffic to FTP. Disallow anonymous ftp, and keep a check one FTP server logs.

10.0.1.2, 10.0.1.3, 10.0.1.4, and 10.0.1.5 have DNS server running on them. Recommend a deeper look at the Zone transfer alerts. 10.0.1.2 needs to be looked at in conjunction with the "connect to 515 from inside" alert that came from it.

A very large proportion of the alerts we saw were triggered due to scans from the outside. These external scanners should be looked at and the owners sent warning notices. Sometimes the external scanning host is compromised the owners will be glad to correct the problem. If there is no response from the owners, determine the business use of allowing access, and accordingly block the addresses at the firewall.
Scans from the outside can also be stopped by configure border routers not to send ICMP Destination Unreachable (Host Unreachable). This will stop networking mapping attempts. Some scans came from the inside as well, and the recommendation is to review the university policy on scanner usage on campus.

Looking at the INFO MSN IM Chat data, we see quite a few internal hosts using the MSN chat software. Recommend reviewing university policy on chat. Also recommend sending advisory notices to the owners of these machines on the security implications of chat, and where to get the latest patches.

We saw some Worm, Trojan and Virus activity. The internal hosts that showed up on the hotlist for this should be checked for false positives and be disinfected if needed.

The scan data files proved to us that Internet Gaming is prevalent at the university. This doesn't seem to server any business case, and recommendation is to review the university policy on gaming.

Lastly, conduct vulnerability scans from a central location. Target the core services that the university provides and also the hosts that showed up on the hotlist.

Analysis Process

I took the following steps to do this audit for the university.

- Selected the 5 days over which I wanted to do the audit. I selected Christmas day to be within the time range, primarily because during the sans class, I learnt that "interesting events" usually occur on Christmas days.
- Downloaded the files, and the files were in compressed format. I couldn't decompress them on my windows machine. Took them over to a Linux box, where I still couldn't decompress them. Spent a couple of hours trying to figure out what was going on, and realized that they were already decompressed on the download. Doh!
- From reading past practicals, I knew that excel wouldn't cut it, so I decided on using a database. I choose Oracle, since I have easy access to one.
- My employer has a product in this industry, and I used their database schema, which is based on IDMEF. I only needed to populate a few of the many IDMEF fields.
- I overrode one of the fields to put in a marker to indicate which file the event came from (Alert/OOS/Scan), so that I could later filter my queries based on event source.
- Wrote a little program to parse the files, and convert "MY.NET" to "10.0".
- Wrote another program to stick the results into the database. Found it a bit slow, and a friend suggested using SQL loader.
- This turned out a much faster way to load the data. All I had to do was to essentially create a delimited file, a control file describing the fields, and feed them to the SQL loader program from oracle, and voila, I got all the data into the database in approximately half an hour.
- Most of the data analysis was done using raw sql queries against this database. Used Oracle's SQL*Plus program to issue the queries.
- For rapid results, I resorted to the good old grep, awk, sort , uniq commands from UNIX.

References

1. networksorcery.com. IP Protocol Information.
   URL: http://www.networksorcery.com/enp/topic/ipsuite.htm
2. SQL Tutorial .A gentle introduction to SQL.
   URL: http://www.dcs.napier.ac.uk/~andrew/sql/
3. George Koch, Kevin Loney. Oracle 8, The Complete Reference. Oracle Press.
   ISBN: 0-07-212364-8
4. NorthCutt, Steven et al. Intrusion Signatures and Analysis. New Riders
   Publishing. ISBN: 0-7357-1063-5
5. NorthCutt, Steven, Novak, Judy. Network Intrusion Detection. New Riders
   Publishing. ISBN: 0-7357-1008-2
6. Bell, Mike. GCIA Practical. January 2001.
   URL: http://www.sans.org/y2k/practical/Mike_Bell_GCIA.doc
7. Bidwell, Teri. mystery SF scan tool = Idlescan correlation. Nov. 13, 2000
   URL: http://www.securityfocus.com/archive/75/144723
8. Bidwell, Teri. GCIA Practical. October 2000
   URL: http://www.giac.org/practical/Teri_Bidwell_GCIA.doc
9. Shinberg, Scott. GCIA Practical. July 2001.
   URL: http://www.giac.org/practical/Scott_Shinberg_GCIA.doc
10. Bell, Anthony. Adore Worm – Another Mutation. April 6, 2001.
    URL: http://rr.sans.org/threats/mutation.php
11. Product Home Page. CyberKit. URL: http://www.cyberkit.net/.
12. Product FAQ. EDonkey. URL: http://www.edonkey2000.com/faq.html#port
13. Garreth, Jeremiah. GCIA Practical. July 2001.
    URL: http://www.giac.org/practical/Garreth_jeremiah_GCIA.zip

Last Updated: May 26th, 2017

Upcoming Training

Security Operations Center Summit & Training	**Washington, DC**	**Jun 05, 2017 - Jun 12, 2017**	**Live Event**
SANS Houston 2017	**Houston, TX**	**Jun 05, 2017 - Jun 10, 2017**	**Live Event**
SANS Columbia, MD 2017	**Columbia, MD**	**Jun 26, 2017 - Jul 01, 2017**	**Live Event**
SANS London July 2017	**London, United Kingdom**	**Jul 03, 2017 - Jul 08, 2017**	**Live Event**
SANSFIRE 2017	**Washington, DC**	**Jul 22, 2017 - Jul 29, 2017**	**Live Event**
SANSFIRE 2017 - SEC503: Intrusion Detection In-Depth	**Washington, DC**	**Jul 24, 2017 - Jul 29, 2017**	**vLive**
SANS San Antonio 2017	**San Antonio, TX**	**Aug 06, 2017 - Aug 11, 2017**	**Live Event**
SANS Boston 2017	**Boston, MA**	**Aug 07, 2017 - Aug 12, 2017**	**Live Event**
SANS Adelaide 2017	**Adelaide, Australia**	**Aug 21, 2017 - Aug 26, 2017**	**Live Event**
SANS Virginia Beach 2017	**Virginia Beach, VA**	**Aug 21, 2017 - Sep 01, 2017**	**Live Event**
SANS Network Security 2017	**Las Vegas, NV**	**Sep 10, 2017 - Sep 17, 2017**	**Live Event**
SANS vLive - SEC503: Intrusion Detection In-Depth	**SEC503 - 201709,**	**Sep 11, 2017 - Oct 18, 2017**	**vLive**
Baltimore September 2017 - SEC503: Intrusion Detection In-Depth	**Baltimore, MD**	**Sep 25, 2017 - Sep 30, 2017**	**vLive**
SANS Baltimore Fall 2017	**Baltimore, MD**	**Sep 25, 2017 - Sep 30, 2017**	**Live Event**
SANS London September 2017	**London, United Kingdom**	**Sep 25, 2017 - Sep 30, 2017**	**Live Event**
Community SANS Scottsdale SEC503	**Scottsdale, AZ**	**Oct 02, 2017 - Oct 07, 2017**	**Community SANS**
SANS October Singapore 2017	**Singapore, Singapore**	**Oct 09, 2017 - Oct 28, 2017**	**Live Event**
Community SANS Boston SEC503	**Boston, MA**	**Oct 09, 2017 - Oct 14, 2017**	**Community SANS**
Community SANS Ottawa SEC503	**Ottawa, ON**	**Oct 16, 2017 - Oct 21, 2017**	**Community SANS**
SANS Seattle 2017	**Seattle, WA**	**Oct 30, 2017 - Nov 04, 2017**	**Live Event**
SANS San Diego 2017	**San Diego, CA**	**Oct 30, 2017 - Nov 04, 2017**	**Live Event**
San Diego Fall 2017 - SEC503: Intrusion Detection In-Depth	**San Diego, CA**	**Oct 30, 2017 - Nov 04, 2017**	**vLive**
SIEM & Tactical Analytics Summit & Training	**Scottsdale, AZ**	**Nov 28, 2017 - Dec 05, 2017**	**Live Event**
SANS Cyber Defense Initiative 2017	**Washington, DC**	**Dec 12, 2017 - Dec 19, 2017**	**Live Event**
SANS OnDemand	**Online**	**Anytime**	**Self Paced**
SANS SelfStudy	**Books & MP3s Only**	**Anytime**	**Self Paced**