Global Information Assurance Certification Paper

GCIA Practical Assignment, Version 3.4

OLAP: A Tool for Recognizing Attack Trends

Cori Lynn Arnold

Submitted April 24, 2004

Abstract:

So you've got yourself some great, distributed, network data. What in the world are you going to do with it? Given a minimal amount of data, normalized from a variety of security devices you could "view" the attack data, looking for specific patterns that may reveal the nature of the attacks. So how do you view this data you ask? By placing each individual attack log into a database, then letting On-Line Analytical Processing (OLAP) take charge. You could for instance get a feel for what level of activity is being seen for a new vulnerability. You may be able to distinguish between worms, botnets, or basic scanning for vulnerable hosts by a small set of Internet hosts. Once the data is ready for your analysis you may ask how you can benefit from knowing the nature of the activity. Knowing that the activity is botnet in nature may assist you in justifying the budget for patching vulnerable systems. Imagine saying to your CIO, "5% of the activity we are currently seeing is attempting to overflow a service we run on our servers. Currently the activity looks like it's mostly coming from a botnet, but based on past vulnerabilities of this nature it may be a worm soon." This paper will briefly outline a technology used to visually trend attack data, first by briefly explaining OLAP, then by outlining the necessary components for OLAP to trend attack data, and finally by giving some examples of attack trends. The examples shown in this paper use attack data gathered by a large managed security services provider, with more than 500 clients world wide, and a large variety of security devices.

What is OLAP?

OLAP stands for On-Line Analytical Processing. OLAP uses a relational database as its backend, and requires extensive and time intensive setup before it can be processed. Once processed, the OLAP cubes are used to discover and investigate trends. OLAP is used most commonly in retail business situations to find trends in purchasing. Most OLAP technologies hold information on transactions that consumers do at shopping malls and grocery stores. Marketing teams use OLAP to discover the types of consumers that buy their products, in order to make better decisions on how to advertise. They have also been used to increase efficiency in factory settings. By analyzing the time taken and tasks involved in each step of manufacturing, a product a factory owner may make changes in the location of certain steps in the manufacturing process.

In terms of network trending, the OLAP technology may possibly provide you with enough network information to make business decisions during or prior to an emerging threat. You must provide your OLAP database with a relational database, that is a database with fact tables and supporting tables all linked together with common threads, full of your great, distributed, network data. For the sake of consistency, any specific examples given will be based on Microsoft SQL Server 2000 and Microsoft Analysis Services, but most topics discussed in this paper will be generic enough to apply to any OLAP technology. There are many options for OLAP technology, including the Seagate Analysis OLAP client discussed by Leong Ying Siong Clement in the SANS whitepaper "Log Analysis as an OLAP Application".

To begin the design of your data warehouse, a relational database that has a star schema, start with a fact table. Ideally the fact table contains a row for each log line taken from your multiple security devices. This is not always a feasible solution since the more records you have the more time it will take to process. You may have to aggregate your data as required by your environment. As a general rule you should not aggregate your data before placing it in the data warehouse, you will not be able to apply your analysis to some base data if that level of data has been aggregated to a higher level (Peterson et al, p. 100). A fact table row should only contain numerical data, with the possible exception of dates. A fact table with only numerical data will assist greatly in the speed of processing your OLAP cubes. For any data that is not numerical in nature, a key is placed in the row, which references a separate table that includes the key and data pair, each table is a point in your star schema (Peterson et al., p 75).

Once your data warehouse has been defined, begin to define every question you might ask the data warehouse. These questions will help you define your measures, and dimensions. Measures are the value of your OLAP cube that will be analyzed, and are usually numeric. I often use a "noun" in English as an analogous entity when explaining measures. Dimensions are the ways in which you describe your data, usually character data. Again the analogous entity for a dimension would be an "adjective" in English. In retail OLAP technologies the measure could be "number of customers" or "number of products". The examples of dimensions would be "hour of the day" or "customer neighborhood" or "product isle". Given these examples used in retail OLAP, the question "What hour of the day are the most products sold?" could be answered.

In our network example we will have much different measures and dimensions. Some sample questions may include, but are certainly not limited to:

1. How many external IP addresses per day are seen by my firewalls?

2. How many external IP addresses per day are from Istanbul?
3. How many probes for FTP are seen in a month?
4. How many IDS sensors are detecting the Internet Explorer URL parsing vulnerability?
5. How many distinct destination IP addresses are scanned on weekends?

Based on these questions you must include as a measure the number of security devices, the number of sources, the number of destinations and the number of logs. Dimensions needed for these questions include source IP address country, security device type, attack signature, month of the year, and day of the month. You may notice that as your questions get more in-depth your dimensions and measures will get more specific.

The Basics of Your Network Trending Data

The great, distributed, network data from a variety of security devices is going to have to have some minimal data. As described above you're going to need measures and dimensions. First, to introduce a novel idea in network trending you need to have country of origin data. The country of origin will only reflect the country of the host for the network traffic you are researching. You will likely not be able to trace an attack to the original attacking host. As explained by Ramneek Puri in "Bots & Botnet: An Overview" the attacker machine can be far removed from the hosts which are actually generating the network traffic you are detecting (p. 2). However, country data will help you find botnets and will definitely give you a feel for how distributed a botnet is. Country of origin data may be a bit hard to come by. You could subscribe to a service to lookup this information, which may be expensive depending on your budget. You could also use whois info, but due to the non-standardized method for displaying whois records, parsing the data may also be difficult. If you're truly desperate you could forgo the country of origin data and replace it with the source first octet or first two octets. Analyzing down to the individual IP level however, is not recommended, as OLAP needs to have something to aggregate.

Next, you are going to need time. Of course you need time to work on your database, designing your OLAP cubes and research, but in this case you're going to need timestamps of the activity. Timestamps are on every log for every security device; otherwise it is probably not a security device. Your data will need to be precise, and will be much more helpful to you if it all reflects the same time zone, although you could always update a field to normalize your data to the same time zone.

Finally for dimensions, you need to have a description of the activity you're seeing. This could be in the form of probes from various Firewall logs or signatures triggered from multiple Intrusion Detection Systems. If your data includes different vendors Intrusion Detection Systems, you will probably need

to map the signatures so that they are all pointing to the same attack activity. For instance, the Generic WebDAV/Source Disclosure "Translate: f" HTTP Header Request Attack may trigger "IIS 5 Translate: f Source Disclosure" on a Cisco IDS, and may trigger "HTTP_Translate_F_SourceRead" on an Internet Security System IDS. In your relational database for your OLAP cubes you can have an entry for Generic WebDAV/Source Disclosure "Translate: f" HTTP Header Request Attack. Then another table would have reference to that entry for both the Cisco IDS and ISS signatures as well as any other security device that can detect the Generic WebDAV/Source Disclosure "Translate: f" HTTP Header Request Attack.

Even though it is not recommended to do any aggregations on your data, I've found that log data from 500 distinct companies is far to overwhelming for OLAP to handled, especially if you encounter a situation where you have to make any changes to your OLAP cubes. The data shown in the examples have been aggregated for each unique pair of source IP and signature, for each combination a sum of logs is in another column of the table. This then drives the measures to be count of unique instances of source IP, the count of unique signatures and the sum of logs.

Attack Data Extras:

The basics above described what dimensions would be required for a minimal system to view attack data as will be presented in this paper. If you had the data you might want to have more dimensions in order to do more research. Take head! As you add to your dimensions and measures you will introduce more areas for error and increase the time it takes for each process of your cubes. Error is generally introduced when your OLAP technology is given two keys for the same piece of data in your fact table, or when you do not enter a key for a piece of data in your fact table.

Strange results also often occur if your data is not mutually exclusive. If for instance you had a signature hierarchy and example signatures such as:

Category	Subcategory	Signature
Web Attacks	Web Attack on Vulnerabilities	Code Red
Web Attacks	Worm Signatures	Code Red

If you are then counting distinct source IP's for all "Web Attacks" for April 3, 2004 you may get 5000 as your answer. However if you then looked at all distinct source IP's for "Web Attacks->Web Attack on Vulnerabilities->Code Red" you may get 3000, and all distinct source IP's for "Web Attacks->Worm Signatures->Code Red" will get you 3000. Users of your OLAP cubes will question how you can have 6000 by adding the two columns, but 5000 total. The answer is that the signature is really the same signature, but categorized different. The OLAP technology will count these instances twice. This is an

overly simplified example, but as much as possible you should keep your data mutually exclusive, since the problem will often occur and will be noticed by all the wrong people at unexpected and inopportune times.

If you are trending off of a small system, or you have a lot of processing power at your disposal, you may consider adding destination country, or destination host (for an extremely small system) to your dimensions. You may find that some network traffic is particularly prone to hitting your advertised web host, even though it is supposed to be random worm traffic.

On the other hand, if you are trending off of a quite large system you may want to include Industry as one of your dimensions. Industry analysis is very popular if you are secondarily using the OLAP trending to do any kind of marketing; Industry analysis can add to the gee-whiz factor when showing trends to your marketing, sales or executive staff. Even better, some of the techniques used to identify worm and botnet traffic through source IP country and region analysis may also be applied to industry analysis!

Finally, you may want to add a hierarchical dimension to describe your data source. The dimension can include the security device category, such as Firewall, IDS, or Router, and the security device vendor, such as Cisco, ISS or Symantec, and version, such as Cisco IDS 4, or ManHunt 3.1, and can also include a description of the security device's location on a network, such as intranet, DMZ or internet. Adding device type and vendor information may help you understand where the bulk of your information is coming from, and furthermore what may be missing.

For an extra measure you may want to include a unique identifier for the security device. This will help answer questions pertaining to how many sensors you are observing with particular activity, or whether or not you are seeing activity from one type of IDS and not another.

Viewing the Attack Data

With the Microsoft SQL server and Microsoft Analysis Services, a good (but not great) client front end will be Microsoft Excel Pivot Table Services. There are a few problems with using Excel as your front-end application. First, Excel is not pretty, nor that impressive when showing a demonstration of your trending ability to clients. The object box for selecting dimensions and measures is small, and difficult to maneuver if you have a lot of dimensions and measures. The drop down boxes for selecting particular dimensions within a hierarchy is somewhat difficult if you have a lot of choices within your measure. However, there is very little time associated with setting up the front-end application if you use Excel, whereas setting up your own front end may take lots of development time, and is not usually as adaptive addictions and

subtractions from your dimensions and measures as Excel pivot table services can be.

Now that all of the essential and extra measures have been defined you are going to need a dictionary of what to look for in a trend. The trends found below are by no means exhaustive. When I first began analyzing network traffic with OLAP technology I spent many hours scouring through the data, trolling for any obscure trend I might find. The result was extensive knowledge of what did not constitute a trend, and better methods for eliminating false positives. I'd compare the learning curve for trending analysis, with the learning curve used when a network administrator examines logs from a newly purchased IDS. The helper scripts written to automatically detect ebb and flows in network trends in order to figure out what trends should be more thoroughly investigated could be compared to the scripts written by a network administrator to alert him to a possible serious situation requiring more investigation.

Trending Example Botnets: Pre-Blaster 135/tcp

From the introduction, you are to imagine yourself describing to your CIO: "5% of the activity we are currently seeing is attempting to overflow a service we run on our servers. Currently the activity looks like it's mostly coming from a botnet, but based on past vulnerabilities of this nature it may be a worm soon." In this case you are going to need to be able to identify a botnet from your great, distributed, network data. The botnet is a network of Victim hosts all taking part in the same action at the same time. One of the cases presented in Puri's "Bots and Botnets" explains how botnets are used to create bigger botnets through scanning and exploiting a known vulnerability (p. 7,13). You may use your

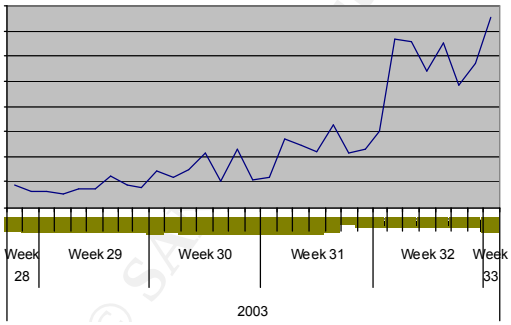


Figure 1: Unique sources probing for port 135/tcp by day

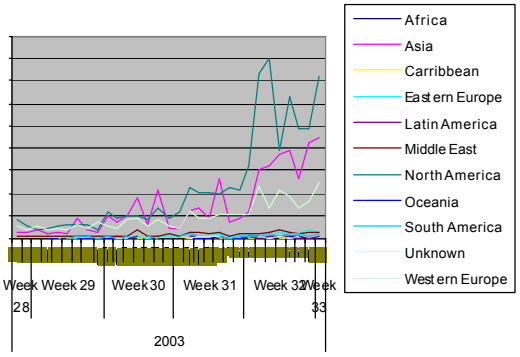


Figure 2: Unique sources probing for port 135/tcp by region by day

OLAP cubes to visually track the number of scans per day seen across your great, distributed, network data. The first example, Figure 1, shows a trend of hosts looking for port 135/tcp, which according to the Internet Assigned Numbers Authority is assigned to epmap or DCE endpoint resolution. Figures 1 and 2 display the activity seen after Microsoft Security Bulletin MS03-026, and details about the vulnerability found in the RPC DCOM interface, listening on port

135/tcp. Additionally, the graph's timeline is before Blaster, the well-known worm explained in detail on Symantec's Security Response web site. The graph on the left is a count of all hosts looking for the RPC DCOM by day; the graph on the right is broken down by country. Notice that on the right side the peaks are congruent amongst most of the regions with no noticeable period.

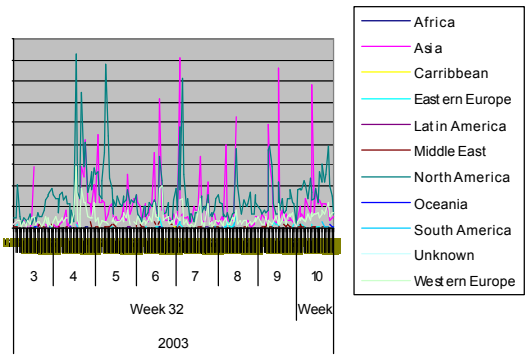


Figure 3: Unique sources probing for port 135/tcp by hour by region

Furthermore, looking at Figure 3, which is a detail of Figure 2, showing unique sources scanning for port 135/tcp for all of Week 32 and the first day of Week 33 by GMT hour, again we see congruent peaks across different geographical regions. These trends are most noticeable in the early hours of Day 7 and middle of Day 9. Activity with these characteristics is almost always a botnet, or in this case it could be multiple Botnets. The next section on worms will discuss the same port probes and trend seen after Blaster.

Trending Example Botnets: Sinit

As discussed by Puri, not all Bots rely on service vulnerabilities to propagate, some bots trick users into installing Trojan software (p. 7). Such is the case with Sinit, a Trojan with a peer-to-peer network distribution model. As discussed in the article by the LURHQ Threat Intelligence Group, Sinit is suspected to use an Internet Explorer browser exploit known as Java-ByteVerify (p. 1). As each new variant flies through the network your great, distributed, network data may log the activity seen as malformed DNS packets, since the peer-to-peer feature of this malicious code communicates over port 53/udp. Some intrusion detection systems have the ability to check protocol packets against RFC specifications. Specifically the ManHunt picks up Sinit peer-to-peer communication by detecting that the "DNS packets", while obeying all other rules for DNS, have entries in the length field of the question label section that are larger than the size limit. The label can be no longer than 63 bytes according to RFC 1035 (Mockapetris, p 10).

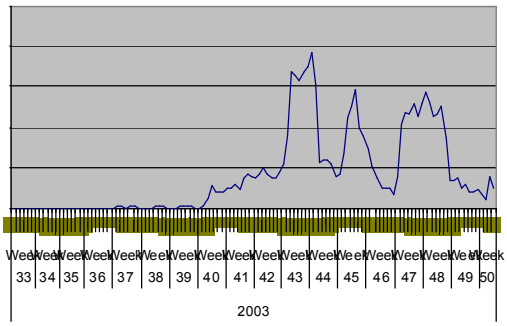


Figure 4: Unique sources triggering ManHunt's DNS_Bad_Label_Length by day

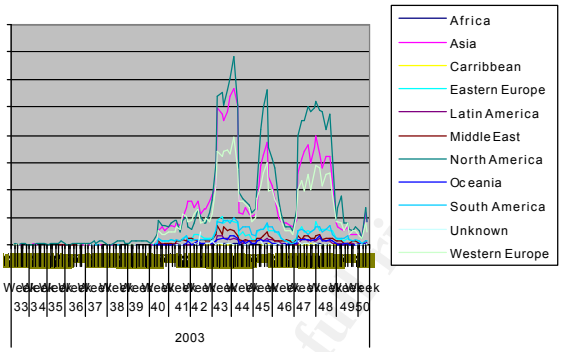


Figure 5: Unique sources triggering ManHunt's DNS_Bad_Label_Length by day and by region

Figures 4 and 5 show number of sources seen by ManHunt's signature DNS_Bad_Label_Length by day, over a period of 116 days. The graph on the right is a break down of the activity by geographical region of the source of the activity. Across all of the geographical regions the peaks and valleys are congruent in all but magnitude. The peaks and valleys seen may be contributed to new files being distributed across the Sinit infected hosts.

Often comparing your results to that of Isc.sans.org, an extremely distributed network of security devices, will also reveal a congruent graph, nearly

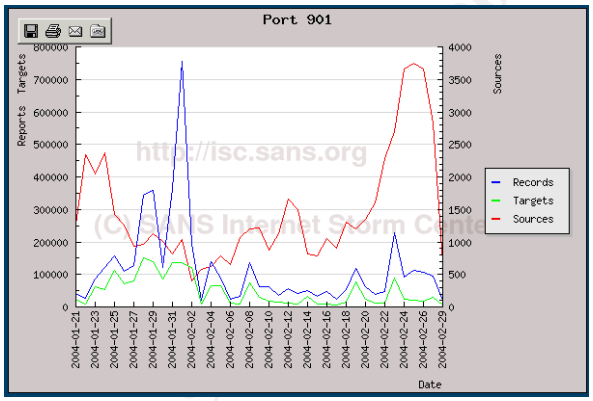


Figure 6: Counts of Records, Targets and Sources for port 901, taken from Isc.sans.org

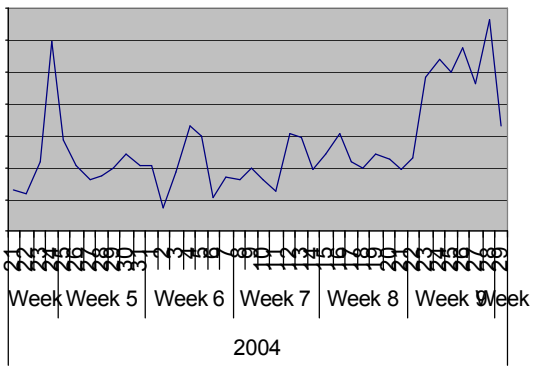


Figure 7: Unique sources probing for port 901/tcp by day

guaranteeing you are looking at the trends of a Botnet. One such example is shown in Figures 6 and 7, graphs of distinct sources probing for 901/tcp, which according to Iana.org belongs to smpnameres. However, on the same page you can click on "User Comments" and get an idea of what other analysts might suspect activity would be related to. Daniel Grim suspects the graphed activity

below to be botnets looking for Trojanized NetDevil hosts, as stated in the "User Comments" section for port 901 in the Internet Storm Center on Isc.sans.org (p. 1). The graph on the left is one generated by Isc.sans.org, by placing a port number in the dialog box. The number of distinct sources is usually drown out by the default view of the data, because records and targets usually double or triple that of the sources, so I recommend moving the sources to a separate axis, as I've done for the example here. Note that when comparing the red line in the graph on the left and the graph on the right, peaks are seen on the same days with similar magnitude.

Trending Example Worms: Blaster

On August 11, 2003 a sudden and extreme spike in probes for port 135/tcp was noticed. These probes were well distributed across the entire Internet, and carried with them an exploit for the RPC DCOM vulnerability. This activity is now well known as the Blaster Worm. Figure 8 shows the graph of unique sources probing for 135/tcp by day. The graph seen here is often called

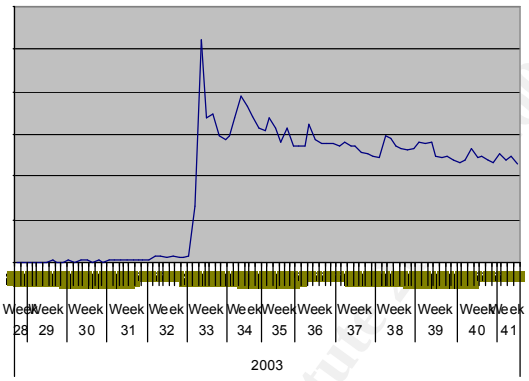


Figure 8: Unique sources probing for port 135/tcp by hour

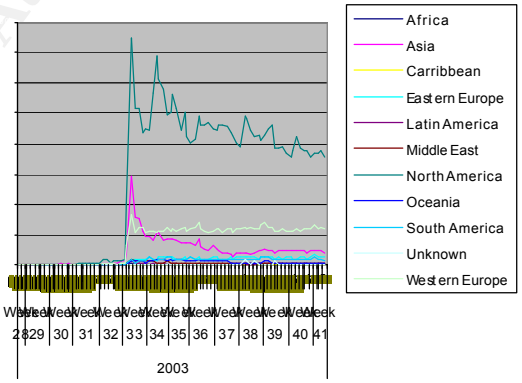


Figure 9: Unique sources probing for port 135/tcp by hour by region

a worm sigmoid, since it resembles a sigmoid "population growth curve that governs the growth of bacterial colonies and other populations of living organisms to a saturation level" (Gannis, p. 1).

One should notice that the activity seen in Figures 1 and 2 attributed to botnets probing for 135/tcp are hardly visible after the worm outbreak. Also of note is that once the initial peak of activity has occurred, the trend is for the number of infected hosts seen in a day to trickle down but remain a constant presence over time.

Figure 9 shows the same activity but the unique sources are broken down by region, what should be now a familiar method of analyzing the attack data. As we can see the worm activity seen for each region differs only in the magnitude, and a worm sigmoid can be seen for each region.

Trending Example Worms: Code Red.F

Code Red began its existence on July 16th, 2001 exploiting a vulnerability released on June 18th, 2001. The most unusual aspect of the worm is that it existed only in the memory of the infected machine, proving more difficult to detect for anti-virus vendors, but could be detected well by an IDS (Hayes, p. 1). The code its self contained a period where it stopped attempting to propagate and concentrated all of his efforts on a distributed denial of service attack against a hard-coded in the worm. This period was to occur between the 20th and the 28th of the month. Unfortunately, as noted by David Moore, some infected machines had incorrect dates on the first of each month the worm began to spread again (p. 1). Fortunately, this made for fabulous demonstrations of the OLAP technology when used to view attack data, since most other trends were not as cyclical. More than a year and a few variants later, the Code Red.F version was released on March 11, 2003. This version did not have the 20th to the 28th Denial of Service only period (Hurley, p. 1). The change in this behavior can be seen in the graph of unique sources attempting Buffer overflows in the IIS server's ISAPI extension below in Figure 10.

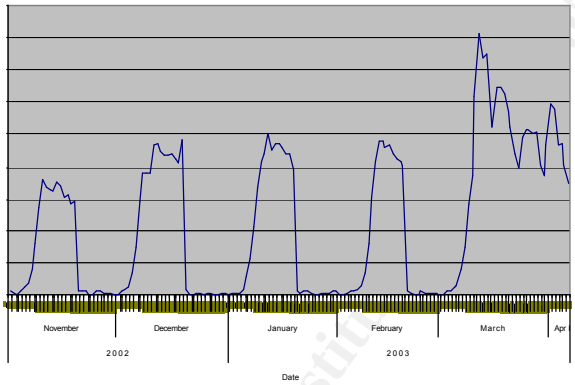


Figure 10: Code Red emerged March 11, 2003

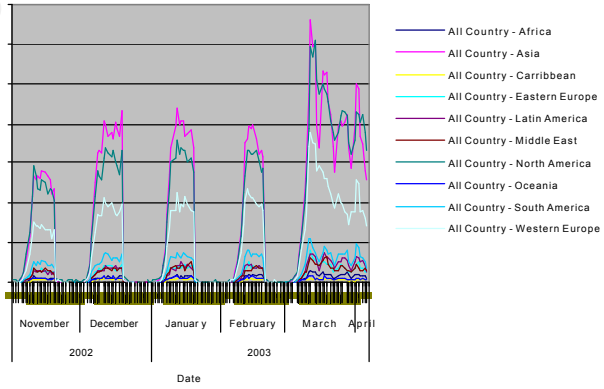


Figure 11: Unique Sources by Country of Origin, Code Red

In Figure 11 we again show this activity by country, and again see that all countries are congruent in the rise and fall of each month leading up to the release of Code Red.F on March 11, 2003. Having dropped the code killing the propagation threads, the Code Red.F trend alone looks like a worm sigmoid, and the activity continues to trickle on at a constant declining trend, until Blaster.

Trending Example Worms: Blaster Eats Code Red

So far we have concentrated on looking at one attack trend at a time, but a true differentiating factor for using OLAP as your attack trending tool is the ability to change your normal views of the attack data and find interesting, yet

hidden trends. If one were to keep track of the valleys as well as the peaks in attack data through constant querying of the data in an application outside of OLAP, one may have noticed a drop in Code Red traffic around the emergence of Blaster. Much like human diseases, where one disease can eradicate or severely hamper the effectiveness of another, one malicious code can have an effect on another piece of malicious code. A study in the journal "Nature" shows that in human pathogens "We see that one strain actually competes with the other for susceptible hosts." (Carlyle, p. 1) The same turns out to be true for network malicious code. Blaster affected Microsoft Windows machines, usually workstations, but the vulnerability applied to servers as well, and once the malicious code was installed, the machine was rebooted. Code Red propagated with an IIS server vulnerability, but only existed in the victim machine's memory. Therefore, when Blaster infected the Code Red victim's machine the Code Red infection was cleared. Blaster was then competing with Code Red for susceptible hosts. The figure below shows a steep decline in the number of unique sources attempting Buffer overflows in the IIS server's ISAPI extension, while at the same time a worm sigmoid of the number of probes for MSRPC. The graphs are shown on different axis since the number of unique sources seen with Blaster probes are nearly ten times that of the number of sources seen with Code Red probes.

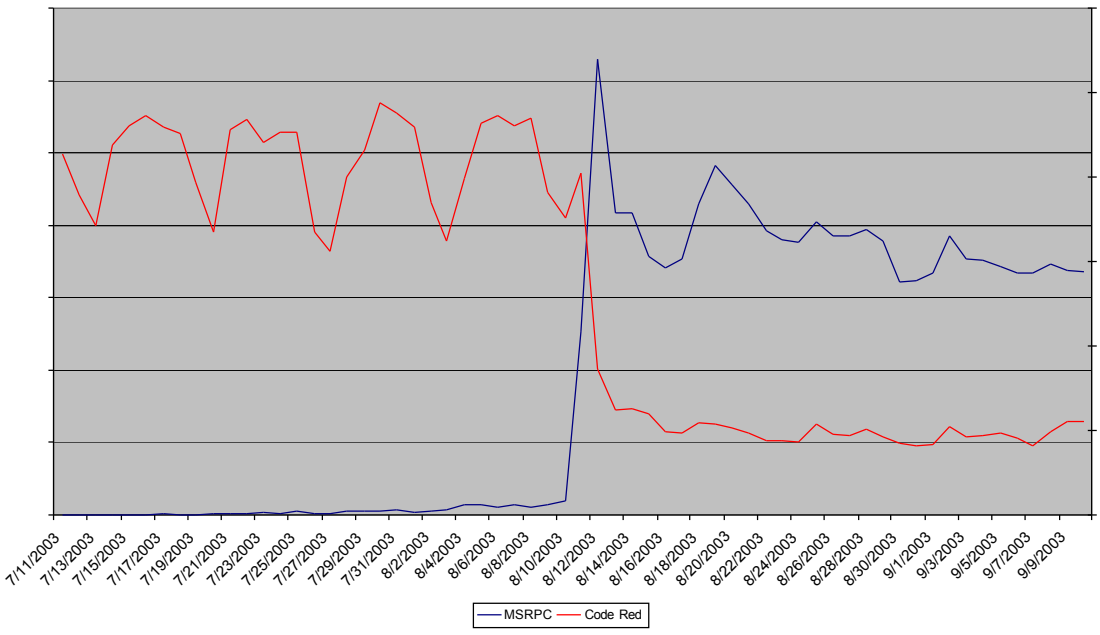


Figure 12: Unique Sources Code Red and Blaster

Conclusion

This paper has discussed a brief overview of OLAP, and outlined the necessary data for an OLAP technology you could develop in order to view

trends in attack data. There has been a presentation of a novel method for differentiating worm activity, and botnet scanning using source IP country of origin data. There have been examples of a botnet using the scan and exploit method of distributing it's malicious code, a botnet scanning for Trojanized hosts, a botnet participating in peer-to-peer communication, a classic worm sigmoid, and a worm variant. Furthermore, it has been shown that using OLAP you may be able to view the interaction of one piece of malicious code with another. With all of the examples we've seen that automated, well-distributed, malicious code will appear as many congruent lines when viewed by source country of origin over time.

References

1. Carlyle, Kimberly. "UGA SCIENTISTS SHOW ECOLOGICAL INTERACTION BETWEEN 'COMPETING' PATHOGENS." UGA News April 2003 URL: http://www.uga.edu/news
2. Gannis, Mike. "The Worm Returns: CAIDA Researchers Track New Infestations of 'Code Red'." National Partnership for Advanced Computational Infrastructure Online August 2001 URL: http://www.npaci.edu/online/v5.16/wormreturns.html
3. Grim, Daniel. "User Comment – Port 901." [No date] URL: http://isc.sans.org/show_comment.html?id=451
4. Hayes, Bill. "The Year of the Worm" SecurityFocus: INFOCUS August 2001 URL: http://www.securityfocus.com/infocus/1291
5. Mockapetris, P. "Domain Names - Implementation and Specification." November 1987 URL: ftp://ftp.rfc-editor.org/in-notes/rfc1035.txt
6. Moore, David et al. "Hot on the Trail of 'Code Red' Worms" enVision Vol.17 No. 3, September 2001 URL: http://www.npaci.edu/envision/v17.3/worms.html
7. Microsoft. "Microsoft Security Bulletin MS03-026: Buffer Overrun In RPC Interface Could Allow Code Execution." July 16, 2003 URL: http://www.microsoft.com/technet/security/bulletin/MS03-026.asp
8. Peterson, Tim et al. Microsoft OLAP Unleashed Second Edition. Indianapolis: Sams, 2000
9. Puri, Ramneek. "Bots & Botnet: An Overview." August 2003 URL: http://www.sans.org/rr/papers/index.php?id=1299
10. Siong, Leong Ying. "Log Analysis as an OLAP Application: A Cube to Rule Them All." June 2003 URL: http://www.sans.org/rr/papers/33/1152.pdf
11. Symantec. "W32.Blaster.Worm" August 17, 2003 URL: http://securityresponse.symantec.com/avcenter/venc/data/w32.blaster.worm.html
12. Symantec "Backdoor.NetDevil" February 13, 2002 URL: http://securityresponse.symantec.com/avcenter/venc/data/backdoor.netdevil.html

Part 2 – Network Detects

Network Detect 1: Sinit Infected Hosts

ManHunt IDS v 3.1 Logs
date|time|signature|SourceIPAddress:sourceport|DestinationIPAddress:destinationport| Payload of packet;

2/29/2004|20:39:22|RCRS/DNS_BAD_LABEL_LENGTH|80.72.160.51:53|141.22.169.254:53|01 02 00 07 d1 86 3f c3 26 14 01 01 **44** 99 86 dc 35 00;

2/29/2004|20:39:27|RCRS/DNS_BAD_LABEL_LENGTH|80.72.160.51:53|212.123.120.131:53|01 02 00 07 d1 86 3f c3 26 14 01 01 **44** 99 86 dc 35 00;

2/29/2004|20:39:27|RCRS/DNS_BAD_LABEL_LENGTH|80.72.160.51:53|212.123.120.131:53|01 02 00 07 d1 86 3f c3 26 14 01 02 **d4** 7b 78 83 35 00 81 13 a0 52 35 00;

2/29/2004|20:39:38|RCRS/DNS_BAD_LABEL_LENGTH|80.72.160.51:53|24.158.159.145:53|01 02 00 07 d1 86 3f c3 26 14 01 02 **c4** 28 17 b2 35 00 43 78 0c 51 35 00;

2/29/2004|20:39:43|RCRS/DNS_BAD_LABEL_LENGTH|80.72.160.51:53|196.40.23.178:53|01 02 00 07 d1 86 3f c3 26 14 01 02 **c4** 28 17 b2 35 00 43 78 0c 51 35 00;

2/29/2004|20:40:26|RCRS/DNS_BAD_LABEL_LENGTH|80.72.160.51:53|133.54.226.54:53|01 02 00 07 d1 86 3f c3 26 14 01 02 **d4** 7b 78 83 35 00 81 13 a0 52 35 00;

2/29/2004|20:40:31|RCRS/DNS_BAD_LABEL_LENGTH|80.72.160.51:53|63.123.164.225:53|01 02 00 07 d1 86 3f c3 26 14 01 02 **d4** 7b 78 83 35 00 81 13 a0 52 35 00;

2/29/2004|20:40:36|RCRS/DNS_BAD_LABEL_LENGTH|80.72.160.51:53|157.112.44.96:53|01 02 00 07 d1 86 3f c3 26 14 01 02 **d4** 7b 78 83 35 00 81 13 a0 52 35 00;

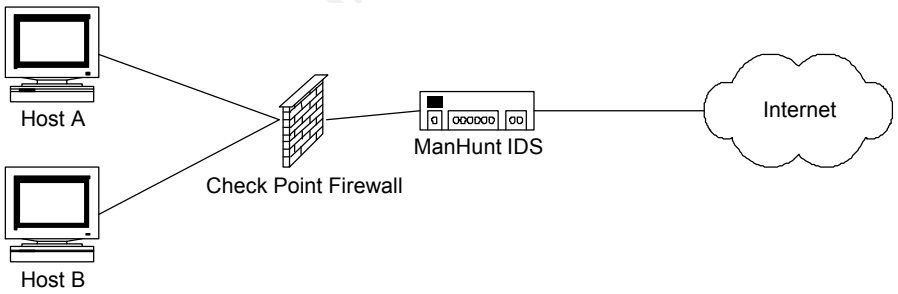
Check Point NG Firewall Logs
date;time;ruleaction;sourceipaddress;sourceport;destinationipaddress;destinationport;protocol;gatewayipaddress

29Feb2004;20:39:24;accept;HostA.some.com;53;144.84.73.130;53;udp;FW.so

me.com
29Feb2004;20:39:25;accept;HostA.some.com;53;62.47.69.5;53;udp;FW.some.com
29Feb2004;20:39:25;accept;HostB.some.com;53;180.108.231.5;53;udp;FW.some.com
29Feb2004;20:39:29;accept;HostB.some.com;53;186.189.67.115;53;udp;FW.some.com
29Feb2004;20:39:39;accept;HostB.some.com;53;216.2.243.253;53;udp;FW.some.com
29Feb2004;20:39:40;accept;HostA.some.com;53;80.55.13.162;53;udp;FW.some.com
29Feb2004;20:39:40;accept;HostA.some.com;53;141.85.31.12;53;udp;FW.some.com
29Feb2004;20:39:41;accept;HostA.some.com;53;151.92.121.120;53;udp;FW.some.com
29Feb2004;20:39:45;accept;HostA.some.com;53;164.199.222.149;53;udp;FW.some.com
….
29Feb2004;21:01:15;accept;HostB.some.com;53;0.72.34.173;53;udp;FW.some.com

Source of Trace:

The source of the trace was a client network. The ManHunt is placed on the dirty side of the network, as shown in the graph below.



Detect was generated by:

ManHunt IDS v 3.1

The signature DNS_BAD_LABEL_LENGTH triggered on DNS packets that have large entries in the length field of the question label section. The label can be no longer than 63 bytes according to RFC 1035. The length has been bolded in each payload above and ranges from 68 to 212.

CheckPoint FW NG Logs

DNS traffic is allowed from any host on the network, a rule that is on by default on this type of firewall.

Probability the source address was spoofed:

Low; the hosts are client workstations.

Description of the Attack:

HostA.some.com and HostB.some.com in the logs above are seen connecting to multiple hosts over port 53/udp over a period of twenty-two minutes. All port 53/udp connections are made going out of the firewall with the FW.some.com address. The ManHunt is detecting DNS_BAD_LABEL_LENGTH. The activity seen here is the peer-to-peer communication of the known Trojan Sinit. The hosts have been infected for some time and are attempting to communicate to other Trojanized hosts over port 53/udp, a known element of Sinit. The hosts were likely infected upon viewing a Trojanized web site, a known infection vector for Sinit {insert reference here}.

Attack mechanism:

The two hosts in question were detected while looking for the source of the DNS_BAD_LABEL_LENGTH signature triggered on the dirty side of the network. The hosts created a total of more than 25,000 port 53 checkpoint logs on the day shown here. Many of the logs are going to destinations that could not possibly be DNS, such as the last CheckPoint log in the example where an attempt was made to connect to 0.72.34.173. Since the Trojanized host attempting to connect to an internet black hole, there is probably a flaw in the Trojan's randomization code.

Correlations:

The LURHQ posting, mentioned above has more detail on the break down of the Sinit Trojan, and its communication methods. To my knowledge, the first activity was noted on George Bakos' web site at http://people.ists.dartmouth.edu/~gbakos/bindsweep/

Evidence of active targeting:

There is no evidence of active targeting. The hosts are participating in randomized peer-to-peer Trojan activity to thousands of hosts on the internet.

Severity:

Criticality = 0

The hosts in question are workstations on the client network.

Lethality = 5

The original attack succeeded, we are seeing the result of further communication by the Trojan.

System Countermeasures = unknown, but likely 1

Neither host is likely running a current version of Anti-Virus software, which would likely pick the activity up.

Network Countermeasures = 1

The hosts are allowed to continue looking for more Sinit Trojans to exchange files with, and were likely allowed to get infected through a peer-to-peer exchange in the first place. The firewall has poor egress filtering applied and allows port 53/udp traffic to pass through the firewall from any host on the network without inspection.

Severity = (0 + 5) - (1 + 1) = 3

Defensive Recommendation:

All workstations on the network should have Anti-Virus software on the systems, updated regularly. Sinit is a well-known Trojan, more than six months old, and all major vendors of Anti-Virus software have definitions for this Trojan. Users on this network need to be trained how to use and update their workstation for new Anti-Virus definitions.

Default settings on all firewalls should be inspected for security problems, in this case port 53/udp is allowed to pass through the firewall without inspection by default. Egress filtering needs to be applied to firewalls to prevent further activity after the initial infection has occurred. Egress filtering may also prevent the initial download completion. It is possible that the writer of such a Trojan could take over control of the workstation and act as the user, possibly disclosing corporate information, or worse, just through the port 53/udp connections that are allowed to pass through the firewall with out inspection.

Multiple choice question:

What is wrong with the following DNS *payload*?

0102 0007 d186 3fc3 2614 0101 4499 86dc
3500

A) There are no questions.
B) The question label is greater than 63 bytes
C) The names are greater than 255 bytes
D) The question label is less than 63 bytes

Answer: B

Network Detect 2: Phishing

Enterasys Dragon 5 Log

Datetime|Signature|sourceipaddress|destinationipaddress|sourceport|destinationport|protocolnumber|

2004-03-01 02:17:24|IE:URI-OBFUSCATION2|24.159.188.86|mail.good.com|2691|25|6|

Session Data (using the mksession application on Dragon)

g: 8bit{D}{A}
{D}{A}
< html > {D}{A}
< head > < /head > {D}{A}
< body > {D}{A}
Dear Westpac Internet Banking Customer! < br > < br > {D}{A}
This email was sent by the Westpac server to verify < br > {D}{A}
your e-mail address. You must complete this process < br > {D}{A}
by clicking on the link below and submiting Westpac < br > {D}{A}
secure verification form which appears in your browser < br > {D}{A}
< br > {D}{A}
This is done for your protection --- because some of < br > {D}{A}
our members no longer have access to their email addresses < br > {D}{A}
and we must verify it. < br > {D}{A}
< br > {D}{A}
Please use this unique link to the Westpac verification form < br > {D}{A}
to verify your e-mail: < a
href="http://olb.westpac.com.au%01%01%01%01%01%01%01%01%01%01%0
1%01%01%01%01%01%01%01%01%01%01%
01%01%01%01%01%01%01%01%01%01%01%01%01%01%01%01%01%01%01%01%01%
01%01%01%01%01%01%01%01%01%01%01%01%01%01%01%01%01%01%01%01%01%
01%01%01%01%0
1%01%01%01%01%01%01%01%01%01%01%01%01%01%01%01%01%01%01%01%01%0
1%01%01%01%01%01%01%01%01%01@210.15.78.10/img/.w/westpac.html" >
https:/
/olb.westpac.com.au?EmailID=ksjfh86fgHGSDG > < /a > < br > < br > {D}{A}
This message is digitally signed by Westpac server.{D}{A}

Source of Trace:

The source of the trace is a client network.

Detect was generated by:

Enterasys Dragon 5 Intrusion Detection System

The signature, IE:URI-OBFUSCATION2, triggered while looking for the string “http” followed by a URL with the string “%01@” in it.

Probability the source address was spoofed:

Low, the mail session needed to complete the three-way TCP handshake to send this mail message. The source IP address is part of a dial-up ISP and is probably being used as a spam mailer, as it is on at least one spam sender list: http://www.albury.net.au/cgi-bin/deroute

Description of the attack:

This attack targets users with accounts at Westpac bank to go to an illegitimate web site and enter account information as though the user was really logged into https://olb.westpac.com. The attack uses Internet Explorer URL parsing vulnerability by placing a "%01" character before an @ symbol in the URL, thereby hiding the real web site address that the user would connect to, detailed in CAN-2003-1025. The attack also uses social engineering through statements like "This is done for your protection" & "This message is digitally signed by Westpac server." to make the user feel more comfortable about responding to the email.

Attack mechanism:

This email was likely sent out with the intent of getting to the most email addresses, and seeing if anyone would bite. The day of the incident the web site was online, but two days after this detect the web site at 210.15.78.10 was no longer available. The attacker most likely was detected and kicked off of a compromised host, or moved off to prevent being detected. However good the social engineering portions of the email are, as stated above, the email it's self contains a spelling error, "submiting", which is a rarity for Bank generated mail.

Correlations:

These detects are chronicled by the Anti-Phishing group at http://www.antiphishing.org, and currently include two Westpac Bank phishing examples, only one of which uses the Internet Explorer URL parsing vulnerability.

Evidence of active targeting:

None, phishing emails tend to be sent out like spam. This email was likely sent out with the intent of getting to the most email addresses, and seeing if anyone would bite.

Severity:

Criticality: 0 This is not a system compromising attack, but rather an attack on a user's identity or financial information.

Lethality: 0 Again, this attack is not considered a system compromising attack.

System Countermeasures: 2 It is unknown if the host in question has been patched for the Internet Explorer URL parsing vulnerability, but the patch has been available for one month and this client does keep up with host patches.

Network Countermeasures: 1 Unknown, but although there is an IDS in place to detect emails attempting to use the Internet Explorer URL parsing vulnerability, it is not likely mail filters in place to block the user from getting emails with the vulnerability in it.

Severity = (0 + 0) -(2 + 1) = -3

Defensive recommendation:

First, all versions of Internet explorer should be patched to include the Cumulative Security Update for Internet Explorer (832894), detailed in Microsoft Security Bulletin MS04-004 at http://www.microsoft.com/technet/security/bulletin/ms04-004.asp.

A proxy firewall may be able to prevent users from logging into a web site that attempts to exploit the Internet Explorer URL parsing vulnerability.

Educating users on how to verify they are on a secure web site, and how to view a site's certificate may gain further defense. This phishing scheme could be easily converted into a more corporate information-gathering scheme; a phishing scheme that could fake a corporate intranet web site could phish for network username or passwords could be extremely dangerous and lead to internal host compromises.

Multiple choice question:

What techniques are used for phishing attacks?

A) Fake web sites that are branded to look like real banks or retailers
B) Internet Explorer URL parsing vulnerability in emailed web links
C) Social Engineering
D) All of the above

Answer: D All of the above.

Detect 3 – Load Balancing (posted on Intrusions List 3/18/2004)

Logs using windump:

```
08:00:55.074488 IP (tos 0x0, ttl 45, id 60256, len 40) 61.221.200.2.80 >
226.185.36.234.80: . ack 0 win 1400
08:01:00.564488 IP (tos 0x0, ttl 45, id 60659, len 40) 61.221.200.2.80 >
226.185.36.234.80: . ack 1 win 1400
08:01:03.874488 IP (tos 0x0, ttl 45, id 61047, len 40) 61.221.200.18.80 >
226.185.36.234.80: . ack 0 win 1400
08:01:08.874488 IP (tos 0x0, ttl 45, id 61451, len 40) 61.221.200.18.80 >
226.185.36.234.80: . ack 1 win 1400
08:01:13.874488 IP (tos 0x0, ttl 45, id 61852, len 40) 61.221.200.34.80 >
226.185.36.234.80: . ack 0 win 1400
08:01:18.874488 IP (tos 0x0, ttl 45, id 62275, len 40) 61.221.200.34.80 >
226.185.36.234.80: . ack 1 win 1400
08:01:23.874488 IP (tos 0x0, ttl 45, id 62709, len 40) 61.221.200.50.80 >
226.185.36.234.80: . ack 0 win 1400
08:01:28.874488 IP (tos 0x0, ttl 45, id 63133, len 40) 61.221.200.50.80 >
226.185.36.234.80: . ack 1 win 1400
08:01:33.874488 IP (tos 0x0, ttl 45, id 63544, len 40) 61.221.200.66.80 >
226.185.36.234.80: . ack 0 win 1400
08:01:38.874488 IP (tos 0x0, ttl 45, id 63948, len 40) 61.221.200.66.80 >
```

226.185.36.234.80: . ack 1 win 1400
08:01:44.464488 IP (tos 0x0, ttl 45, id 64359, len 40) 61.221.200.82.80 >
226.185.36.234.80: . ack 0 win 1400
08:01:49.234488 IP (tos 0x0, ttl 45, id 64750, len 40) 61.221.200.82.80 >
226.185.36.234.80: . ack 1 win 1400
08:01:53.874488 IP (tos 0x0, ttl 45, id 65148, len 40) 61.221.200.98.80 >
226.185.36.234.80: . ack 0 win 1400
08:01:58.884488 IP (tos 0x0, ttl 45, id 17, len 40) 61.221.200.98.80 >
226.185.36.234.80: . ack 1 win 1400
08:02:03.874488 IP (tos 0x0, ttl 45, id 414, len 40) 61.221.200.114.80 >
226.185.36.234.80: . ack 0 win 1400
08:02:08.874488 IP (tos 0x0, ttl 45, id 838, len 40) 61.221.200.114.80 >
226.185.36.234.80: . ack 1 win 1400
08:16:41.644488 IP (tos 0x0, ttl 44, id 58230, len 40) 61.221.99.242.80 >
226.185.22.232.80: . ack 0 win 1400
08:16:46.604488 IP (tos 0x0, ttl 44, id 58513, len 40) 61.221.99.242.80 >
226.185.22.232.80: . ack 1 win 1400
08:16:51.554488 IP (tos 0x0, ttl 47, id 58780, len 40) 163.22.229.253.80 >
226.185.22.232.80: . ack 0 win 1400
08:16:56.554488 IP (tos 0x0, ttl 47, id 59042, len 40) 163.22.229.253.80 >
226.185.22.232.80: . ack 1 win 1400
08:17:01.574488 IP (tos 0x0, ttl 44, id 59329, len 40) 203.69.227.10.80 >
226.185.22.232.80: . ack 0 win 1400
08:17:06.584488 IP (tos 0x0, ttl 44, id 59606, len 40) 203.69.227.10.80 >
226.185.22.232.80: . ack 1 win 1400

Source of Trace:
The source of the trace is a log file: 2002.5.2 posted to http://www.incidents.org/logs/Raw/. All logs have been obfuscated to remove any references to the protected networks, and the checksums were altered for the truly clever, as stated in the README file in the same directory. WinDump was used to analyze this detect. The “Bad cksum” messages were removed since the destination IP addresses were obfuscated for the purpose of this exercise.

Little to no information about the network is known.

Detect was generated by:

According to the README file for the logs, the log files were detected by a Snort running in binary logging mode, and only logs that violated the Snort rule set were included.

By running tcpreplay against another snort alerts such as the one below are triggered by the ACK number = 0 in the packets shown above.

[**] [1:628:3] SCAN nmap TCP [**]
[Classification: Attempted Information Leak] [Priority: 2]

03/16-15:20:06.221703 61.221.200.2:80 -> 226.185.36.234:80
TCP TTL:45 TOS:0x0 ID:60256 IpLen:20 DgmLen:40
A* Seq: 0x1E2 Ack: 0x0 Win: 0x578 TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS28]

Probability the source address was spoofed:

The probability the source address was spoofed is High. Each packet is nearly exactly five seconds apart, a little too exact to be from different hosts. The protocol is TCP, which usually needs a three-way handshake to complete, but these are likely lone ACK packets with zero for sequence numbers, indicating that the three-way handshake probably did not actually happen.

Description of the attack:

The incident was immediately highlighted because of the regularly interval (5 seconds apart) ACK packets from several source hosts 16 IP addresses higher each time, all with source and destination port 80. Such a coordinated methodology would be a sure sign of pre-reconnaissance, with a follow on attack likely.
Isolating the most interesting patterns of this attack we get:
The source and destination ports are 80/tcp for all packets.
The window sizes are all 1400.
The first eight hosts are all 16 IP addresses apart.
The IP identification number is approximately 400 apart from the packet before for each packet (if you add 65535 to the last three packets, since the id number starts again at 17 for the third to last packet).
The time to live entries are all very close to 45, if not 45 in most cases.

Attack mechanism:

Upon further research it has been found that TCP ACK packets are commonly used in to find hosts that are up on the network. Nmap uses TCP ACK packets in parallel with ICMP echo request packets, since some sites block echo request packets, according to the nmap documentation at:
http://www.insecure.org/nmap/data/nmap_manpage.html
After careful inspection and correlations it was determined that this is likely not an attack, but a load balancing query. The source of the activity is looking for the best ISP route for traffic by sending several TCP ACK packets from different sources, under the assumption that the sources may use different routes, and measuring the time delay of the destination's TCP RST packet will reveal the best ISP.

Correlations:

This detect was documented in Incidents.org Intrusions mailing list on April 26, 2002 by Chris Brenton with the following URL:
http://www.incidents.org/archives/intrusions/msg08119.html

The network traffic detect is also noted with very similar traffic in Loic

Juillard's Practical Detect at: http://cert.uni-stuttgart.de/archive/intrusions/2003/08/msg00125.html
The general traffic description generated by Radware Linkproof application is found in the granted patent application number ***6,665,702***:
"A TCP ACK may be sent to the client's source IP address and port. If the client's request was via a UDP connection, a TCP ACK to the client's source IP address and port 80 may be used. One or both TCP ACK messages should bypass any intervening NAT or firewall and cause client 26 to send a TCP RST message, which may be used to determine both latency and TTL. While TTL does not necessarily indicate the number of hops from the client to the load balancer, comparing TTL values from LB1, LB2, and LB3 should indicate whether it took relatively more or less hops."
The full patent grant can be found at http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PALL&p=1&u=/netahtml/srchnum.htm&r=1&f=G&l=50&s1=6,665,702.WKU.&OS=PN/6,665,702&RS=PN/6,665,702.

Evidence of active targeting:

The network detect was generated by a load balancing application, and therefore is not actively targeting the destination host.

Severity:

Criticality 3, Nothing is known about the destination host in this case. The criticality score is given a medium/high rating since the host may be a DNS or Web server.

Lethality 1, Had this been a reconnaissance scan (and not the Linkproof application) nothing malicious would result in an nmap ACK scan.

System countermeasures 1, Since nothing is known about the actual countermeasures but there isn't much for the system to be protected from a load-balancing server, the rate was given a one.

Network countermeasures 1, Since nothing is known about the actual countermeasures, but a simple stateful firewall could drop unestablished TCP connections, which is likely in place here the rate was given a one.

Severity = (3 + 1) – (1 + 1) = 2

Defensive recommendation:

To avoid reconnaissance by nmap a stateful firewall may be used to limit the returned traffic from unestablished TCP connections.

Multiple choice test question:

Given the packet:

```
0000  00 00 0c 04 b2 33 00 03 e3 d9 26 c0 08 00 45 00   .....3....&...E.
0010  00 28 ec f3 00 00 2d 06 05 c9 3d dd c8 02 e2 b9   .(....-...=.....
0020  24 ea 00 50 00 50 00 00 02 44 00 00 00 00 50 10   $..P.P...D....P.
```

0030 05 78 0c 65 00 00 00 00 00 00 00 00 .x.e........

Which of the below Snort Signatures would fire?

A. alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN nmap XMAS"; stateless; flags:FPU,12; reference:arachnids,30; classtype:attempted-recon; sid:1228; rev:3;)
B. alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN nmap fingerprint attempt"; stateless; flags:SFPU; reference:arachnids,05; classtype:attempted-recon; sid:629; rev:2;)
C. alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN nmap TCP"; stateless; flags:A,12; ack:0; reference:arachnids,28; classtype:attempted-recon; sid:628; rev:3;)
D. alert tcp $HOME_NET 80 -> $EXTERNAL_NET any (msg:"BACKDOOR BackOrifice access"; flags: A+; content: "server|3a| BO|2f|"; reference:arachnids,400; sid:112; classtype:misc-activity; rev:3;)

Answer: C

Response from Intrustions List by Oliver Viitamaki:

Hi,
A good analysis, although I think that you need to reexamine your assertion that this is a load balancer. If you perform a Whois lookup of the address block 61.221.200.x that may provide more insight. If you still believe this is a Load Balancer, please provide more information, how this large group of source addresses, can be viewed that way. Please feel free to enlighten me as well.

ov

My reply:
My basic assumption is that the sources have too similar characteristics and are too evenly spaced to be from the same host, and there fore must be spoofed, or in this case “generated” by a best route system called Linkproof.

Part 3: Analyze This!

Executive Summary

The following is a detailed analysis for preformed for Unknown Atlantic University (UAU) for the period of 3/3/2004 to 3/7/2004. While the logs analyzed were quite large, more than 300 MB, nearly 4.7 Million events, most of the activity was found to be benign and or redundant, and caused by normal use, and possible misuse, within the University.

There were a few instances of presumed infected hosts on the network. These hosts were found to be exhibiting suspicious scanning activity for services known to carry worms. The scanning seen by these hosts commenced at a rate higher than can be commanded by a user. The technique used for discovering these hosts can be found in the technical section. Based on the activity seen from the hosts involved in these activities, the compromised machines were assumed to be user maintained machines, and not mission critical servers.

Presumed Infected Hosts

Internal Host	Possible Worm Type	Port Scanning For
MY.NET.80.224	RPCDCOM Overflow / Blaster or Welchia	135/tcp
MY.NET.81.39	RPCDCOM Overflow / Blaster or Welchia	135/tcp
MY.NET.70.37	File Print Sharing / Opaserv	137/udp
MY.NET.97.147	Unknown	80/tcp

Excessive peer-to-peer usage on networks such as Gnutella, EDonkey2000, and WinMX, was also found being used by several internal hosts. The hosts involved in these activities were also assumed to be user maintained machines and not mission critical servers, based on the lack of any other activity but the P2P.

Overall the malicious code and bandwidth usage activity seen on the network is much less than would be expected from a largely autonomous network, with the greater percentage of student maintained machines, such as a University. The lack of any seriously questionable activity can only attest to the Universities hard working and security conscious Information Technology group.

Files Analyzed

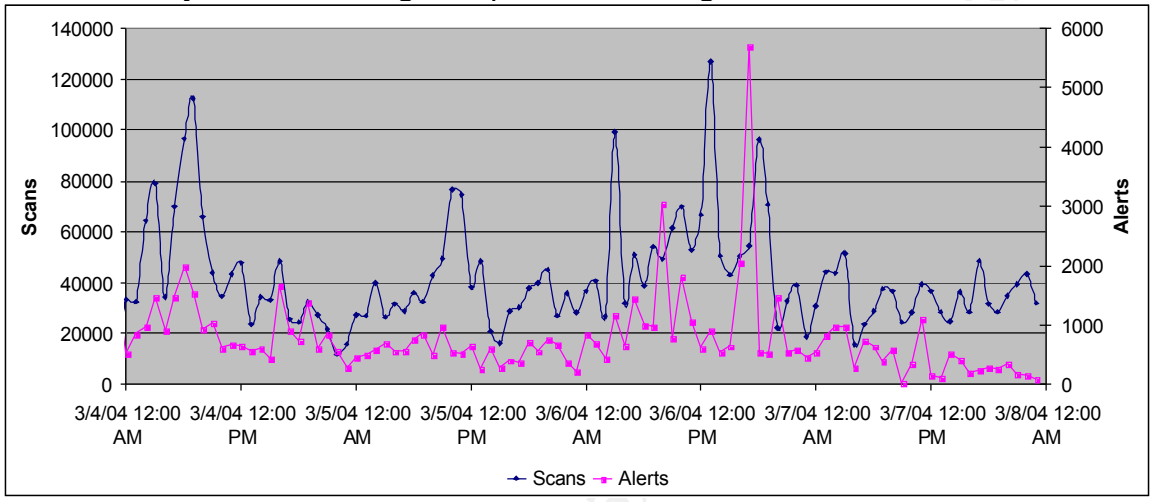
What follows is a detailed listing of Alerts and Scans, and anomalous activity found within the log files provided. The OOS logs had a date in the file name dissimilar from the dates in the logs, because of this OOS files were used based on the date in the logs, rather than the date in the filename.

Log Files

Alerts	Scans	OOS
alert.040303	scans.040303	oos_report_040228.txt (date in logs is 3/3/2004)
alert.040304	scans.040304	oos_report_040229.txt (date in logs is 3/4/2004)
alert.040305	scans.040305	oos_report_040301.txt (date in logs is 3/5/2004)
alert.040306	scans.040306	oos_report_040302.txt (date in logs is 3/6/2004)
alert.040307	scans.040307	oos_report_040303.txt (date in logs is 3/7/2004)

Technical Analysis

As you can see below the rate of alerts and scans are regular and steady with only a few peaks of alerts and scans. The peak activity for both alerts and scans occurs on Saturday, March 6th during afternoon hours, IRC and Novell server activity was found in great quantities during these hours.



Analysis: Alerts

The following is a list of all alerts generated for the five day period. The alerts have been ordered by the number of times the alert occurred over the entire period. Eight different IRC alerts appear in this list, and will be discussed in the Alerts Analysis section below.

Alerts Generated

Alert	Count	Alert	Count
[UMBC NIDS IRC Alert] XDCC client detected attempting to IRC	45276	connect to 515 from outside	24
MY.NET.30.4 activity	17512	TFTP - Internal UDP connection to external tftp server	23
SMB Name Wildcard	6473	EXPLOIT x86 setgid 0	23
MY.NET.30.3 activity	6303	FTP DoS ftpd globbing	19
Incomplete Packet Fragments Discarded	1906	EXPLOIT NTPDX buffer overflow	14
EXPLOIT x86 NOOP	1180	[UMBC NIDS IRC Alert] User joining XDCC channel detected. Possible XDCC bot	11
[UMBC NIDS IRC Alert] IRC user /kill detected, possible trojan.	1138	DDOS mstream handler to client	9
SUNRPC highport access!	1049	EXPLOIT x86 stealth noop	6

High port 65535 tcp - possible Red Worm - traffic	752	EXPLOIT x86 NOPS	6
NMAP TCP ping!	623	SYN-FIN scan!	4
Null scan!	392	External FTP to HelpDesk MY.NET.53.29	4
High port 65535 udp - possible Red Worm - traffic	182	[UMBC NIDS IRC Alert] User joining Warez channel detected. Possible XDCC bot	3
IRC evil - running XDCC	137	DDOS shaft client to handler	3
TCP SRC and DST outside network	81	External FTP to HelpDesk MY.NET.70.50	3
SMB C access	77	External FTP to HelpDesk MY.NET.70.49	3
Possible trojan server activity	66	DDOS mstream client to handler	3
TCP SMTP Source Port traffic	54	RFB - Possible WinVNC - 010708-1	3
FTP passwd attempt	51	TFTP - External TCP connection to internal tftp server	2
[UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected.	47	NETBIOS NT NULL session	2
EXPLOIT x86 setuid 0	44	HelpDesk MY.NET.70.49 to External FTP	2
External RPC call	38	Attempted Sun RPC high port access	1
Tiny Fragments - Possible Hostile Activity	35	TFTP - External UDP connection to internal tftp server	1
[UMBC NIDS] External MiMail alert	32	Probable NMAP fingerprint attempt	1
[UMBC NIDS IRC Alert] K\:line'd user detected, possible trojan.	30	NIMDA - Attempt to execute cmd from campus host	1
TFTP - Internal TCP connection to external tftp server	28	[UMBC NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC	1
connect to 515 from inside	28	[UMBC NIDS] Internal MiMail alert	1

The chart below lists the noisiest Alert generators from source IP's not in the MY.NET network. The DNS records have been presented for reference. The sources found to exhibit malicious behavior are listed with whois and abuse notification information in the Registrant Information section.

Alerts - Top 10 External IP Generators

Source IP	DNS	Alert Count	Dest IP Count	Dest Port Count
68.50.102.64	bgp01546912bgs.longhl01.md.comcast.net	8695	4	5
68.55.191.197	pcp05510211pcs.owngsm01.md.comcast.net	1710	1	2
68.34.27.67	pcp05404064pcs.towson01.md.comcast.net	1518	1	1
68.55.250.229	pcp261188pcs.howard01.md.comcast.net	1254	2	1
63.159.88.57	0-1pool88-57.nas26.vienna1.va.us.da.qwest.net	962	1	2

209.126.201.99	desire.of.hotgirlz.org	934	2	843
68.55.148.5	pcp259943pcs.howard01.md.comcast.net	859	2	1
161.53.66.27	krov.zvne.fer.hr	665	3	1
68.33.138.193	esx136dhcp705.essex01.md.comcast.net	660	2	3
141.157.21.74	pool-141-157-21-74.balt.east.verizon.net	642	2	2

The chart below lists the noisiest Alert generators from source IP's in the MY.NET network. The list includes the alerts triggered by the hosts.

Alerts - Top 10 Internal IP Generators

Source IP	Alert Count	Alerts Triggered	Dest IP Count
MY.NET.27.103	45281	[UMBC NIDS IRC Alert] XDCC client detected attempting to IRC, SMB Name Wildcard	10
MY.NET.190.97	1413	SMB Name Wildcard	6
MY.NET.70.37	1297	SMB Name Wildcard	1297
MY.NET.11.7	1151	SMB Name Wildcard	1
MY.NET.21.67	724	Incomplete Packet Fragments Discarded	4
MY.NET.21.69	696	Incomplete Packet Fragments Discarded	4
MY.NET.190.93	483	SMB Name Wildcard	9
MY.NET.75.13	361	Possible trojan server activity, SMB Name Wildcard	132
MY.NET.190.92	343	SMB Name Wildcard	5
MY.NET.150.198	267	SMB Name Wildcard	113

The charts below show the most prevalent Alerts broken by external and internal sources. Note that only a few of the alerts appear in both columns, which may be a reflection of the signatures themselves. The signature "MY.NET.30.4 activity" is discussed below, but was written to track only external hosts, since no University hosts were found to be the source of the activity.

The Analysis section below the charts will tie many of the alerts found in the Top Alerts from External Hosts to alerts found in the Top Alerts from Internal Hosts. We will see that, especially in terms of the IRC alerts, some alerts are generated by internal hosts, and the return traffic from an external host will trigger a different alert.

Also note that the linked graph will show a relationship between one of the internal hosts, the internal alerts generated, and the alerts generated by external hosts on the return traffic to this host.

Alerts – Top Alerts, External Hosts

Alert	Count
MY.NET.30.4 activity	17512
MY.NET.30.3 activity	6303
EXPLOIT x86 NOOP	1180
[UMBC NIDS IRC Alert] IRC user /kill detected, possible trojan.	1138
SUNRPC highport access!	1049
NMAP TCP ping!	623
High port 65535 tcp - possible Red Worm – traffic	406
Null scan!	392
High port 65535 udp - possible Red Worm – traffic	117
Incomplete Packet Fragments Discarded	105

Alerts – Top Alerts, Internal Hosts

Alert	Count
[UMBC NIDS IRC Alert] XDCC client detected attempting to IRC	45276
SMB Name Wildcard	6473
Incomplete Packet Fragments Discarded	1801
High port 65535 tcp - possible Red Worm - traffic	346
IRC evil - running XDCC	137
High port 65535 udp - possible Red Worm - traffic	65
Possible trojan server activity	37
connect to 515 from inside	28
TFTP - Internal TCP connection to external tftp server	17
DDOS mstream handler to client	9

Analysis of Most Frequent or Worrisome Alerts

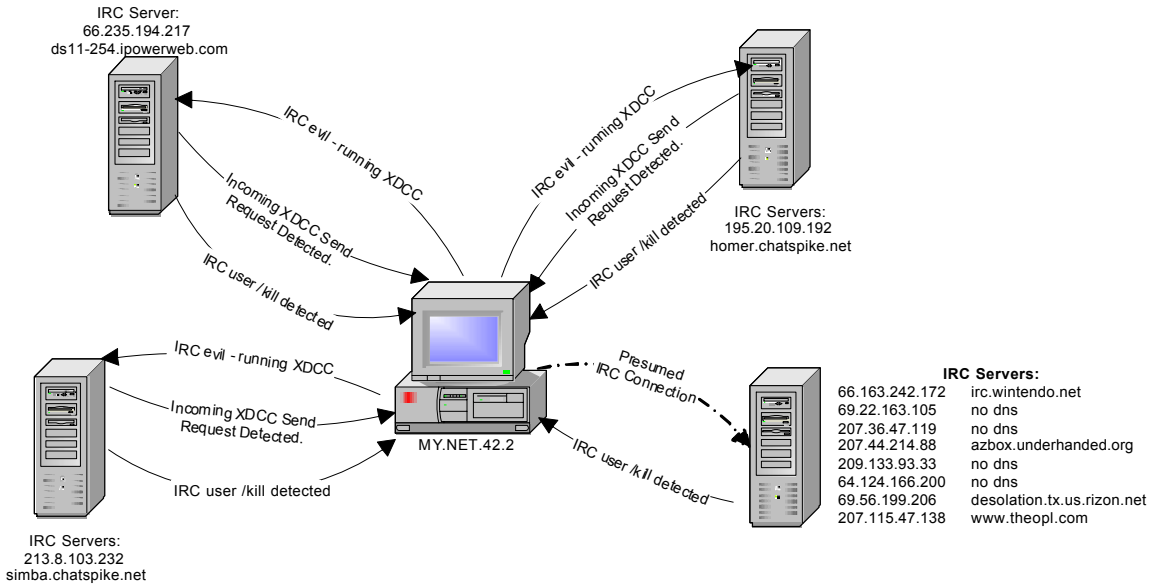
IRC Alerts

The "[UMBC NIDS IRC Alert] XDCC client detected attempting to IRC" is a custom alert, and not found as a default signature on www.snort.org. 99% of the activity was generated by traffic from internal host MY.NET.27.103 to 209.126.201.99 (desire.of.hotgirlz.org) over normal IRC ports: 6667, 6668, 7000, and 6669. By analyzing how many times per hour the activity occurred, hovering around a thousand alerts per hour, and the constancy of the alerts, one would suspect that the host MY.NET.27.103 is a member of a botnet. Judging from the signature's name the IT team is well aware of the signs of the activity, but may not have had time to track down this zombie host. The XDCC client has been used as an element in a few malicious code samples including Aladinz, detailed in the write up at: backdoor.irc.aladinz.l. The version of Aladinz detailed in the write up had only been discovered a few days before the date in the logs, making it a likely candidate if the victim host did not yet have an anti-virus update for this malicious code.

The "[UMBC NIDS IRC Alert] IRC user /kill detected, possible trojan" alert was almost entirely generated by 209.126.201.99, just over 82% of the "…IRC user /kill…" alerts. The destination IP for all 209.126.201.99 "…IRC user /kill…" alerts were always the host MY.NET.27.103. The "user /kill" is an IRC administrator command. The IRC administrator at 209.126.201.99 probably detected a botnet being hosted at his site "desire.of.hotgirlz.org" and was killing IRC connections to particular channels matching that of the botnet.

The other main host detected generating the alert "[UMBC NIDS IRC Alert] IRC user /kill detected, possible trojan" was 65.248.51.47, with 7.9% of the "…IRC user /kill…" alerts. The alerts were always directed towards the internal host MY.NET.42.3.

The "IRC evil - running XDCC" and "… Possible Incoming XDCC Send Request Detected." signatures have a relationship. The first signature detects the internal host connecting to an IRC channel and running XDCC bot. The second signature detects the IRC server sending files to the internal host. The linked graph below shows the relationship between these signatures in reference to the host at MY.NET.42.2 and all of the IRC servers the host connected to.



Recommendations:

The host MY.NET.27.103 should be examined for compromise since rate of IRC connections and alerts generated are likely the result of an IRC botnet named Aladinz.

The hosts at MY.NET.42.2 (shown above), MY.NET.42.3, and MY.NET.42.6 are actively participating in IRC communications. Theses hosts should be examined for possible compromise. All hosts should have anti-virus installed and scanned for virus infections regularly.

MY.NET.30.4 activity and MY.NET.30.3 activity

This is a custom alert, and is designed to be triggered by any connection to the host MY.NET.30.4 and to host MY.NET.30.3. Considering this alert was created to track all activity to this host, it is likely a treasured asset. A total of

361 distinct external sources, and 99% of the logged activity triggered this alert connecting to MY.NET.30.4 or MY.NET.30.3 on ports 51443/tcp, 524/tcp and 80/tcp. Joanne Schell noted this activity in her analysis from August 25, 2003 as belonging to a Novell NetWare implementation (p. 43). Based on these observations it is my estimate that these alerts are precautionary only, and are alerting off of normal traffic, most likely generated by the University's Staff or Students from off campus. This is reasonable to assume since the connecting hosts (top 10 listed in a chart below) generally belong to similar ISP's from very similar locations ("howard" and "balt" are prevalent in the DNS records). The rest of the activities seen directed at these hosts were scans for well known services as seen by nearly all hosts on the network.

As the signature seems to indicate the critical nature of these servers, below is a listing of the top 10 hosts and all of the ports connected to for MY.NET.30.4 or MY.NET.30.3:

Host	DNS	Ports Connected (TCP	Instances
68.50.102.64	bgp01546912bgs.longhl01.md.comcast.net	51443, 80	8692
68.55.191.197	pcp05510211pcs.owngsm01.md.comcast.net	51443, 80	1710
68.34.27.67	no dns	524	1518
68.55.250.229	pcp261188pcs.howard01.md.comcast.net	524	1254
63.159.88.57	0-1pool88-57.nas26.vienna1.va.us.da.qwest.net	51443	943
68.55.148.5	pcp259943pcs.howard01.md.comcast.net	524	859
68.33.138.193	esx136dhcp705.essex01.md.comcast.net	51443, 80	659
141.157.21.74	pool-141-157-21-74.balt.east.verizon.net	524	635
68.49.76.164	pcp04635310pcs.gambrl01.md.comcast.net	524	518
131.92.177.18	aeclt-cf00a4.apgea.army.mil	524	510
151.196.113.173	pool-151-196-113-173.balt.east.verizon.net	51443, 524	268

SMB Name Wildcard

This alert was not found on www.snort.org, but was found on a few user group sites (Vision, p. 1 & Martin, p. 1). The alert is generated from a host sharing their network drive to another host; in this case the second host may be an attacker. When correlated with probe activity from an attacker the SMB Name Wildcard alert may indicate that the host is responding to exchange NetBIOS names, and reveal available network shares. I queried for all hosts responding to the probing host with a SMB Name Wildcard alert in order to determine what ports were available on the hosts scanned for, a technique used in Pete Storm's analysis (p. 59).

The following hosts were found to be sharing their network drive over the ports listed:

Host	Ports Shared to External Scanning Attackers
MY.NET.109.86	1080, 1813, 21, 25, 3128, 443, 4899, 6129, 80, 8000, 8080
MY.NET.150.198	1080, 1813, 20168, 21, 25, 3128, 4000, 443, 4899, 6129, 7755, 80, 8000, 8080
MY.NET.150.44	1080, 1813, 20168, 21, 25, 3128, 4000, 443, 4899, 6129, 7755, 80, 8000, 8080
MY.NET.190.102	135
MY.NET.190.93	135
MY.NET.190.97	135
MY.NET.42.11	8000, 8080
MY.NET.42.2	21
MY.NET.42.4	1080, 3128, 80, 8080
MY.NET.42.7	1080, 3128, 80, 8080
MY.NET.75.13	25
MY.NET.84.151	1080, 1813, 3128, 80, 8000, 8080,
MY.NET.84.253	1080, 1813, 3128, 80, 8000, 8080,

A quick vulnerability scan could determine if these hosts are vulnerable to common vulnerabilities existing on these ports, or have proxying available for the entire internet on the common proxy ports. Network administration should determine if the hosts are compromised, and take actions necessary to quarantine and clean the hosts as quickly as possible.

Incomplete Packet Fragments Discarded

This alert was available in an older Snort version, and has been discussed on the Snort user group list (Roesch, p. 1). The alert is generated by an older snort fragmentation preprocessor, the most current fragmentation preprocessor is the frag2, released first in Snort's version 1.8. The alert is most likely caused by a bad NIC somewhere between the source and destination, resulting in incomplete packets and various packet corruptions. The sources of the alert are mostly to or from the hosts MY.NET.21.67, MY.NET.21.69, MY.NET.21.68, or MY.NET.21.89. A damaged router or cable somewhere near these sources may be to blame. I would recommend upgrading the preprocessor and attempting to find the corrupted hardware generating this signature.

EXPLOIT x86 NOOP

This alert is generated by an attack that contains shell code or no-op code to pad a request, which is common in buffer overflow attacks and worms. The alert is very good as a secondary indicator of an attack. Of the 157 unique source IP addresses generating the alert "EXPLOIT x86 NOOP" none of the

sources were in the MY.NET network. Just over half of the sources were also found in the scan logs. Of the hosts found scanning, all but two were scanning for the RPC DCOM port 135/tcp alone or with a combination high port, most of the high ports were 4444/tcp. The return traffic from these probes was investigated and the university sources: MY.NET.190.102, MY.NET.190.93, MY.NET.190.95, and MY.NET.190.97, returned "SMB Name Wildcard" (the alert is discussed above) to the attacking hosts. Unless the attacked MY.NET network hosts exhibit signs of compromise, such as excessive scanning or Exploit code to other hosts, which none of the university sources did, the traffic can probably be ignored.

SUNRPC highport access!

This alert is used to find connections from external hosts to port 32771/tcp in older versions of Snort. This signature has been replaced by more robust signatures to detect various portmapper vulnerabilities, since normal communications can use this ephemeral port. The source port in 92% of the "attacks" were port 80/tcp, leading me to believe that the destination in the logs were participating in normal web browsing, and happened to use the port as the client side port. I recommend installing a new set of signatures designed to detect the attack activity with less false positives.

NMAP TCP ping!

In nmap there is an option for using TCP ACK = 0 instead of using ICMP to ping a host, since some sites do not allow ICMP through their firewall. According to RFC 793 the host receiving an incorrect ACK number must send a TCP RST, and therefore live hosts not protected by this type of probing would be seen as live hosts to nmap. Most hosts exhibiting the activity use the default port 80/tcp as the source port, to make it look like normal web traffic to a firewall. This activity should be considered reconnaissance only, and all sources should be examined for follow on activity. In this case none of the sources were found with other alert signatures, nor were the sources found in any of the scan logs.

Null scan!

In several scanners a Null scan, one in which no TCP options are set, is used to determine what operating system the attacking host is running, since the response to this scan varies by the operating system scanned (Fyodor, p. 1). The nmap OS fingerprinting documentation at: http://www.insecure.org/nmap/nmap-fingerprinting-article.html is a wealth of resource on the details involved with determining OS from sending a host varying TCP flag options.

All of the 82 unique source IP addresses alerting with "Null scan!" were

found in the Scan logs with 131 *other* combinations of TCP options set such as: ******S* (21%), **U**RS* (5%), **U**RSF (4%), **U***** (2%), *2**P*S* RESERVEDBITS (1%), 12UAPRSF RESERVEDBITS (1%), **U*P*S* (1%). The sources found to be exhibiting this behavior are certainly participating in reconnaissance. Only eight of the sources were found with other alert types such as: Probable NMAP fingerprint attempt, TCP SMTP Source Port traffic, Tiny Fragments - Possible Hostile Activity, and SYN-FIN scan. All of these alerts should also be considered reconnaissance, and these sources should be monitored for follow on attack activity in the near future.

High port 65535 tcp - possible Red Worm – traffic

The alert "High port 65535 tcp - possible Red Worm – traffic" is intended to detect Red (or Adore) Worm traffic. According to analyst Sami Rautiainen in the F-Secure write up on the Adore worm, the worm will activate and open a backdoor on port 65535/tcp upon receiving a ping packet of the correct size (p. 1). In about half of the cases the source ports used were below 1024/tcp, leading me to believe the 65535/tcp is a coincidental ephemeral port. The most interesting source generating the alert was MY.NET.97.88. The alert was triggered by the MY.NET.97.88 host between 2004-03-05 20:30:14.000 and 2004-03-05 20:44:46.000, a short 15 minutes, to just 10 different destinations. The source port for the MY.NET.97.88 ranges between 1780/tcp and 1846/tcp, a range which may be used for an application, The MY.NET.53.36 had similar traffic, but with source port 6257/tcp, the WinMX application. This activity will be discussed later in the report under "Peer-to-Peer" application usage.

Scans

Only 17% of scan logs are from external hosts, which are explained by the Peer-to-Peer and Infected Host sections below.

The chart below lists the most prevalent scan flag types seen in the five day period. As mentioned earlier when discussing the Null Scan! alert, the most common flag combination for scanning hosts is the SYN scan.

Top Scan Flag Types

Flags	Scan Logs
******S*	1,345,283
*******F	12883
12****S* RESERVEDBITS	6094
***A*R*F	967
********	256
*2*A**S* RESERVEDBITS	32
1****R** RESERVEDBITS	31
*2***R** RESERVEDBITS	26

URS*	24
URSF	18

The chart below lists the noisiest scan generators from source IP's not in the MY.NET network. The DNS records, where available, have been presented for reference. The sources found to exhibit especially malicious behavior are listed with whois and abuse notification information in the Registrant Information section.

Scans - Top 10 External IP Generators

Source IP	DNS or Short Whois Info	Dest IPs	Scan Logs	Target(s)
204.152.186.189	www.dnsbl.us.sorbs.net	3	36731	One each of High Ports on MY.NET.1.3 and MY.NET.1.4 and a full vertical scan (30714 ports) on MY.NET.25.70.
203.210.150.36	"localhost" Whois: Vietnam Posts and Telecommunications	6532	32573	Popular Proxy Ports: 81/tcp, 1080/tcp, 3128/tcp, 8000/tcp, 8080/tcp, 8081/tcp
61.190.81.190	unknown; Whois: CHINANET Anhui province network	4451	28164	Popular Proxy Ports: 80/tcp, 1080/tcp, 1813/tcp, 3128/tcp, 8000/tcp, 8080/tcp, 8088/tcp, 8888/tcp, 65506/tcp
61.190.81.118	unknown; Whois: CHINANET Anhui province network	5509	26894	Popular Proxy Ports: 80/tcp, 1080/tcp, 1813/tcp, 3128/tcp, 8000/tcp, 8080/tcp, 8088/tcp, 8888/tcp, 65506/tcp
217.34.37.208	host217-34-37-208.in-addr.btopenworld.com	15124	25061	4000/tcp
149.166.207.227	in-207-227.dhcp-149-166.iupui.edu	14930	24225	4000/tcp
202.179.154.6	unknown; Whois: EXCELNET CYBER HOUSE, PAKISTAN	13100	23766	Web Services: 80/tcp, 443/tcp
80.180.143.101	host101-143.pool80180.interbusiness.it	5771	19904	Proxy ports: 1080/tcp, 3128/tcp, 4480/tcp, 8080/tcp And ports: 3127/tcp and 3332/tcp ?
12.35.193.194	ip-12-35-193-194.hqglobal.net	12069	16607	20168/tcp

129.22.166.233	kml7.STUDENT.CWRU.Edu	10285	135	224000/tcp

The chart below lists the noisiest Alert generators from source IP's in the MY.NET network. The list includes the target ports scanned by the hosts.

Scans Top 10 Internal Generators

Source IP	Dest IPs	Scan Logs Generated	Target(s)
MY.NET.1.3	84412	2532647	DNS, NTP and ephemeral UDP Ports: 53/tcp, 123/udp, 45002/udp, 10123/udp…
MY.NET.53.169	22626	354546	Gnutella Peer to Peer Ports: 6346/udp, 6346/tcp, 6348/udp…
MY.NET.1.4	24292	313739	DNS, NTP and ephemeral UDP Ports: 53/tcp, 123/udp, 45004/udp, 10123/udp…
MY.NET.110.72	12239	261423	"Teamspeak" an IRClike Gaming Communication Application with Source ports 8767/udp, 12203/udp, 12300/udp, 32808/udp… Destination Ports are all well distributed amongst ephemeral udp ports.
MY.NET.34.14	1137	183514	SMTP & ident Ports 25/tcp and 113/tcp
MY.NET.80.224	81113	140981	Bursts of RPC scanning Activity on 3/4 and 3/6 (5000 hosts+ per hour); 80/tcp and 443/tcp
MY.NET.81.39	125667	126339	Bursts of RPC scanning Activity on 3/4 and 3/5 (5000 hosts+ per hour) and 80/tcp
MY.NET.153.79	10084	111808	eDonkey2000 Peer to Peer Traffic port 4672/udp, 4665/udp …
MY.NET.82.15	1884	63379	Source Ports are Napster Peer to Peer Traffic ports: 8888/udp & 8889/udp
MY.NET.97.74	17037	48816	Gnutella Peer to Peer Ports: 6346/tcp, 6348/tcp, 2500/tcp…

The list below shows the most commonly scanned for ports, and the common service name for those ports as a reference. The chart is ordered by the total number of logs seen, and the log field is split by number of logs seen from external hosts, and number of logs seen by internal hosts.

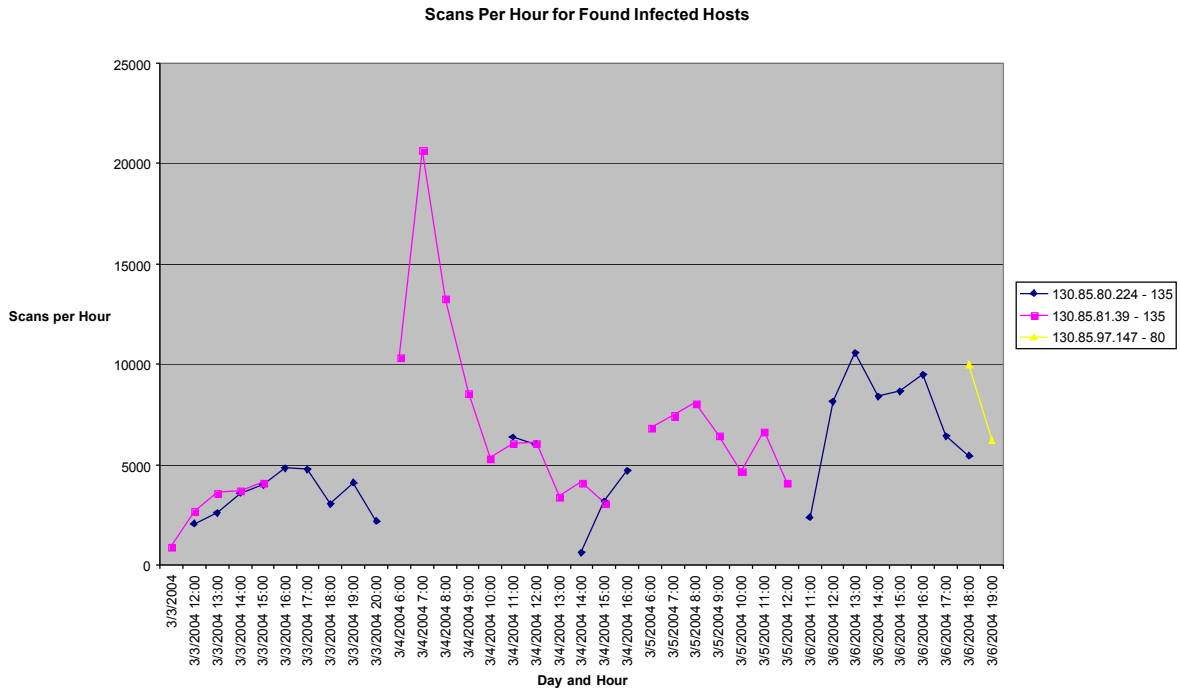
Scans for Common Ports

Port	Common Name	Internal Scan Logs	External Scan Logs
53/udp	DNS	2427943	54
135/tcp	RPC DCOM	253610	7982
25/tcp	SMTP	154401	26571
6129/tcp	DameWare	12035	127259
80/tcp	HTTP	19354	90979
443/tcp	HTTPS	79	71120
4000/tcp	ICQ / Terabase	122	62832
21/tcp	FTP	124	52385

6346/tcp	Gnutella	40960	285
6346/udp	Gnutella	39012	1

Possible Infected Hosts

A query was run against the data to detect hosts that scanned common ports: 53/udp, 137/udp, 1434/udp, 445/tcp, 80/tcp, 139/tcp, 17300/tcp, and 135/tcp, and scanned at a rate greater than 500 hosts in an hour. The following graph shows the detected three hosts, two scanning for 135/tcp and one for 80/tcp that may be infected with a worm. The scanning for 135/tcp on host MY.NET.81.39 is particularly worrisome. The scanning for this host peaked at more than 20,000 hosts in an hour. These hosts should immediately be quarantined, virus scanned and possibly re-imaged to remove all malicious code.



Peer-to-Peer Application Use

Peer-to-Peer usage can eat bandwidth, and may cause the user to infect themselves, much in the same way the user will infect themselves through email viruses. Peer-to-Peer viruses are particularly clever with social engineering tactics such as copying itself to file sharing directories as a “Crack” program or even an antivirus program. Peer-to-Peer usage can be especially damaging to bandwidth when a user’s machine is picked as a “supernode” and

every user looking for awesome pictures of Britney Spears has a quick look through your network to find the best place to get it. In my experience "supernoding" on Peer-to-Peer networks on companies with otherwise fabulous bandwidth has been known to bring very pricy firewalls to their knees.

The University of Chicago has released a comprehensive list of ways to blocking peer-to-peer file sharing, which includes ways to block WinMX (p. 1). The University of Chicago has also released a letter to the University concerning Peer-to-Peer file sharing and the usage policy for the University (p. 1). Their letter explains the legal actions they feel obligated to report to the federal authorities if they find a user sharing copywrited material. If the Unknown University has not already done so, a letter to the University community is highly recommended.

The following internal hosts were found to be scanning on ports for well-known Peer-to-Peer Applications. Unfortunately it is not always effective to block Peer-to-Peer Applications by their default port, since most file sharing applications can now dynamically negotiate the port. It is more effective for network administrators to attempt to slow the flow of file sharing applications by reducing the bandwidth allowed by certain network segments.

Host	Port	Peer to Peer Application	Scan Logs
MY.NET.53.169	6346	Gnutella (or variant)	35251
MY.NET.97.74	6346	Gnutella (or variant)	29674
MY.NET.153.79	4662	EDonkey2000	25516
MY.NET.53.122	6346	Gnutella (or variant)	7504
MY.NET.153.97	4662	EDonkey2000	5117
MY.NET.97.49	6346	Gnutella (or variant)	4625
MY.NET.53.158	6346	Gnutella (or variant)	3152
MY.NET.97.31	6346	Gnutella (or variant)	2398
MY.NET.84.235	4662	EDonkey2000	1882
MY.NET.84.203	6346	Gnutella (or variant)	1601
MY.NET.53.155	6346	Gnutella (or variant)	1405
MY.NET.152.251	6257	WinMX	785

OOS Logs

Top External OSS Generators

Source IP	DNS	OOS Logs	Destination IP's
68.54.84.49	pcp01741335pcs.howard01.md.comcast.net	1079	1
217.125.5.139	139.Red-217-125-5.pooles.rima-tde.net	402	4
66.225.198.20	unknown.servercentral.net	115	1
67.114.19.186	adsl-67-114-19-186.dsl.pltn13.pacbell.net	81	1
68.122.128.1	adsl-68-122-128-1.dsl.sndg02.pacbell.net	58	1

207.126.229.20	pe-ts10.ycnsg.corp.yahoo.com	58	7
35.8.2.252	mdlv2.h-net.msu.edu	46	1
63.194.83.210	adsl-63-194-83-210.dsl.snfc21.pacbell.net	44	3
63.71.152.2	wall.turbinegames.com	30	1
213.193.132.2	gw-medisearch.gdbru.be.easynet.net	21	1

Most interesting in the OOS logs is the 68.54.84.49 generating over a thousand logs for essentially the same traffic to host MY.NET.6.7 on port 110/tcp (POP3). The source of the data does not appear in either the alert logs or the scan logs, indicating that the activity may be quite benign in nature. All of the logs occur each day between midnight and 0:500 AM. Every OOS packet picked up by the Snort has the 12****S* flags set, indicating that the Explicit Congestion Notification bit and the Congestion Windows Reduced flag have been set. As explained in the article “ECN and it's impact on Intrusion Detection” by Toby Miller these flags can be set during the three way hand shake to negotiate their network traffic and a response from the host must also contain ECN bits set to complete the negotiation (p. 1). Since the host at MY.NET.6.7 did not return any traffic with these bits set, the host does not use ECN. In fact 95% of the OOS logs had the 12****S*, but none of the traffic has ANY of the MY.NET as a source address.

However, the traffic from 217.125.5.139 has these same flags set, but is likely generated by Peer to Peer traffic since all of the ports are either 4662/tcp (EDonkey2000) or 6881/tcp (BitTorrent p2p). The Snort is again picking up the traffic on the 12****S* flags set, and is also likely caused by the more advanced options picked by the 217.125.5.139, but again no bits were set on the return traffic evidenced by the lack of any of the Destination IPs in the OOS logs.

Registrant Information

The following Registrant Information was chosen because of the prevalence of these IP’s in the Top Generators categories, and sources that were not eliminated as false positives due to normal Unknown University assignment hosting traffic.

IP checks	Whois	Abuse/Personal Contact Information
209.126.201.99	NetRange: 209.126.128.0 - 209.126.255.255 CIDR: 209.126.128.0/17 OrgName: California Regional Internet, Inc. OrgID: CALI Address: 8929A COMPLEX DRIVE City: SAN DIEGO StateProv: CA PostalCode: 92123 Country: US	AbuseHandle: ABUSE341-ARIN AbuseName: Abuse AbusePhone: +1-858-974-5080 AbuseEmail: abuse@cari.net

204.152.186.189	NetRange: 204.152.184.0 - 204.152.191.255 CIDR: 204.152.184.0/21 OrgName: INTERNET SOFTWARE CONSORTIUM, INC. OrgID: V6IS Address: 950 CHARTER STREET City: REDWOOD CITY StateProv: CA PostalCode: 94063 Country: US	OrgTechHandle: PV15-ARIN OrgTechName: Vixie, Paul OrgTechPhone: +1-650-423-1300 OrgTechEmail: vixie@isc.org
202.179.154.6	inetnum: 202.179.128.0 - 202.179.159.255 netname: EXCELNET descr: CYBER HOUSE descr: INTERNET SERVICE PROVIDER & SOFTWARE DEVELOPER country: PK admin-c: EH15-AP tech-c: EH15-AP mnt-by: APNIC-HM changed: hostmaster@apnic.net 19990621 status: ALLOCATED PORTABLE source: APNIC	person: ehtsham ul haque address: Suite 1 Al Mustafa Plaza 6th road Satellite Town Rawalpindi 44000 country: PK phone: +92-51-457618 fax-no: +92-51-414879 e-mail: ehtsham@isb.compol.com nic-hdl: EH15-AP mnt-by: MAINT-NEW changed: ehtsham@isb.compol.com 19990516
203.210.150.36	inetnum: 203.210.128.0 - 203.210.191.255 netname: VNN-VN descr: Vietnam Posts and Telecommunications (VNPT) descr: 23 Nguyen Du street, Hanoi capital, Vietnam country: VN	person: Khanh Nguyen Hien address: Vietnam Datacommunications Company (VDC) address: 258 Ba Trieu street, Hanoi capital, Vietnam country: VN phone: +84-4-8212680 fax-no: +84-4-9760397 e-mail: pbthuy29@vnn.vn nic-hdl: KNH1-AP
61.190.81.190	inetnum: 61.190.0.0 - 61.190.255.255 netname: CHINANET-AH descr: CHINANET Anhui province network descr: China Telecom descr: A12,Xin-Jie-Kou-Wai Street descr: Beijing 100088 country: CN	person: Jinneng Wang address: 17/F, Postal Building No.120 Changjiang address: Middle Road, Hefei, Anhui, China country: CN phone: +86-551-2659073 fax-no: +86-551-2659287 e-mail: wang@mail.hf.ah.cninfo.net
61.190.81.118	inetnum: 61.190.0.0 - 61.190.255.255 netname: CHINANET-AH descr: CHINANET Anhui province network descr: China Telecom descr: A12,Xin-Jie-Kou-Wai Street descr: Beijing 100088 country: CN	person: Jinneng Wang address: 17/F, Postal Building No.120 Changjiang address: Middle Road, Hefei, Anhui, China country: CN phone: +86-551-2659073 fax-no: +86-551-2659287 e-mail: wang@mail.hf.ah.cninfo.net

Analysis Methodology

I began by reading quite a few GCIA practicals for part 3 in hopes of developing a theme, and looking for particular formatting that assisted me in understanding what was going on. I found Pete Storm's and Jason Thompson's practicals, listed in my references, the most useful for guiding my formatting.

As my first and most highly honed skill is in TSQL for MS SQL, I instinctively sought to parse the data into files that could be bulk inserted into a database. Perl was used to parse through the data and separate the fields by pipe symbol, a symbol not found in the log files. My Perl skills being quite rusty and the files not being quite so predictable resulted in a lot of trial and error. The scans script (shown below) was quite simple.

```
use FileHandle;

open LOG, ">scan.txt";

sysopen IFILE, "scans.040307", O_RDONLY or print "Unable to open file";

while (<IFILE>){
        chomp($_);
        @field = split / /;
        @ip1 = split(/:/, $field[4]);
        @ip2 = split(/:/, $field[6]);

        print LOG "$field[0] $field[2], 2004
$field[3]|$ip1[0]|$ip1[1]|$ip2[0]|$ip2[1]|$field[7]|$field[8] $field[9]|\n"
}
```

The first alerts script took many case statements to get the IP and port numbers into separate fields, and produced many errors while importing since a few of the alerts did not obey the same IP:Port -> IP:Port formatting, namely the fragment alerts. I took out the scan alerts, since scans are handled by the scan log files. After a few iterations the script was changed a simpler structure, with no case statements, but a few if-then-else statements. The resultant data from the following query did not error when inserting into the database:

```
use FileHandle;

open LOG, ">alert.txt";

sysopen IFILE, "alert.040307", O_RDONLY or print LOG "Unable to open file";

while (<IFILE>){
        $blah = $_;
        chomp($blah);
        @upperfield = split(/\[\*\*\]/, $blah);
        @time1 = split(/-/, $upperfield[0]);


        @time2 = split(/\./, $time1[1]);
```

```
        if ($upperfield[1] =~ m/spp_portscan/) {

                }
        elsif ($upperfield[1] =~ m/Frag/){
                print LOG "$time1[0]/04 $time2[0]|";
                print LOG $upperfield[1]. "|";
                $upperfield[2] =~ s/->/|0|/;
                $upperfield[2] =~ s/ //g;
                print LOG $upperfield[2] . "|0|\n";
                }
        else {  print LOG "$time1[0]/04 $time2[0]|";
                print LOG $upperfield[1]. "|";
                $upperfield[2] =~ s/->/|/;
                $upperfield[2] =~ s/:/|/g;
                $upperfield[2] =~ s/ //g;
                print LOG $upperfield[2] . "|\n";

                }


}
```

The OOS logs were also difficult to parse, since they flow over many lines, but I decided to reset the line counter for each time I encountered a line with: =+=+= all the way across it. The result in Perl was:

```
use FileHandle;

open LOG, ">OOS.txt";

sysopen IFILE, "oos_report_040303.txt", O_RDONLY or print LOG "Unable to open file";

$i = 1;

while (<IFILE>){

        $blah = $_;
        chop($blah);
        $i = 6 if ($blah =~ m/\=\+\=/);

        @field = (split / +/, $blah);

        if ($i % 7 == 1) {
```

```
                @time1 = split(/-/, $field[0]);
                @time2 = split(/\./, $time1[1]);
                @src = split(/:/, $field[1]);
                @dest = split(/:/, $field[3]);
                print LOG "$time1[0]/04
$time2[0]|$src[0]|$src[1]|$dest[0]|$dest[1]|";
                }
        elsif ($i % 7 == 2) {print LOG "$field[0]|$blah"}
        elsif ($i % 7 < 6)  { print LOG " $blah" }
        elsif ($i % 7 == 6) { print LOG "|\n"}
        $i++;

}
```

As for the actual analysis, most of the SQL queries were not time intensive to write. The most intensive query was one written to find hosts with SMB Name wildcard responses to common port scanning. The query assumes that a port is open if the host returns a SMB Name query within a second of the port scan. The query for finding hosts responding to scanners with SMB Name wildcard is shown below:

```
select distinct srcip, destport, timestampstd into #temptable
from tblscans
Where srcip not like 'MY.%'
and destport in (6129,80,443,4899,20168,4000,
        21,25,3128,1080,8080,8000,7755,1813,135
)  --top ports scanned for

select distinct a.srcip, b.destport , alert
        from tblalerts a
join #temptable b
on a.destip = b.srcip
and datediff(second, a.timestampgmt, b.timestampgmt)< 1
order by a.srcip, b.destport
```

During the analysis it was found that a few hosts may be infected with a worm. Using the query below I found three definite hosts infected. The query assumes that an infected host will query for the same port to more than 500 different hosts on the internet in an hour. A normal user participating in normal user network usage will not make this many calls to the same port but so many different hosts per hour. The hosts were found querying many thousands of hosts in an hour.

```
select srcip, destport, count(*), dateadd(hour, datepart(hour,
timestampstd), convert(datetime, convert(varchar(20), timestampstd,
101))) /* format the time so excel likes it better */
```

```
from tblscans
where srcip like 'MY.NET%'
and destport in
        (53, 137, 1434, 445, 80, 139, 17300, 135)
        --common worm infection scanning ports
and srcip not in
        ('MY.NET.1.3', 'MY.NET.1.4') --DNS Servers
group by srcip, destport,  dateadd(hour, datepart(hour, timestampstd),
convert(datetime, convert(varchar(20), timestampstd, 101)))
having count(*) > 500
```

Once the data is collected the data is placed in excel and pivot table services was used to view the different scan patterns for the sources found.

The query used for finding the most frequently scanned ports was easy, until I realized that viewing the number of internal logs and external logs was easier for the viewer to understand what was really going on. I used a method of querying each data set separately, and then joining the data together for a better result. The query for presenting this data is shown below:

```
select a.destport, a.proto, isnull(a.cnt, 0), isnull(b.cnt, 0)
from
        (select destport, proto, count(*) cnt
        from tblscans
        where srcip not like 'MY.NET%'
        group by destport, proto) a
left outer join
        (select destport, proto, count(*) cnt
        from tblscans
        where srcip like 'MY.NET%'
        group by destport, proto) b
on a.destport = b.destport
and a.proto = b.proto
order by a.cnt + b.cnt desc
```

References

1. Fyodor. “Remote OS detection via TCP/IP Stack FingerPrinting” Insecure.org October 18, 1998 URL: http://www.insecure.org/nmap/nmap-fingerprinting-article.html
2. Martin, Daniel. “Re: Spoofed SMB name wildcard probes” Incidents@SecurityFocus Mailing List Archive May 4, 2001 URL: http://lists.jammed.com/incidents/2001/05/0034.html
3. Miller, Toby. “ECN and it's impact on Intrusion Detection.” Security Focus November 2000 URL: http://www.securityfocus.com/infocus/1205
4. Rautiainen, Sami. “F-Secure Virus Descriptions : Adore” F-Secure April

2001 URL: http://www.europe.f-secure.com/v-descs/adore.shtml
5. Roesch, Martin. “Re: [Snort-users] Incomplete Packet Fragments Discarded” Snort-users Mailing List November 26, 2001 URL: http://www.mcabee.org/lists/snort-users/Nov-01/msg00820.html
6. Schell, Joanne. “GCIA Practical Version 3.3”. Incidents.org Intrusions Archive August 2003 URL: http://www.giac.com/practical/GCIA/Joanne_Schell_GCIA.pdf
7. Storm, Pete. “GIAC Certified Intrusion Analyst (GCIA) Practical Assignment” Incidents.org Intrusions Archive August 2003 URL: http://www.giac.com/practical/GCIA/Pete_Storm_GCIA.pdf
8. Thompson, Jason. “GIAC: Intrusion Detection In Depth”. Incidents.org Intrusions Archive July 21, 2003 URL: http://www.giac.com/practical/GCIA/Jason_Thompson_GCIA.pdf
9. University of Chicago Networking Services & Information Technologies. “Disabling Peer to Peer File Sharing” [No date] URL: http://security.uchicago.edu/peer-to-peer/no_fileshare.shtml
10. University of Chicago Networking Services & Information Technologies. “Unlicensed distribution of copyrighted materials” October 7, 2003 URL: http://security.uchicago.edu/peer-to-peer/sharing_letter.shtml
11. Vision, Max. “Re: [snort] 'SMB Name Wildcard” Neohapsis Archives Jan 17, 2000 URL: http://archives.neohapsis.com/archives/snort/2000-01/0220.html
12. Whitehats. “IDS177 "NETBIOS-NAME-QUERY" ” [No date] URL: http://whitehats.com/info/IDS177