Intrusion Detection In-Depth

GCIA Practical Assignment, v3.4

Eric Evans

11 December 2003

TABLE OF CONTENTS

ABSTRACT

This GIAC GCIA version 3.4 practical assignment, Intrusion Detection In-Depth, is offered in three parts. Part 1 is a white paper covering an infamous malicious code attack, and ties it in to the current state of intrusion detection, entitled "Slamming the Door on Slammer". Next, a detailed analysis of three disparate network detects is presented in Part 2, "Network Detects". Lastly, Part 3, "Analyze This!" is a security audit for an organization's network. Five days' of logs were scrutinized and the findings are presented along with recommendations for action and a description of the analysis process. Understanding of the tools utilized for analysis in Part 3 is helped by the background material covered earlier, in Part 1. To support this project, a development environment was setup as reinforcement for proof of concept in Part 1 and served as powerful tools for data-mining throughout the analysis process of Parts 2 and 3.

PART ONE – THE STATE OF INTRUSION DETECTION

SLAMMING THE DOOR ON SLAMMER: Detecting and Preventing Slammer Attacks on MS-SQL Server 2000 Databases

INTRODUCTION: Overview of Slammer and MS-SQL Server 2000 and the threats posed.

The aim is to provide the reader with an understanding of the Slammer worm, the vulnerabilities it exploits, the affect of this malicious code on the server and network, and how to safeguard against it. Precautionary actions will be offered for preventing infection; configuring the operating system, database, firewall, and intrusion detection tools. The use of these tools to identify Slammer will be covered as well. Finally, in the event a system becomes infected, actions to eradicate Slammer will be presented.

Now, let's meet the players: Microsoft SQL Server 2000 will play the part of the victim while Slammer will portray the villain.

The Victim: Microsoft SQL Server 2000 is a database. What's a database? A database is simply a record keeping system, really nothing more than a computerized way of cataloging information for later retrieval. SQL Server 2000, often referred to simply as 'MS-SQL', is the latest enterprise-level database from Microsoft. It is, technically, the second generation of this product; versions prior to 7.0 were actually ported from Sybase's flagship product. SQL Server 7.0 was the first truly Microsoft version of this well-known relational database. [1] MS-SQL, as we'll refer to it within this document, takes the 'SQL' part of its name from 'Structured Query Language'. SQL, the standardized language used by database gurus, is in wide use in the world of Relational Database Management Systems (RDBMS). In addition to MS-SQL, Oracle, Sybase, DB2, Informix, and MySQL all make use of SQL although they have each taken

[1] Delany, Kalen

the liberty of augmenting the ANSI and ISO standards to enhance the functionality of their products.

The Villain: Now, let's get to know our villain. Isn't the villain always the most interesting character? Slammer is a worm. In the case of this scoundrel, we *could* mean 'worm' in the figurative sense. Within the context of this white paper however, we're talking about a computer worm; a self-propagating, usually malicious, program that replicates itself from one vulnerable computer system to the next, wreaking whatever havoc its designer intended. We'll learn more about Slammer a bit later.

The Weakness: There exists in Microsoft SQL Server 2000 a vulnerability that permits malicious code to be executed on computers that have not been properly safeguarded. Microsoft announced this vulnerability, which we'll discuss in more detail shortly, in July of 2002 and a patch was released to protect systems running SQL Server 2000. Half a year later, in late January of 2003, computer users, globally, experienced degraded Internet performance. The vulnerability touched upon above was being exploited. When the code involved in this exploit executes on an infected system, it immediately begins a random search for other machines to infect. In turn, each host that is found and infected runs the malicious code and begins its own indiscriminate hunt for additional machines to contaminate. Although relatively few computers were actually infected, compared to the total number online, the Internet as a whole suffered due to the traffic storm that ensued.

SLAMMER EXPLAINED: The 'nuts and bolts' of Slammer; vulnerabilities exploited, how Slammer propagates, potential damage, and a brief history.

The launch SQL Server 2000 brought expanded functionality and was based on a redesigned database engine. Although additions to the feature set were numerous, we're concerned with only one: the ability to run multiple instances of SQL Server, allowing a single computer to act as a handful of database servers. To support multiple instances, Microsoft created the SQL Server Resolution Service (SSRS), listening on port 1434 and providing a means for specifying a particular instance of SQL Server for database queries. SSRS uses UDP protocol, making it very fast compared to TCP because it does not require the TCP protocol's three-way handshake.

There are three vulnerabilities of concern here; two relating to buffer overflow and one denial of service (DoS). In the case of the DoS vulnerability, it is possible for an attacker to craft a "keep-alive" packet and send it to the SSRS of a SQL Server; the computer receiving this packet is, simply, being told to keep the addressed instance of SQL alive and responds to the sender with another keep-alive message. If this packet was sent by another SQL server, or if the IP of an SQL Server had been spoofed by the actual sender, then another keep-alive packet will be sent by the first machine and, of course, the second machine will promptly respond with yet another keep-alive packet. It's easy to see how this endless loop would gobble up resources on each system in short order and adversely affect network bandwidth. In regards to the buffer overflow vulnerability, two things can happen: 1) a crafted packet with very specific data could allow an attacker to execute code on the server, 2) a crafted packet could create a buffer overrun, causing the SQL Server service to fail; this could also be thought of as a DoS attack.

Slammer (also known as SQLSlammer, W32.Slammer, and Sapphire) takes advantage of the first buffer overflow vulnerability described. Small crafted packets are sent on UDP port 1434 with the hope they will find their way to a machine running Microsoft SQL Server 2000 that has not been properly secured. The IP address of the destination host is generated randomly, by means of an algorithm based upon the timestamp of the victim host. Using a computer's timestamp to generate a pseudo-random value isn't a concept unique to Slammer; it's a widely-used approach and is even taught in programming classes. When Slammer finds a machine running Microsoft SQL Server 2000 and if the administrator hasn't followed best practices in regards to securing the machine, then Slammer will begin to do its dirty work.

Slammer's payload is small, only 376 bytes, and is streamlined for efficiency. The payload consists of only four strings: "h.dllhel32hkernQhounthickChGetTf", "hws2", "Qhsockf", and "toQhsend". Interestingly, there's nothing malicious here; no files are downloaded, existing files are not modified, and no registry entries are added or changed; Slammer resides only in memory. The only goal of the code is to find other hosts that can be infected. This, in and of itself, may seem pretty innocuous. The real problem lies in Slammer's effectiveness; it's very good at what it does.

The first thing Slammer does when it finds its victim is to attempt to overrun the SQL Server service. If this is successful, then the worm's code will execute, with the permissions of the account used to run the SQL Server Service, often an account with system-level access. It is important to note that the account being taken advantage of could be used to cause other harm. However, because no data is returned to the originating host in the case of Slammer, it seems unlikely the attacker would know they've struck gold.

Now that the service account has been compromised, all Slammer cares about is self-replication. To do this, Slammer must first decide on an IP address to send itself to. It does this by grabbing the timestamp of the server using the 'GetTickCount' API. Although there are other ways to randomly generate a value, using the timestamp is ideal in this case, in the eyes of the villain, because there's nothing about the timestamp that gives away any clues about the origin of the attack. Then, Slammer sends its payload to the destination IP it just generated. Next, Slammer generates a new destination IP and sends its payload once again. Each successive random IP generated is derived from the original timestamp; Slammer doesn't grab the timestamp anew each time it needs to generate a new IP. The timestamp it already has will do the job just fine; this saves Slammer time and effort that it can better use to propagate itself. What Slammer lacks in size, it makes up for in efficiency.

Consider our first victim, now subservient to our villain, sending out Slammer's payload as quickly as it can generate IP addresses. Now, picture each new susceptible destination host joining in on the merriment. It's easy to see how this could cause plenty of problems; certainly a traffic storm would begin to brew.

Late in the evening of January 24th 2003, the scenario described above is precisely what came to pass. Within a quarter of an hour, the storm was raging as an estimated 75,000 hosts running SQL Server 2000 were compromised. [2] Although this may not seem like a large number of machines, their resources were unwillingly pooled to flood the Internet. Ironically, Slammer was so effectual, it restricted its self-propagation by replicating so quickly that world-wide network bandwidth was "sucked-up" – slowing the worm's spread. Slammer has earned the dubious distinction of becoming the fastest-spreading worm, so far…

PREVENTIVE MEASURES: How to thwart Slammer; precautionary actions to take with the operating system, database, and network perimeter defense.

Now that we understand how Slammer works and the wrath it can bring, we'll discuss preventive measures. This is the first step in "slamming the door" on Slammer. For starters, let's perform a bit of risk assessment so we can see where to focus our energy. Slammer is known to be a threat only if the following conditions are met:

- You have a computer running applications relying on Microsoft SQL Server 2000 or MS-SQL 2000 Desktop Engine (MSDE 2000).
- The computer is running an out-dated service pack.
- You have not applied necessary security patches.
- The machine is connected to a network and listening on UDP port 1434.

MSDE 2000 consists of the database engine from MS-SQL 2000 intended for integration with third-party applications and is available gratis from Microsoft. A more complete list of MSDE 2000-related applications can be found at the links below.

http://www.microsoft.com/technet/treeview/?url=/technet/security/MSDEapps.asp

http://www.securityfocus.com/bid/5311

How do we determine if a particular machine is at risk? One way would be to run netstat.exe, Microsoft's tool for displaying network connections and protocol statistics. netstat.exe is available by default on all recent versions of the Windows operating system and can be run from the command prompt using the command below.

```
netstat.exe -p udp
```

[2] Pilker, Joanne.

We use the “-p” option along with “udp” to display only UDP-related info. Executing the command above will yield a result similar to what’s shown in the example below, to the left side; a list of the UDP ports the machine is listening on. Or, you could run netstat.exe with the “-a” option to list all protocols and ports, regardless of their status

```
netstat.exe -p udp                  net start
   Active Connections                 These Windows 2000 services are started:
   Proto  Local Address               DHCP Client
   UDP  hostname:epmap                Event Log
   UDP  hostname:isakmp               MSSQLServer
   UDP  hostname:ms-sql-m             Plug and Play
   UDP  hostname:netbios-ns           Print Spooler
   UDP  hostname:netbios-dgm          Workstation
```

To see if the MSSQLServer service is installed, run "net help services" at the command prompt. To check if the MSSQLServer service is indeed running, you can run “net start”. The table above, on the right side, shows an example of the output from “net start”.

Simply inspect the output; the presence of “ms-sql-m”, if you ran netstat.exe, will indicate that the machine is in fact running a MSDE 2000-related application and is at risk. If you used “net start” or “net help services”, look for “MSSQLServer” in the output. Alternatively, you can run the “SQL Critical Update”, available on Microsoft’s Slammer Worm Resources page:

http://www.microsoft.com/sql/techinfo/administration/2000/security/slammer.asp

SQL Critical Update will scan a single computer for Slammer-related vulnerabilities. Another, perhaps better, method is to run Microsoft’s “SQL Scan”, available at the same location as the SQL Critical Update. SQL Scan is capable of scanning an individual computer, a range of IP addresses, or even an entire domain to assist in tracking down systems that may be at risk. Also available on Microsoft’s Slammer Worm Resources page is the Systems Management Server (SMS) Deployment Tool that, if your network operating system is Microsoft and you’re using SMS, can “automagically” deploy the SQL Server Critical Update to machines identified with SQL Scan.

Security patches for the operating system should be applied immediately to any machines found to be vulnerable. Next, the latest SQL Server 2000 Service Pack (SP3a) should be applied to these systems. You’ll also want to ensure the account used to run the SQL Server service has only the permissions it needs to carry out its tasks. Running this service with an account having domain administrator or super-user database-level permissions would be a poor idea. Using the “System” account or another account with no more than local administrator permissions should suffice.

To determine if MS-SQL 2000 installed and to check the version, search your system for the file “sqlservr.exe”, the MS-SQL Server service. Right-click on the file and select

"Properties" see the version number. Next, check for "MSSQLSERVER" on the list in Services from the Control Panel. From here, you can see if the MS-SQL Server service and its startup option (Automatic, Manual, and Disabled). "Automatic is the default when installing MS-SQL. If the service is not needed, set this option to "Disabled" or uninstall it completely. All of these things can be checked without starting the MSSQLSERVER service. To check the service pack version, open Enterprise Manager, right-click on the server icon and select properties. Caution: even if the MSSQLSERVER service is *not* running, it will *automatically* start when you view properties.

Note: We're concentrating on Windows-specific actions, because applications vulnerable to Slammer only run on Microsoft's OS; it's an operating system and application specific exploit.

Finally, if it isn't necessary that your SQL Server be "seen" from outside your network, another precaution you can take is to block UDP port 1434, both inbound and outbound, at your firewall. This will prevent systems on your network from joining in on a Slammer attack. If you apply SP3a, as recommended, listening on port 1434 will be disabled by default. Note: If running MSDE 2000, install its latest patches; SP3a isn't compatible.

DETECTION: Selecting, configuring, and using IA Tools to detect Slammer.

Finally, we get to the "news you can use" for the intrusion analyst. Everything covered thus far has been "must know" background information for Slammer, but now it's time to get down to "brass tacks". Note: Part 3 of this practical, "Analyze This!", uses MS-SQL as the data-mining tool for analysis; so information covered in here in Part 1 will remain pertinent throughout this document.

Obviously, if Slammer tries to squeeze by our preventive measures, we'd like to know about it. This is where an intrusion detection system comes into the picture. An intrusion detection system (IDS) monitors traffic coming into and leaving a network. When the IDS spots something of interest, it reacts in the way defined by the intrusion analyst. An IDS knows what to look for and what to do about it because of sets of rules that define how it behaves. The IDS doesn't necessarily do anything to prevent anomalous traffic, rather it plays the role of an impartial observer.

Think of an IDS as sort of a "neighborhood watch"; it keeps a vigilant eye on your network neighborhood. If the IDS witnesses a burglar trying to break into the house down the street, it doesn't intervene; the IDS just calls the cops and lets the authorities do their job. Another analogy: As a kid, ever play "slug-bug" with a sibling while out on a family road-trip? You and your kid brother look out the car window, scanning the highway for Volkswagen Beetles ("bugs"). Whoever spots one first gets to "slug" the other. (For some reason, that was fun!) As the trip drags on, the rules can get more complex, maybe adding color; slug twice for a silver beetle, but never for a red one. If you slug your brother by mistake because you saw a red bug – he'd get to slug you back. So, in this little game, you were acting like a couple of little kIDS – you'd scan the traffic constantly and when you saw something matching the ruleset – a VW beetle – somebody got slugged. If someone got slugged because a red bug was seen – that's a "false positive". Then again, if a bug goes by and *neither* of you saw it, well that's a false negative. Moving right along…

There are a number of intrusion detection systems available. We'll be using well-known IDS "Snort", available from www.snort.org. There are a couple of reasons we'll use Snort: it's cheap and easy (cost-free and straightforward to implement). Plus, the GCIA course materials are very Snort-centric.

When it comes to describing Snort in a "nutshell", nobody does it better than the folks at www.snort.org, so let's get it right from the source, www.snort.org/about.html:

> Snort is an open source network intrusion detection system, capable of performing real-time traffic analysis and packet logging on IP networks. It can perform protocol analysis, content searching/matching and can be used to detect a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes, OS fingerprinting attempts, and much more.
>
> Snort uses a flexible rules language to describe traffic that it should collect or pass, as well as a detection engine that utilizes a modular plug-in architecture. Snort has a real-time alerting capability as well, incorporating alerting mechanisms for syslog, a user specified file, a UNIX socket, or WinPopup messages to Windows clients using Samba's smbclient.
> Snort has three primary uses. It can be used as a straight packet sniffer like tcpdump(1), a packet logger (useful for network traffic debugging, etc), or as a full blown network intrusion detection system. [3]

The above, along with being cheap and easy, means Snort will meet our needs nicely. Snort runs on a wide array of operating systems and hardware platforms, so detailing how to install and configure this IDS is well beyond the scope of this paper. There already exists a number of excellent sources for all you'd ever want to know about setting up Snort; www.snort.org is a great place to start.

Generally, a network will employ a firewall as part of its perimeter defense plan. An IDS can be added to augment the protection afforded by the firewall, often installed externally – between the firewall and the outside world. Below is a simplified diagram of one configuration, many others are possible.

[3] Snort.Org. "About Snort"

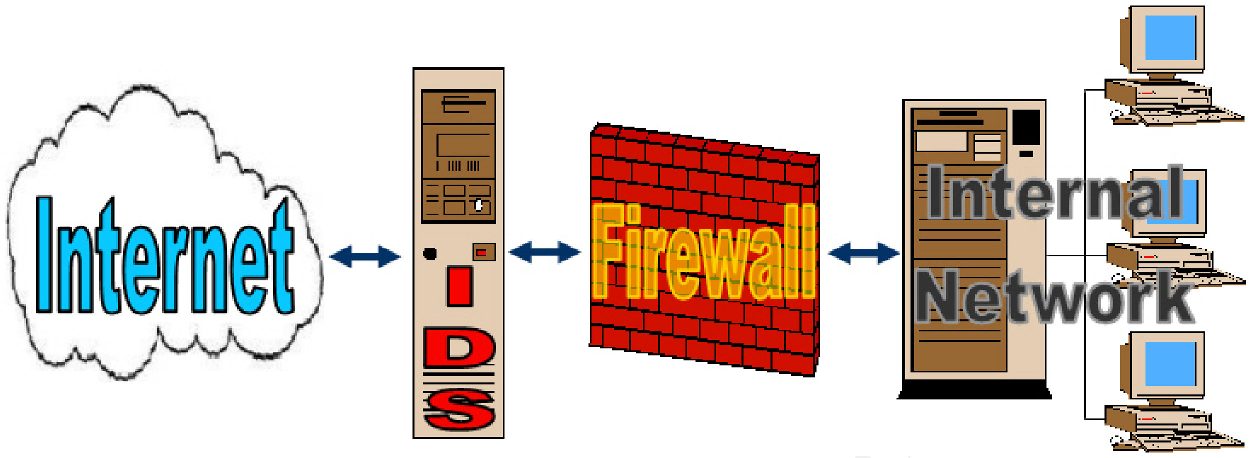


Figure 1: Simplified Network Diagram with IDS

As explained earlier, Snort uses sets of rules to let it know what it should be on the lookout for and what to do if something of interest is spotted. There are two Snort alert rules which address the vulnerability Slammer exploits, SID 2003 and SID 2004. The first of the two rules, SID 2003, is designed to keep Slammer from entering the network while the second, SID 2004, alerts if Slammer attempts to propagate beyond the internal network. Both of these are included in sql.rules, part of Snort's standard stable ruleset, and have been available from www.snort.org since late January 2003. These rules are shown below.

SID 2003:

```
alert udp $EXTERNAL any -> $INTERNAL 1434 (msg:"MS-SQL Worm propagation attempt"; content:"|04|"; depth:1; content:"|81 F1 03 01 04 9B 81 F1 01|"; content:"sock"; content:"send"; reference:bugtraq,5310; classtype:misc-attack; reference:bugtraq,5311; reference:url,vil.nai.com/vil/content/v_99992.htm; sid:2003; rev:2;)
```

SID 2004:

```
alert udp $INTERNAL any -> $EXTERNAL 1434 (msg:"MS-SQL Worm propagation attempt OUTBOUND"; content:"|04|"; depth:1; content:"|81 F1 03 01 04 9B 81 F1|"; content:"sock"; content:"send"; reference:bugtraq,5310; classtype:misc-attack; reference:bugtraq,5311; reference:url,vil.nai.com/vil/content/v_99992.htm; sid:2004; rev:1;)
```

If you're not used to looking at Snort rules, SID 2003 and SID 2004 may look like a bunch of gobbledygook. So, let's break down these two rules and see what makes them tick.

A Snort rule is made up of two parts: "header" and "options". The rule header is comprised of everything up to the first "("; the rule options are everything between the parentheses.

Let's have a closer look at the header for SID 2003.

```
alert udp $EXTERNAL any -> $INTERNAL 1434
```

"alert": First, we have the rule "type"; "alert" in this case. There are two other types of Snort rule types commonly used: pass and log. A "pass" rule would allow packets meeting the rule's constraints to pass without causing any action by the IDS. A rule of type "log" will simply log the traffic to a file if the rule's criteria are met. Here, the rule type of "alert" denotes this as an alert rule and an alert will be raised if all of the rule's conditions are met. The output of alerts is usually directed to a file in tcpdump format.

> "**udp**": this is the protocol we want to rule to trigger on. Other valid values would be "tcp", "icmp". This rule needs to specify "udp"; that's the protocol used by SQL Server Resolution Service (SSRS).
>
> "**$EXTERNAL**": This is the source IP address. In this case, we use "$EXTERNAL". "$EXTERNAL" is a variable declared in the Snort configuration file, snort.conf, and is often set to "!= $INTERNAL"; "$EXTERNAL" means hosts that are outside of our own network.
>
> "**any**": This is the source port; we need to alert on incoming traffic no matter what port it was sent from.
>
> "**->**": This delimits the source and destination information and also indicates the direction of traffic.
>
> "**$INTERNAL**": The destination IP address, here represented by "$INTERNAL", a variable declared in snort.conf that is set to the range of IP addresses used on your network.
>
> "**1434**": The destination port. SSRS uses port 1434, so we'll specify that here.

In short, this rule's header is telling Snort generate an alert for anything coming in on UPD port 1434 from outside our network.

Now for the rule options: The header is pretty general, the options get more specific. Here's the options header of SID 2003:

```
(msg:"MS-SQL Worm propagation attempt OUTBOUND"; content:"|04|"; depth:1;
content:"|81 F1 03 01 04 9B 81 F1|"; content:"sock"; content:"send";
reference:bugtraq,5310; classtype:misc-attack; reference:bugtraq,5311;
reference:url,vil.nai.com/vil/content/v_99992.htm; sid:2004; rev:1;)
```

Notice that the options must be within parentheses and each option must by closed by a semicolon (;) – including the last option. A valid Snort rule must always end with ";)".

"**msg**": "msg" and the text that follows up to the next semicolon is part of the message option. When an alert is raised, the text specified is inserted into the alert, letting the analyst know what the alert is about. For this rule, "MS-SQL Worm propagation attempt" will be included in every alert generated.

"**content**": The content option let's you specify anything you might be looking for in the payload. You can have more than one content option within a rule. The string you want to search for must be surrounded by double-quotes; when dealing with hexadecimal values, place a pipe character "|" on each side of the string, inside the double-quotes. Example: content:"abcd" searches for the ASCII string "abc"; content:"|abcd|" would search for hex. Here, we search for content that is consistent with Slammer.

"**reference**": Any source that can be helpful in getting more information about the rule, the vulnerabilities addressed, etc. Often, links to websites are listed here.

"**classtype**": The classification type as defined in the classification configuration file , configuration.conf, where they are defined and prioritized. Snort rules can also belong to a classtype. SID2003 belongs to the "misc-attack" classification.

"**sid**": This is the number assigned to the rule, the Snort identifier; Snort ID, or simply "SID". SIDs are assigned to rules by the Snort Team (www.snort.org).

"**rev**": This is the revision number of the rule, as assigned by the rule's creator and maintainers.

Now that we understand why SID 2003 works the way it does, we're well on our way to having a strong grasp of *any* Snort rule. We won't go into great detail to explain every piece of SID 2004 because it's very similar to SID 2003. The big difference is that SID 2004 is catching traffic that is exiting the network, so the source is HOME_NET and the destination is EXTERNAL_NET. The word "OUTBOUND" is added to the message text. The payload is a little different, so the content options account for this. The options for sid and rev are set appropriately. We'll be able to use the information covered about Snort rules throughout this practical assignment. More information on Snort rules can be found at http://www.snort.org/docs/writing_rules/ [4]

Armed with our knowledge of Snort rule basics, we're ready to take a look at examples of what a Slammer packet would look like in tcpdump format as well as the corresponding Snort alert that would have been triggered by the packet, SID 2003.

In the example of a tcpdump output, we can see that the protocol in use is UDP, the destination port is 1434, the first byte of the payload is 0x04, and 81 F1 03 01 04 9B 81 F1 01 (hex) is also part of the payload.

[4] Roesch, Martin and Green, Chris. "Writing Snort Rules".

```
11/11-12:34:56.123456 xxx.xxx.xxx.123:1234 -> xxx.xxx.xxx.456:1434
UDP TTL:110 TOS:0x0 ID:51082 IpLen:20 DgmLen:404 Len: 376
04 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 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 DC C9 B0 42 EB 0E 01 01 01 01 01 01 01 70 AE  ....B.........p.
42 01 70 AE 42 90 90 90 90 90 90 90 90 68 DC C9  B.p.B........h..
B0 42 B8 01 01 01 01 31 C9 B1 18 50 E2 FD 35 01  .B.....1...P..5.
01 01 05 50 89 E5 51 68 2E 64 6C 6C 68 65 6C 33  ...P..Qh.dllhel3
32 68 6B 65 72 6E 51 68 6F 75 6E 74 68 69 63 6B  2hkernQhounthick
43 68 47 65 74 54 66 B9 6C 6C 51 68 33 32 2E 64  ChGetTf.llQh32.d
68 77 73 32 5F 66 B9 65 74 51 68 73 6F 63 6B 66  hws2_f.etQhsockf
B9 74 6F 51 68 73 65 6E 64 BE 18 10 AE 42 8D 45  .toQhsend....B.E
D4 50 FF 16 50 8D 45 E0 50 8D 45 F0 50 FF 16 50  .P..P.E.P.E.P..P
BE 10 10 AE 42 8B 1E 8B 03 3D 55 8B EC 51 74 05  ....B....=U..Qt.
BE 1C 10 AE 42 FF 16 FF D0 31 C9 51 51 50 81 F1  ....B....1.QQP..
03 01 04 9B 81 F1 01 01 01 01 51 8D 45 CC 50 8B  ..........Q.E.P.
45 C0 50 FF 16 6A 11 6A 02 6A 02 FF D0 50 8D 45  E.P..j.j.j...P.E
C4 50 8B 45 C0 50 FF 16 89 C6 09 DB 81 F3 3C 61  .P.E.P........<a
D9 FF 8B 45 B4 8D 0C 40 8D 14 88 C1 E2 04 01 C2  ...E...@........
C1 E2 08 29 C2 8D 04 90 01 D8 89 45 B4 6A 10 8D  ...).......E.j..
45 B0 50 31 C9 51 66 81 F1 78 01 51 8D 45 03 50  E.P1.Qf..x.Q.E.P
8B 45 AC 50 FF D6 EB CA                          .E.P....
```

The Snort alert below, although it does not show the payload, includes the protocol of UDP along with the destination port of 1434.

```
[**] [1:2003:2] MS-SQL Worm propagation attempt [**]
[Classification: Misc. Attack] [Priority: 2]
11/11-12:34:56.123456 xxx.xxx.xxx.123:1234 -> xxx.xxx.xxx.456:1434
UDP TTL:107 TOS:0x0 ID:65475 IpLen:20 DgmLen:404 Len: 376
[Xref => http://vil.nai.com/vil/content/v_99992.htm][Xref =>
http://www.securityfocus.com/bid/5311][Xref =>
http://www.securityfocus.com/bid/5310]
```

A lot of material was covered in this section, “DETECTION”. However, the upshot is simple: if Snort is configured correctly and rules SID 2003 and SID 2004 are in-place, the presence of Slammer-related traffic will be detected.

CONTAINMENT AND REMOVAL: Slammer Unplugged. Actions to take in the event of a Slammer attack; stopping propagation and eradicating Slammer.

OK, so despite our best efforts, or because we've just been too darned lackadaisical, we find our poor network is being slammed with Slammer. We need to know what to do to keep our predicament from escalating further and how to clean it up.

Could-a, would-a, should-a… Now might be a good time to re-read the "PREVENTIVE MEASURES" section, because most everything we'll do to clear this up could have, and should have, been done earlier!

First thing's first. It's bad enough our own network's been infected; let's not exacerbate the situation by blasting lots of infectious traffic to the outside world. Worse case: we'll infect other networks. Best case: We'll launch a worldwide advertising campaign, telling everyone how laissez-faire we've been.

If you already know which of your hosts is running MS-SQL, it's a fair bet that's the machine that's fallen victim to Slammer. To bar the infected system from being party to Slammer's exercise in self-indulgence, you could just shut it down. Or, you could simply break the host's connection to the network. Disconnecting from the network, rather than powering down, would allow for whatever forensics may be possible; shutting down leaves no chance for that.

If, after the action you've taken thus far, the data storm persists, then something's running MS-SQL that you weren't aware of. I'll spare you the lecture on the importance of configuration management.

To keep Slammer from propagating beyond your network, and to prevent re-infection, you may need to go as far as temporarily breaking ties with the outside world; blocking all inbound and outbound traffic. However, this may be a bit heavy-handed; blocking UDP port 1434, inbound and outbound, at your network's perimeter should suffice. This ingress/egress filtering is also necessary for long-term operation, as noted in "PREVENTIVE MEASURES".

Removing Slammer is a cinch. Because this worm lives only within the host's memory, it will vanish when the machine loses power. However, if you leave the host connected to the network and power-up without first fixing whatever allowed it to become infected, then it will likely be re-infected in short order. Below is a recap of recommendations from the "PREVENTIVE MEASURES" section.

Prior to re-attaching the affected host to the network, do the following:

- Apply Service Pack 3a (SP3a) for MS-SQL or latest patches for MSDE 2000.
- Ensure the account used to run the MS-SQL service has limited permissions; a local system account is a good candidate.
- Set the MS-SQL service's start-up option to "Disabled" if SQL-Server functionality is not required.

The steps above must be taken for all affected hosts. Additional methods and tools for cleaning and protecting vulnerable hosts, as well as identifying them across your domain, can be found in “PREVENTIVE MEASURES”.

Lastly, if there’s a chance Slammer was successful in fully exploiting the vulnerability, gaining system-level access to a host, refer to the CERT Coordination Center’s “Steps for Recovering from a UNIX or NT System Compromise” to take immediate action. This document provides guidance on responding to a compromise and can be found at the link below.

http://www.cert.org/tech_tips/win-UNIX-system_compromise.html

CLOSING REMARKS: Wrapping-up.

As Slammer’s blizzard of January 2003 subsided, and system administrators began the task of digging out, many may have thought the damage was as bad as it gets. However, it well could have been worse, much worse. This vulnerability had been identified half a year before Slammer hit.

Another, potentially more dangerous, exploit was published by David Litchfield in his July 25th 2002 advisory for NGSSoftware Insight Security Research, #NISR25072002. [5] His exploit provides unauthenticated system-level access to a compromised host and he assigned, fittingly, a severity rating of critical. Litchfield’s actual program code, which he named "Advanced Windows Shellcode", was posted at many websites. While Slammer spreads itself very noisily, Litchfield’s exploit is quite restrained; it supports spoofing of the source IP address and returns a command prompt to the attacker. Never a good thing!

An example, modified by “lion” and renamed “MSSQL2000 Remote UDP Exploit!”, can be found at the link below.

http://www.security-corporation.com/exploits-20030822-000.html

Perhaps, in retrospect, Slammer was a mixed blessing. Although it did cause its share of headaches, it also served as a forewarning; giving us all a chance to “get our act together”, before a more brutal assault could be set into motion.

Got Slammer? If so, Carnegie Mellon University’s CERT Coordination Center Software Engineering Institute would like to hear about it. Refer to CERT Advisory CA-2003-04 for contact and other information about Slammer. This advisory can be found here: http://www.cert.org/advisories/CA-2003-04.html. [6]

[5] Litchfield, David. “Advanced Windows Shellcode”
[6] CERT.ORG. “CA-2003-04”.

In closing, here's a point that bears repeating: Everything covered for how to get rid of Slammer is what should have been done to keep you from getting it in the first place. Due diligence and following best practices can go a long way in keeping your network Slammer-free.

PART TWO: Network Detects; in-depth analysis of three events of interest.

OVERVIEW

In Part 2, three different network detects are presented with in-depth analysis. To help reinforce the material covered, each detect is followed by a multiple choice test question. My analysis for the first detect, "SID 1322 BAD-TRAFFIC bad frag bits", was submitted to the incidents.org intrusions forum to request constructive criticism from other security professionals. The feedback received, along with my response, is found at the end of Detect 3.

Information applying to all three detects was consolidated and is presented next, to avoid redundancy within the actual detect analysis.

1. Source of Trace:

The source for all three traces is the file 2002.10.15 from http://www.incidents.org/logs/raw/. This is a tcpdump binary file. Note: The file's name indicates it's from 15 October 2002, although the date next to the file on the website is Dec 2, 2002. However, the dates for the alerts within the file are October 15. The file size was approximately 3MB.

2. Detect was generated by:

The detects were generated using the Win32 version of snort 2.0.4 running on a Windows XP Pro workstation. The standard stable snort ruleset, which was posted along with snort 2.0.4 was used along with Politecnico di Torino's Winpcap 3.0.0.18. Snort and the ruleset were downloaded from www.snort.org; winpcap was downloaded from winpcap.polito.it.

The snort command below was issued to generate the alerts (full directory paths have been removed for readability):

```
snort -r 2002.10.15 -c snort.conf –l 2002.10.15.snort -d -X -v
```

The options selected for snort (-r, -c, -d, -v, and -X), are explained below:

-r: Tells snort we want to read-in a file in tcpdump format and process it against snort's ruleset. This option is followed by the path and name of the tcpdump file we wish to analyze. In this case, we used "2002.10.15" (our tcpdump file's name).

-c: Lets snort know the name and location of the snort configuration file we want to use; “snort.conf” in this case.

-l: This is the directory we want snort to export the data it analyzes, “2002.10.15.snort”. Without this option, the processed data is streamed only to the console; only a good idea if you can read really fast and have a great memory!

-d: Use this option to include the application layer in the output.

-X: Similar to “-d”, but dumps the link layer.

-v: Provides a more verbose output.

Running snort with no options selected simply displays the version number, guidance on usage, and a list of options. Alternately, you can use the -? option to display all of this help information. The “-V” option displays the version only.

Note: The section “ANALYSIS PROCESS AND LESSONS LEARNED” of Part 3 (“Analyze This!”), includes explanations on the methods used to prepare the log files for analysis. Here, we’ll just get right to the detects.

Severity for each detect was ranked on a scale from 1 to 5, 5 being the highest. The formula below was used to calculate Severity:

Severity = (Criticality + Lethality) - (System countermeasures + Network countermeasures)

DETECT 1: SID 1322 BAD-TRAFFIC bad frag bits

Below are all of the alerts relating to the events of interest for Detect 1, only two:

```
[**] BAD-TRAFFIC bad frag bits [**]
11/15-09:01:40.376507 213.107.102.7 -> 170.129.168.122
TCP TTL:111 TOS:0x0 ID:22346 IpLen:20 DgmLen:1468 DF MF
Frag Offset: 0x0000   Frag Size: 0x05A8

[**] BAD-TRAFFIC bad frag bits [**]
11/15-09:01:43.366507 213.107.102.7 -> 170.129.168.122
TCP TTL:111 TOS:0x0 ID:22490 IpLen:20 DgmLen:1468 DF MF
Frag Offset: 0x0000   Frag Size: 0x05A8
```

Items of special interest for this detect are the fragmentation-related flag **DF** and **MF**, shown in **bold** for readability. **DF** means “don’t fragment”, **MF** means “more fragments”. This will be explained in more detail later.

1. Source of Trace: Please see introduction to Part 2.

2. Detect was generated by: Please see introduction to Part 2.

3. Probability the source address was spoofed:

It's unlikely the source IP has been spoofed. There are a couple of hits to/from the same hosts in close succession; not what you'd usually see in the case of spoofing. The IP IDs are 22346 and 22490, I'd expect these to be much closer if spoofing were happening. Finally, if the goal is reconnaissance, then spoofing wouldn't serve the attacker's needs; they'd get no response if they were spoofing the source.

A whois query on both www.geektools.com and www.ripe.net for source IP address 62.253.181.11 offers the following:

```
inetnum:     62.253.180.0     -
62.253.183.255
netname: NTL
descr: NTL Internet
descr: Renfrew site
country: GB
admin-c: NNMC1-RIPE
tech-c: NNMC1-RIPE
status: ASSIGNED PA
mnt-by: AS5089-MNT
changed:
hostmaster@ntli.net
20010330
changed:
hostmaster@ntli.net
20020815
source: RIPE

route: 62.252.0.0/14
descr: NTL-UK-IP-BLOCK-3
origin: AS5089
mnt-by: AS5089-MNT
changed:          bob.procter@ntli.net
20010205
source: RIPE
role:   NTLI   Network   Management
Centre
address: NTL Internet
address: Crawley Court
address: Winchester
address: Hampshire
address: SO21 2QA
```

4. Description of attack:

The snort rule that generated the alerts is shown below and is located within bad-traffic.rules, v 1.22 2003/07/18 15:19:16.

```
alert ip $EXTERNAL_NET any -> $HOME_NET any (msg:"BAD-TRAFFIC bad
frag bits"; fragbits:MD; sid:1322; classtype:misc-activity; rev:5;)
```

These events triggered the alert above because both the DF (Don't Fragment) and MF (More Fragments) options were set. These two options would never, under normal circumstances, be used together as they as are intended to be mutually exclusive. The DF flag indicates that the packet should not be fragmented (all data should be sent as a single unit). However, the MF flag means more packets are to follow.

Barring the possibility that this packet was simply corrupted in transit (perhaps by a misconfigured network device), it seems reasonable to assume this packet was crafted and possibly sent for the purpose of reconnaissance. Certainly, this is not the sort of activity one wishes to see on their network and could be a prelude an attack. The sender of this crafted packet, "the attacker", could be expecting to receive an ICMP error message in return ("fragmentation needed but DF set"). The attacker could use the information gathered to piece together the targeted host's configuration, as well as the network's topology, for a future attack.

Having used the snort option "-X", the payload of the traffic causing the alert was dumped (hex and ASCII) to a file under a directory named for the source IP. Examining the actual payload of the first packet reveals evidence of the CodeRed-like exploit. Here's an excerpt:

```
GET /default.i
da?NNNNNN
NNNNNNNN

…

8%u0000%u00=a  H
TTP/1.0..Content
-type: text/xml.
HOST:www.worm.co
m. Accept: */*.C
ontent-length: 3
```

The presence of "GET /a.ida" should have, I thought, triggered snort rule SID 1243 "WEB-IIS ISAPI .ida attempt" (from web-iis.rules). The rule SID 1243 is shown below.

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-IIS ISAPI .ida attempt"; flow:to_server,established; uricontent:".ida?"; nocase; reference:arachnids,552; classtype:web-application-attack; reference:bugtraq,1065; reference:cve,CAN-2000-0071; sid:1243; rev:8;)

At first, it wasn't clear to me why the rule above didn't trigger an alert. Then, I recalled that Snort uses a "first match" policy when processing alerts. Snort's rulesets are processed in the order they are listed in the "include" section of the snort.conf file. Snort processes all of the rules within the first ruleset before moving on to the next ruleset. If any matches are found in the first ruleset, then snort triggers the appropriate alert and ceases further processing of all subsequent rulesets.

In the case of this detect, SID 1322 ("BAD-TRAFFIC bad frag bits") is part of the bad-traffic.rules ruleset while SID 1243 ("WEB-IIS ISAPI .ida attempt") belongs to web-iis.rules. In the snort.conf used for this detect, bad-traffic.rules is right at the top of the list; web-iis.rules is much further down the line. So, based on how Snort processes rulesets using "first match" and the order of the rulesets listed in snort.conf, it's easy to see why we didn't see an alert for CodeRed.

It is possible the attacker crafted the packet with the DF and MF flags set in order to call attention away from the Code Red payload, allowing it to slip by the IDS. If that's the case, then perhaps reconnaissance wasn't part of the plan after all. Also of note, there was no other traffic seen from this source in the log file analyzed.

5. Attack mechanism:

CodeRed is a worm that exploits a buffer overflow vulnerability in Microsoft Internet Information Services (IIS). This vulnerability is well known and is documented in CVE-2001-0500, version: 20030402. If exploiting this vulnerability is successful, then the attacker would be permitted to execute arbitrary commands with system-level privileges.

On the infected host, the worm creates multiple threads of itself. The initial thread checks to see if the infected host is configured with a language setting of American English; if so, then webpage defacement is attempted. If any other language setting is used, then the worm attempts to infect other hosts on the network; during the first two weeks of any given month. A Denial of Service (DoS) attempt will be made against www.whitehouse.gov's web service (198.137.240.91), if the day of the month falls between the 20th and 28th. During the final week of the month, the worm goes into a "sleep-state".

The additional threads spun off by the first will attempt to infect other hosts using a pseudo-random list of IP addresses. As additional hosts become infected, a DoS can result.

Although I can't state with absolute certainty that this is a CodeRed attack, it does seem to meet the criteria; "...if it walks like a duck, quacks like a duck...". Note the presence of "... GET /default.ida?NNNNNN ..." and "... www.worm.com ..." in the first packet's payload. So, if this is not CodeRed, it is surely an attempted CodeRed-like exploit of the IIS overflow vulnerability.

The actual alert for this detect (SID 1322 "BAD-TRAFFIC bad frag bits"), as stated earlier, could indicate reconnaissance activity on the part of the attacker. If a large number of these crafted packets had been sent, perhaps a DoS could have resulted; the router on the targeted network may have stalled while waiting for additional packets that would never arrive. However, that didn't occur in this case; there were only two "hits". Therefore, it would seem the attacker sent the crafted packets hoping to receive an ICMP message ("fragmentation needed but DF set"), revealing information about the host and the network on which it resides.

With the above in mind, it seems possible that this attack was two-fold: attempt to exploit the IIS buffer overrun vulnerability; if that fails, at least some useful information may be returned.

6. Correlations:

In regards to CodeRed, this worm is quite famous and I found no shortage of information about it on the Internet including:

Mitre's Common Vulnerabilities and Exposures (CVE):
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0500

Carnegie Mellon University's CERT Coordination Center:
http://www.cert.org/incident_notes/IN-2001-08.html

Network Associates Technology's Virus Information Library:
http://vil.nai.com/vil/content/v_99142.htm

As for the issue of the setting of both MF and DF is each of the examined packets, I was not able to find anything concrete with which to correlate nor could I find any reference to the use of this sort of crafted packet to evade an Intrusion Detection system (IDS) in order to pass a packet containing a CodeRed-like payload.

7. Evidence of active targeting:

There were only two instances of this alert (SID 1322 "BAD-TRAFFIC bad frag bits"). These alerts were caused due to packet crafting. CodeRed is known to self-propagate to targets based on a randomized list of IP addresses. Therefore, I believe this is a case of active targeting.

8. Severity: Assessing the detect’s seriousness.

Criticality: 5

There isn't enough information in the logs to determine the importance of the targeted host. However, because the attacker went directly to this host and, presumably, only this host, I'll assume the attacker knows something I don't about the target. Also, it seems the attacker attempted to evade the NIDS by sending crafted packets. Finally, any attack that, if successful, could grant system-level permissions to an intruder should be taken very seriously.

Lethality: 5

If the targeted host is running Microsoft Internet Information Services (IIS) and its service packs are very out of date, a DoS could result; both due to a spike in network traffic and website defacement. Because an apparent attempt was made to evade the IDS, a high rating for Lethality is warranted.

System countermeasures: 3

There doesn't seem to be enough information to determine this with any degree of accuracy; the crafted packets were never delivered to the intended host, no response was seen from the target. So, the target never had the opportunity to demonstrate if its guard was up.

Network countermeasures: 3

Although the "wrong" alert was generated for this event, the alert that did fire did lead us to the problem. The network countermeasures that were in place appear to have done their job, but only adequately.

Restating our formula for Severity:

Severity = (Criticality + Lethality) - (System countermeasures + Network countermeasures)

Severity = (5 + 5) - (3 + 3) = 4

With a Severity of 4, this does constitute an EOI (event of interest).

9. Defensive recommendation:

System defense:

- Disable web services if not required
- If web service is required, make sure security best practices are followed
- Ensure current service packs are applied

Network defense:

Intrusion Detection System (IDS)

- Ensure IDS is configured properly
- Use Snort's fragmentation and stream reassembly preprocessors to help prevent malicious packet fragments from entering the network.
- Keep signatures current and fine-tune as required
- Monitor generated alerts
- Investigate and take appropriate action as necessary.

Firewall, router, etc

- Setup a firewall to stop dangerous traffic from entering the network.
- Include filters to prevent traffic with invalid fragment flags from entering network

10. Multiple choice test question:

The fragment options MF and DF could be set within the same packet under the following circumstances:

a) The MF (Maintain Fragments) and DF (Defend Fragments) options are the result of a crafted packet.

b) The MF option must always be set if the DF option is used; otherwise the packet could cause a Denial of Service.

c) It is impossible for both the MF and DF options to be set because they occupy the same space within the IP header.

d) Packet crafting is the most likely reason the MF (More Fragments) and DF (Don't Fragment) options would both be set.

Answer: d

There’s no such thing as “MF (Maintain Fragments)” and “DF (Defend Fragments)”. The DF option does not require MF to be set; no DoS would occur. The MF and DF options do not occupy the same space in the IP header. If the MF and DF flags are both set, this is probably due to packet crafting.

COMMUNITY FEEDBACK: Postings to incidents.org’s intrusions mailing list.

On November 17, 2003, I posted my analysis of this detect and solicited feedback from the GCIA community. The Subject of my posting is “LOGS: GCIAC GCIA Version 3.4 Practical Detect #1 - Eric Evans”.

I received a prompt reply from Joe Bowling; he’s newly-certified and is already doing his part to give back to the community. I’ve included three questions posed by Joe, along with my answers as well as an excerpt from my analysis that prompted Mr. Bowling’s questions. My response to Joe’s questions have been posted to intrusions@incidents.org.

Question #1:

Stimulus for the question: “…The sender of this crafted packet, "the attacker", could be expecting to receive an ICMP error message in return …”

Question: “Which ICMP error message would the attacker be hoping to see in response to the above packets?”

My answer: “I believe the message returned would have been "fragmentation needed but DF set". This would be good for the reader to know, I’ve added it to my analysis of this detect.”

Question #2:

Stimulus for the question: “… Examining the actual payload of the first packet reveals evidence of the CodeRed-like exploit. Here's an excerpt: …”

Question: “How did you find the payload?”

Answer: “When running Snort, I used the -X option to dump the payload (as hex and ASCII) to a file that I could examine. I explained these things earlier in my analysis. Because of your question, I thought it would be good to remind the reader in the ‘Description of attack’ section.”

Question #3”

Stimulus for the question: “… Lethality: 5 …”

Question: “If you think this was a recon attempt do you think that the lethality is a bit high?

Answer: “That’s a good point. A simple probe wouldn’t be too worrisome. However, if the MF and DF flags were both set as a means to try to get around the NIDS, then I think this should be treated more seriously than a scan. I suppose I could be talked out of it, but for now I’ll stick with a high mark for lethality. I added a sentence to my analysis to help clarify my reasoning.

DETECT 2: SID 524 BAD-TRAFFIC tcp port 0 traffic

Here are the alerts relating to the events of interest for detect 2. There were actually 14 alerts, I’m only providing the first four here as the remaining alerts are quite similar.

```
[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/15-20:08:17.016507 211.47.255.23:46919 -> 170.129.23.96:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x83442FEE  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/15-20:08:19.996507 211.47.255.23:46919 -> 170.129.23.96:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x83442FEE  Ack: 0x0  Win: 0x16D0  TcpLen: 32
```

```
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/15-20:08:25.996507 211.47.255.23:46919 -> 170.129.23.96:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x83442FEE  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/15-20:08:37.996507 211.47.255.23:46919 -> 170.129.23.96:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x83442FEE  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
```

Items of special interest for this detect are the destination port and IP ID of zero (**0**), shown in **bold** for readability. The significance with be discussed later.

1. Source of Trace: Please see introduction to Part 2.

2. Detect was generated by: Please see introduction to Part 2.

3. Probability the source address was spoofed:

In the case of a SYN flood attack, the SYN flag would be set, as it is here, and we would suspect the source may be spoofed. However, other than the SYN flag, there's nothing here to lead me to think this is a SYN flood; I'd expect to see much more traffic. Here, what we're seeing is different; there's an initial packet followed by another 3 seconds later, a third at 6 seconds after the second, finally a fourth at 12 seconds behind the third. This seems more like retries, not something seen in a SYN flood.

Mostly what is of interest here, and the reason the alert was raised, is because the destination port is 0. Port 0 isn't normally used as it is designated as reserved. This part of these packets must be crafted. It's possible that the tool used to generate this traffic was hping2; described later. hping2 is capable of crafting packets and setting the destination port as the user likes. Also, hping2 allows the source IP to be crafted. So, if hping2 is in use in this case, then the attacker could be spoofing the IP. If the purpose of sending the packets to port 0 is to elicit a response from the receiving host for reconnaissance, then the attacker wouldn't be spoofing the source IP. Although there are ways to still see the responses if the source is spoofed; this will be discussed later.

So, from the information we have, it isn't possible to determine, without doubt, if the source is spoofed. If we knew the attacker's intent, then we could better speculate. My hunch is that the source is not spoofed.

A whois query on www.apnic.net, via www.geektools.com, offers the following:

```
inetnum:     211.42.0.0 - 211.51.255.255
netname:     KRNIC-KR
descr:       KRNIC
descr:       Korea Network Information Center
country:     KR
admin-c:     HM127-AP
tech-c:      HM127-AP
remarks:     ******************************************
remarks:     KRNIC is the National Internet
Registry
remarks:     in Korea under APNIC. If you would
like to
remarks:     find assignment information in detail
remarks:     please refer to the KRNIC Whois DB
remarks:
http://whois.nic.or.kr/english/index.html
remarks:     ******************************************
mnt-by:      APNIC-HM
mnt-lower:   MNT-KRNIC-AP
changed:     hostmaster@apnic.net 19991118
changed:     hostmaster@apnic.net
20010606
status:      ALLOCATED PORTABLE
source:      APNIC
person:      Host Master
address:     11F, KTF B/D, 1321-11,
Seocho2-Dong, Seocho-Gu,
address:     Seoul, Korea, 137-857
country:     KR
phone:       +82-2-2186-4500
fax-no:      +82-2-2186-4496
e-mail:      hostmaster@nic.or.kr
nic-hdl:     HM127-AP
mnt-by:      MNT-KRNIC-AP
changed:     hostmaster@nic.or.kr
20020507
source:      APNIC
```

4. Description of attack:

The snort rule that generated the alerts is shown below and is located within bad-traffic.rules, v 1.22 2003/07/18 15:19:16.

```
alert tcp $EXTERNAL_NET any <> $HOME_NET 0 (msg:"BAD-TRAFFIC tcp
port 0 traffic"; classtype:misc-activity; sid:524; rev:6;)
```

There are a total of 14 packets sent from the source host (211.47.255.23) to a single destination host (170.129.23.96). In all cases, the ID is set to 0 and the DF flag is set. The packets are in groups of three or four and are sent in intervals of 3, 6, and 12 seconds. After the last retry, the source port is changed, always to a higher number, and in the range above 1024 (the ephemeral ports). For all packets, the payload is identical, as shown below:

```
.....3....&...E.
.4..@....../....
.`.G...D/.......
..Y.............
..
```

Here's a list of the basic info for these packets, for reference:

Date	Time	SrcIP	SrcPrt	DstIP	DstPrt	Seq
10/15/2002	20:08:17.016507	211.47.255.23	46919	170.129.23.96	0	0x83442FEE
10/15/2002	20:08:19.996507	211.47.255.23	46919	170.129.23.96	0	0x83442FEE
10/15/2002	20:08:25.996507	211.47.255.23	46919	170.129.23.96	0	0x83442FEE
10/15/2002	20:08:37.996507	211.47.255.23	46919	170.129.23.96	0	0x83442FEE

```
10/15/2002     20:08:49.016507     211.47.255.23  47154  170.129.23.96  0     0x858405A9
10/15/2002     20:08:52.016507     211.47.255.23  47154  170.129.23.96  0     0x858405A9
10/15/2002     20:09:10.016507     211.47.255.23  47154  170.129.23.96  0     0x858405A9
10/15/2002     20:09:21.006507     211.47.255.23  47382  170.129.23.96  0     0x86DAFDF0
10/15/2002     20:09:23.996507     211.47.255.23  47382  170.129.23.96  0     0x86DAFDF0
10/15/2002     20:09:29.986507     211.47.255.23  47382  170.129.23.96  0     0x86DAFDF0
10/15/2002     20:09:42.006507     211.47.255.23  47382  170.129.23.96  0     0x86DAFDF0
10/15/2002     20:09:56.016507     211.47.255.23  47620  170.129.23.96  0     0x88C07AEB
10/15/2002     20:10:02.006507     211.47.255.23  47620  170.129.23.96  0     0x88C07AEB
10/15/2002     20:10:14.016507     211.47.255.23  47620  170.129.23.96  0     0x88C07AEB
```

A point of interest is that all 14 packets are identical, with the exception of timestamp and the source port and sequence numbers that increment with each group of packets. Especially suspect is the ID: it's the same for all of these packets (0), this is not normal and, along with the destination port of 0, indicates these are crafted packets.

The attacker is probably just probing the destination to see if it responds, to confirm it's alive. There isn't sufficient information in the logs provided by incidents.org to allow us to determine if a response was sent back to the source. However, because what we're seeing appears to be retries from the source, I'll assume the destination host did not reply. If the host did send a response to the attacker, then it could be setting itself up as a target for a future attack; arming the attacker with the target's operating system fingerprint and helping them narrow down the list of exploits they may wish to employ.

5. Attack mechanism:

What seems to be going on here is operating system (OS) fingerprinting. Crafted packets are being sent to a non-standard port (port 0) in order to elicit a response (RESET). Each OS will reply in differently to such traffic and the way the host responds gives the attacker clues about what the OS might be.

I already knew that port 0 is a reserved port. I did discover something new about port 0 here: http://compnetworking.about.com/library/ports/blports_0.htm. According to this site, specifying port 0 in a UNIX program simply tells the OS to use the next dynamic port. [7] So, perhaps the attacker is using a UNIX-based system, or perhaps Linux. However, it seems to me that if port 0 were specified in a UNIX program, barring any bugs in the code, a valid port would have been selected.

I learned something else about port 0 from Jason Thompson, recently GCIA certified. Although his practical has not yet been posted to sans.org, Jason did write in his detect posting that port 0 is a signature for the Linux operating system. Mr. Thompson's message can be found here: http://cert.uni-stuttgart.de/archive/intrusions/2003/07/msg00183.html. [8]

[7] COMPNETWORKING. "Port 0".
[8] Thompson, Jason, "LOGS: GIAC GCIA Version 3.3 Practical Detect - Repost"

My assumption is that the attacker is using hping2, or a similar tool such as Back Orifice, to craft and send these packets. hping2 has the functionality required to craft ports, TTL, set TCP flags, etc. It can also send fragments and spoof the source IP.

However, I was unable to confirm in the hping manpage if hping2 will allow setting the IP ID to zero, as is the case with this traffic. According to the manpage: the "-N" option can be used to set the IP ID field. If the IP ID is not specified, then a random IP ID is used. [9] The manpage does not state that port 0 cannot be used. As I am not running an operating system compatible with hping, I was not able to prove whether or not hping can set the IP ID to zero. The hping manpage is located here: http://www.hping.org/manpage.html.

Fragments and packets sent to port 0 have a greater likelihood of making their way through a network's perimeter. Additionally, hping2 can enlist another host to assist with its reconnaissance. The attacker will choose as its helper a "silent host" that is running an older operating system that generates IP IDs that can be fairly easily predicted. The attacker spoofs the address of the silent host, the target replies to the silent host, and the attacker just listens in. Other tools are available, which may be able to accomplish what I suspect hping2 is being used for. hping2 is available for download from http://www.hping.org/download.html. [10]

6. Correlations:

SANS GIAC GCIA (Intrusion Detection In-Depth) course materials (Available only to current GCIA students.):

http://giactc.giac.org/cgi-bin/momgate/

ID_23_1102.pdf - ID_25_1102.pdf, ID_42_1102.pdf, and ID_410_1102.pdf

Ewen Fung's GCIA Practical Detect posting, now available here:

GIAC GCIA Version 3.2 Network Detect #3
cert.uni-stuttgart.de/archive/intrusions/ 2002/09/msg00012.html

Information about the Linux OS signature as well as confirmation of by Back Orifice theory came from Jason Thompson, http://cert.uni-stuttgart.de/archive/intrusions/2003/07/msg00183.html.

7. Evidence of active targeting:

This does not seem to be a SYN flood or any other type of scan, unless it is a "low and slow" scan. The attacking source went directly to the destination host; no other traffic was seen from the source. With this in mind, I believe active targeting is taking place.

8. Severity: Assessing the detect's seriousness.

[9] Hping manpages.
[10] Hping download.

Criticality: 3

Because we don't have enough information to determine the role of the targeted system, we'll assign a criticality value of 3.

Lethality: 1

The attacker seems to be just probing and perhaps OS finger printing, so we'll assign the lowest value to lethality (1).

System countermeasures: 3

There's no evidence the host replied to the attacker, however, we know the logs are incomplete. Again, we don't have enough information about the host to know if its OS patches are current, etc. We'll assign a medium value of 3 for system countermeasures.

Network countermeasures: 3

We do know an intrusion detection system is in-place and it is generating alerts for this sort of traffic as it should. However, there's either no firewall in place between the NIDS and the outside world or the firewall simply isn't filtering traffic to/from port 0, although there may be a firewall located behind the NIDS. We'll assign 3 for network countermeasures.

Restating our formula for Severity:

Severity = (Criticality + Lethality) - (System countermeasures + Network countermeasures)

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

With a Severity of –2 (negative two), this does not constitute an event of very high interest.

9. Defensive recommendation:

System countermeasures:

System defense:

- Determine if host is running Linux. If so, traffic on port 0 may not be out of place. Of course, an IP ID of 0 is not normal.

- Ensure patches for all hosts on the network are current.

Network defense:

Intrusion Detection System (IDS)

- Setup an IDS
- Keep signatures current and fine-tune as required
- Monitor generated alerts
- Investigate and take appropriate action as necessary.

Firewall, router, etc

- Setup a firewall between the NIDS and the outside world to stop dangerous traffic from entering the network, if one is not already present.
- Include filters to prevent unwanted traffic from entering network; drop all port 0 traffic.

10. Multiple choice test question:

```
[**] BAD-TRAFFIC tcp port 0 traffic [**]
11/15-20:08:17.016507 211.47.255.23:46919 -> 170.129.23.96:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq: 0x83442FEE  Ack: 0x0  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
0x0000: 00 00 0C 04 B2 33 00 03 E3 D9 26 C0 08 00 45 00  .....3....&...E.
0x0010: 00 34 00 00 40 00 2E 06 B8 9B D3 2F FF 17 AA 81  .4..@....../....
0x0020: 17 60 B7 47 00 00 83 44 2F EE 00 00 00 00 80 02  .`.G...D/.......
0x0030: 16 D0 59 A5 00 00 02 04 05 B4 01 01 04 02 01 03  ..Y.............
0x0040: 03 00
```

In regards to the information above, which of the conclusions below can be made with certainly?

a) The source IP address is spoofed using hping2.

b) This is a SYN flood attack.

c) The destination port is a reserved port.

d) The IpLen, DgmLen, and TcpLen are crafted.

Answer: c

a) It is not possible to know from the information given if hping2 was used. B) We're only shown a single packet, so we can't determine if this is part of a SYN flood attack. d) IpLen, DgLen, and TcpLen are all reasonable values and correspond with one another. The destination port is 0; port 0 is a reserved for both source and destination traffic.

DETECT 3: SID 527 BAD-TRAFFIC same SRC/DST

For this detect, all alerts show identical IP addresses (as source and destination) within each alert; 11 IP addresses were paired across the alerts. These alerts use the IGMP protocol and have the same timestamp, 04:36:45.306507. The MAC addresses are consistent throughout the 11 alerts and indicate Cisco routers. Because these are designated Cisco router MAC addresses [11], our IDS must be located between them.

To see if there was a trend for this traffic, and not just a single-shot burst, I downloaded log files for five days before and three days after this event. There were no instances of this alert in the five days leading up to this detect. On the following day and two days after, there were 12 instances each of this alert, all very similar to the detect under scrutiny; although the events didn't occur at the same time each day and the group of IP addresses differed on each occasion.

Here are the alerts relating to the events of interest for detect 3. There were actually 11 alerts, I'm only listing the first three here in the interest of brevity.

```
[**] [1:527:4] BAD-TRAFFIC same SRC/DST [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
11/15-03:36:45.306507 170.129.215.99 -> 170.129.215.99
PROTO002 TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:28
[Xref => http://www.cert.org/advisories/CA-1997-28.html][Xref =>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0016]

[**] [1:527:4] BAD-TRAFFIC same SRC/DST [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
11/15-03:36:45.306507 170.129.215.104 -> 170.129.215.104
PROTO002 TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:28
[Xref => http://www.cert.org/advisories/CA-1997-28.html][Xref =>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0016]

[**] [1:527:4] BAD-TRAFFIC same SRC/DST [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
11/15-03:36:45.306507 170.129.215.115 -> 170.129.215.115
PROTO002 TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:28
[Xref => http://www.cert.org/advisories/CA-1997-28.html][Xref =>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0016]
```

Above, there are numerous things catching our attention, shown in **bold** for readability.

[11] Cisco Systems. "How CGMP Leave Processing Functions".

Among the items of concern are the matching source and destination IP addresses, the use of the IGMP protocol (PROTO002), identical timestamps on all packets, and a high setting for TTL. What these things mean will be talked about later

1. Source of Trace: Please see introduction to Part 2.

2. Detect was generated by: Please see introduction to Part 2.

3. Probability the source address was spoofed:

The source and destination IP addresses should never be the same. Therefore, something's gone wrong here to cause this; either maliciously or inadvertently. I'm leaning toward the former.

The alerts indicate the protocol used is "PROTO002". PROTO002 (protocol 2) is IGMP (Internet Group Management Protocol). IGMP is the protocol for IP multicasting. Details about IGMP can be found in RFC 1112 and RFC 2236. Network Sorcery also has some good information about IGMP on their website (http://www.networksorcery.com/enp/protocol/igmp.htm). [12]

Also of note is that all of the source MAC addresses are the same; the destination MAC addresses also match. Additionally, the timestamps for all of these alerts are the identical. If this is legit multicasting, then it seems reasonable that the timestamps are all the same and perhaps the source/destination IP address mix-up we're seeing is due to a misconfigured router. However, that wouldn't explain the matching MAC addresses; the source MAC addresses may be the same, but not the destination.

To get a bit more info, I decided to use windump (the Windows OS-compatible version of tcpdump). Here's the command I used:

```
windump -n -X -vv -e -r 2002.10.15 > 2002_10_17_windump_output.txt
```

The options used above are as follows:

-n: host address and port numbers don't get translated to host names and port names.
-X: Output hex data in addition to ASCII
-vv: Very verbose mode
-e: Provides the link-level header
-r: Tells windump which file to analyze.

Here's an excerpt of the windump output:

```
    04:36:45.306507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 0800 60: IP (tos 0x0, ttl 47, id 0,
    len 28) 170.129.215.93 > 170.129.215.93: igmp query v2 [gaddr 240.0.1.197]
```

[12] Network Sorcery, “IGMP, Internet Group Management Protocol”.

```
0x0000    4500 001c 0000 0000 2f02 8822 aa81 d75d    E......./.."...]
0x0010    aa81 d75d 1164 fcd5 f000 01c5 0000 0000    ...].d..........
0x0020    0000 0000 0000 0000 0000 0000 0000         ..............
```

A couple of things caught my attention: 1) the TTL value of 47, according to RFC 2236, the TTL value for IGMP traffic must be 1. 2) The "gaddr" (group address) is 240.0.1.197. According to Cisco, the designated range for multicast addresses is 224.0.0.1 - 239.255.255.255; 240.0.1.197 doesn't fit in. Cisco also says that the valid range for MAC addresses with IGMP is 01-00-5E-00-00-00 and 01-00-5E-00-00-FF; the MAC addresses in our trace don't fall into this range either. So, we have IGMP packets that are not valid.

Based on the above, I believe the source IP was spoofed and other portions of the packets were crafted.

4. Description of attack:

The snort rule that triggered this alert is located within bad-traffic.rules, v 1.22 2003/07/18 15:19:16. The alert is shown below.

```
alert ip any any -> any any (msg:"BAD-TRAFFIC same SRC/DST"; sameip;
reference:cve,CVE-1999-0016; reference:url,www.cert.org/advisories/CA-1997-
28.html; classtype:bad-unknown; sid:527; rev:4;)
```

Initially, this appears to be an at attempt at some sort of loopback Denial of Service (DoS) attack; whereby the attacked host bounces traffic to itself in an endless loop; like a dog chasing its own tail. A Land attack was my first thought because it involves spoofed source IPs that are the same as the destination as well as the same ports for source and destination. Land (land.c) attacks work a bit like SYN flood attacks. However, the Land attack uses TCP; we're dealing with IGMP and Land attacks haven't really been much of a threat over the last couple of years; most of the operating systems affected have gone by the wayside (Windows 98, NT (pre-4.0), etc. So, we'll scratch land.c off the list of suspects.

Since there were only a total of 11 packets, if this was a DoS attempt, it was a pretty lame one. It seems none of the destination hosts "bought in" to the crafted packets and didn't join; even if they had, nothing much would have happened unless the hosts were part of the multicast group.

5. Attack mechanism:

Possibly, crafted IGMP packets were sent with spoofed IP source addresses matching the destination. These packets violate the standards for IGMP in several ways: TTL other than 1, MAC addresses are not within required range (01-00-5E-00-00-00 - 01-00-5E-00-00-FF) nor are the host addresses (224.0.0.1 - 239.255.255.255; 240.0.1.197).

By setting TTL to a relatively high value, perhaps the attacker is hoping the packet will reach its target. Since the IP and MAC addresses are out of spec for IGMP, none of the targeted hosts would likely cause any problems with the network's multicast activities.

Based on what I've discovered, I don't understand what an attacker would hope to accomplish; it just doesn't add up. Perhaps the would-be attacker is just messing with the intrusion analyst's mind! If that's the case, then this was a successful attack and did cause a DoS for at least one analyst (a brain cramp).

Therefore, I wouldn't rule out that this may not be an attack at all and is perhaps simply a case of network hardware or software misconfiguration. Either way, it certainly seems that no harm was done.

6. Correlations:

In Daniel Wesemann's GCIA practical, it's suggested traffic such as this could have been due to misconfiguration. [13] Otherwise, I had a difficult time correlating this detect. If the events triggering this event persist, I'd expect this will become a hot topic in the future.

CA-1997-28 and CVE-1999-0016 are referenced in the documentation for the snort alert that was triggered. CVE-2001-0796 covers as IGMP DoS attack. However, the information contained in these does not relate to what was seen in this detect.

A whois search of whois.arin.net (via www.geektools.com) provides the network registration information for the hosts:

```
170.129.0.0 - 170.129.255.255
OrgName: Standard Microsystems Corporation
OrgID: SMC-9
Address: 300 Kennedy Drive
City: Hauppauge Industrial Park
StateProv: NY
PostalCode:
Country: US
NetRange:170.129.0.0 - 170.129.255.255
CIDR: 170.129.0.0/16

NetName: SMCORP
NetHandle: NET-170-129-0-0-1
Parent: NET-170-0-0-0-0
NetType: Direct Assignment
NameServer: NS.PSI.NET
NameServer: NS2.PSI.NET
Comment:
RegDate: 1994-04-29
Updated: 1994-05-25
```

7. Evidence of active targeting:

I find no evidence of any active targeting to a particular host or hosts. The small group of hosts involved don't seem to have anything in common and I see no reason why they

[13] Wesemann, Daniel. "Practical Detect #1 -- Suspicious IGMP Packets"

were selected over any other hosts on the network. The selection of these hosts seems, more or less, random. As noted at the beginning of the analysis for this detect, although similar activity was seen in the five days leading up to this event, two of the three following days did see comparable events; a day in the middle of the three had no alerts like this. Also, a different pool of IP addresses was used in each of the three occurrences.

8. Severity: Assessing the detect's seriousness.
Criticality: 3

I'm not aware of the role of the destination hosts on this network. However, I will assume that they are active, production hosts and have at least some importance to the organization. I'll give a value of 3 for criticality.

Lethality: 2

I'll rate a 2 for lethality; even if the attack (if it is an attack at all) were successful, it may have little impact.

System countermeasures: 2

Since the crafted packets apparently caused no harm, it seems that what ever system-level countermeasures are in place are adequate. Also, because the IGMP were invalid, they likely would have been discarded by the receiving hosts anyway. System countermeasures gets a 2.

Network countermeasures: 1

I believe these erroneous IGMP packets with their loopback IP addresses should have been discarded at the network's entry point, never making it to the destination hosts. Because the network's perimeter security failed on this point, I'll rate Network countermeasures with a 1.

Restating our formula for Severity:

Severity = (Criticality + Lethality) - (System countermeasures + Network countermeasures)

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

With a Severity of only 1, this doesn't make for a high interest event. However, I would want to be on the lookout for similar traffic in the future. If the activity persists, appropriate action should be taken

9. Defensive recommendation:

System defense:

- If this alert recurs, attempt to locate the affected hosts for forensics and corrective action.
- Ensure patches for all hosts on the network are current.

Network defense:

Intrusion Detection System (IDS)

- Keep IDS signatures current and fine-tune as required.
- Monitor generated alerts, keeping an eye out for this particular issue to see if it persists.
- Investigate and take appropriate action as necessary.
- Additional snort rules could be added; to ensure IGMP packets are in the correct range of MAC addresses, multicast addresses, and have a TTL of 1.

Perimeter Security:

- Setup a firewall to stop dangerous traffic from entering the network, if one is not already present.
- Use egress filtering to ensure inbound traffic is blocked if the source IP belongs to the internal network.
- Drop packets when the destination matches the source IP address.
- Block outbound IGMP traffic if not required.
- Review configuration of network perimeter hardware/software to ensure misconfiguration could not cause the problem seen in this detect.
- If IGMP is not needed, then it should be disabled. This applies to any another protocols that are not required.

10. Multiple choice test question:

Which of the following is NOT true?

a) IGMP (Internet Group Management Protocol) is the protocol for IP multicasting.

b) The designated range for IGMP multicast addresses is 224.0.0.1 - 239.255.255.255.

c) The valid range for MAC addresses with IGMP is 01-00-5E-00-00-00 and 01-00-5E-00-00-FF.

d) IGMP requires a TTL value greater than 0, but less than 2.

e) None of the above answers are correct.

Answer: e Answers a - d are ALL correct!

PART THREE: Analyze This!

EXECUTIVE SUMMARY

Presented here is a network security audit for the University, as requested. Numerous audits have been conducted in the past by other analysts and their findings were consulted to correlate present-day events. It's evident great strides have been made to improve network defense. This review covers five days of logs, 19 - 23 October. More than 12 million events were recorded in the security logs during this timeframe; about 30 events per second. That's a lot of "goings on" to investigate. My intent at the outset of this project was to distill this extensive data into valuable information that will lead to increased protection for the University's network.

Upon my initial, cursory review of the logs, it was apparent there are many new, custom-made rules for the Intrusion Detection System, Snort. This is a positive development; taking an interest in protecting the network by tailoring the ruleset is certainly a step forward. If the rules are well-designed, then no doubt the University's network is safer.

These "home-grown" rules were responsible for logging more than 20,000 alerts between during the audit period. However, only about 700 of these events are actually security-related. Several home-grown alerts are apparently designed to collect network statistics; such as visits to web servers, file transfers by the HelpDesk, etc. I appreciate the importance of this sort of information, particularly to management. But, these alerts add greatly to the analyst's workload; this assumes the analyst reviews these alerts at all. My suspicion is that the personnel responsible for daily monitoring of the University's security logs simply discount them. This is far from optimal. Ideally, an analyst would only be represented with alerts representing events requiring investigation. Of course, there will always be "false positives" to contend with, but nothing should be dismissed without consideration. This complacency could easily influence the scrutiny of meaningful alerts. An analyst should never be given mounds of data and then told to take no notice. Always strive to give the analyst useful data, so they can provide useful security-related information to management in return.

Also of import is the version of the Snort engine; it's several versions behind. Each revision of Snort offers additional functionality to help safeguard networks. The cost of upgrading is free and the administrative effort is low. Also, the already existing Snort rules are quite out outdated. It's likely they haven't been updated since Snort was first installed at the University. Like the Snort engine, updating the rules takes nothing from the University's automation budget. There's no reason to not upgrade Snort to the latest version, along with the ruleset, soon.

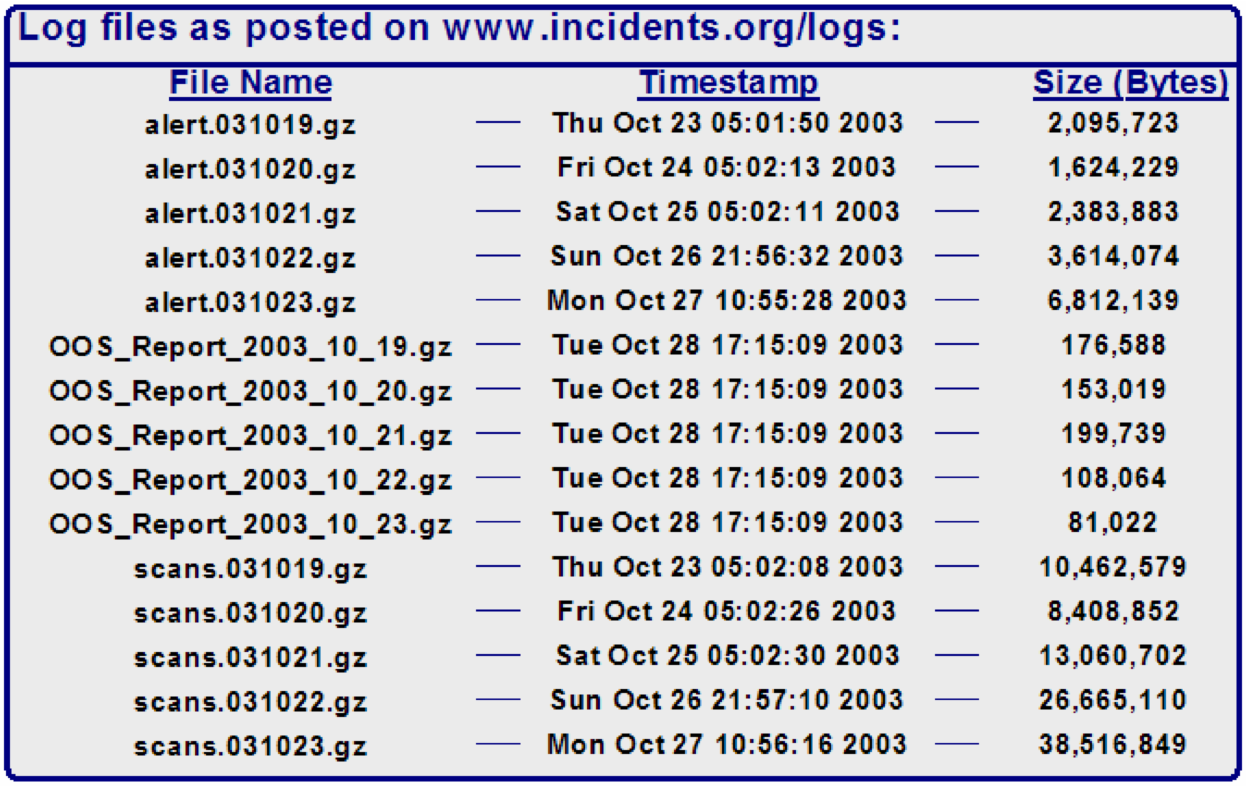
During these five days, a number of entities tried their hand at causing problems for the University's network; ranging from taking inventory of network assets, intrusion attempts, and introducing malicious software. Also, there are University computers being used for things they ought not. Additionally, a number of things should be done to reinforce the security perimeter. These are all things that should be addressed immediately. Recommended actions are provide throughout this report and summarized in "DEFENSIVE RECOMMENDATIONS".

A large number of events have been given painstaking analysis throughout this report; especially in sections "DETECT IN-DEPTH ANALYSIS", "TOP 10 TALKERS", and "NOTEWORTHY EXTERNAL HOSTS". Visual aids are often included to allow a better understanding of the events; for instance, a link graph is used to simplify a complicated relationship between hosts.

FILES ANALYZED: Network Security Log Files Analyzed

Three types of log files, covering five days of the University's network activity, were downloaded for analysis from www.incidents.org/logs. Alert, Out of Spec (OOS), and Scan logs were represented for each day's capture. According to the text on www.incidents.org/logs, these logs were collected using Snort. Only events that actually triggered alerts are represented in the files and ICMP, DNS, SMTP, and web traffic was deleted from the files. A total of 15 files with a combined size of close to 115 MB were downloaded.

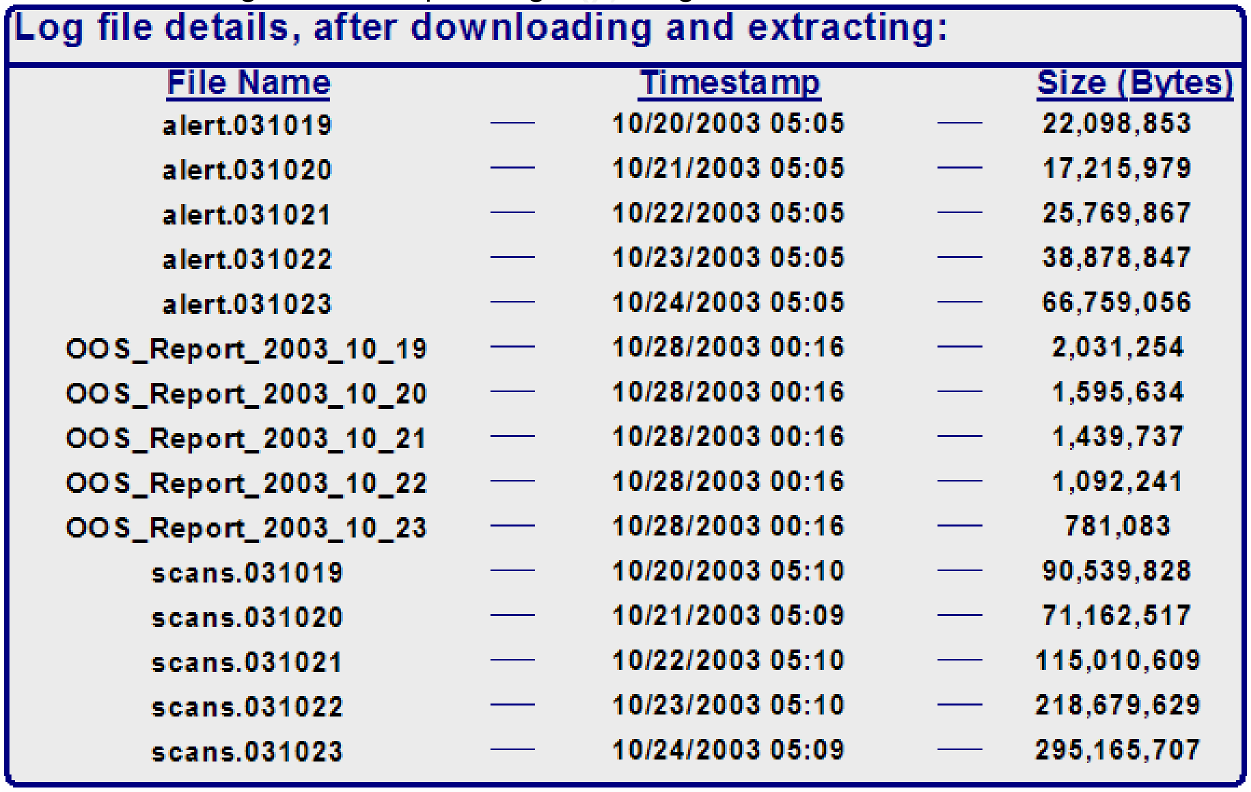
Listed below are the log files selected for analysis, covering the period of 19 - 23 October. These dates were selected because this was the only five-day timeframe having all three log types fully represented; allowing for the most complete snapshot of the network's activity. It is understood that some information, such as IP addresses has been obfuscated and that this affects the accuracy of whois queries and reverse DNS lookups. Because these tools are part of routine analysis, they are used for this audit. However, data returned was considered valid only if corroborated by other means.



Log files as posted on www.incidents.org/logs:

File Name	Timestamp	Size (Bytes)
alert.031019.gz	Thu Oct 23 05:01:50 2003	2,095,723
alert.031020.gz	Fri Oct 24 05:02:13 2003	1,624,229
alert.031021.gz	Sat Oct 25 05:02:11 2003	2,383,883
alert.031022.gz	Sun Oct 26 21:56:32 2003	3,614,074
alert.031023.gz	Mon Oct 27 10:55:28 2003	6,812,139
OOS_Report_2003_10_19.gz	Tue Oct 28 17:15:09 2003	176,588
OOS_Report_2003_10_20.gz	Tue Oct 28 17:15:09 2003	153,019
OOS_Report_2003_10_21.gz	Tue Oct 28 17:15:09 2003	199,739
OOS_Report_2003_10_22.gz	Tue Oct 28 17:15:09 2003	108,064
OOS_Report_2003_10_23.gz	Tue Oct 28 17:15:09 2003	81,022
scans.031019.gz	Thu Oct 23 05:02:08 2003	10,462,579
scans.031020.gz	Fri Oct 24 05:02:26 2003	8,408,852
scans.031021.gz	Sat Oct 25 05:02:30 2003	13,060,702
scans.031022.gz	Sun Oct 26 21:57:10 2003	26,665,110
scans.031023.gz	Mon Oct 27 10:56:16 2003	38,516,849

After downloading and decompressing the 15 log files, here's the result:



Log file details, after downloading and extracting:

File Name	Timestamp	Size (Bytes)
alert.031019	10/20/2003 05:05	22,098,853
alert.031020	10/21/2003 05:05	17,215,979
alert.031021	10/22/2003 05:05	25,769,867
alert.031022	10/23/2003 05:05	38,878,847
alert.031023	10/24/2003 05:05	66,759,056
OOS_Report_2003_10_19	10/28/2003 00:16	2,031,254
OOS_Report_2003_10_20	10/28/2003 00:16	1,595,634
OOS_Report_2003_10_21	10/28/2003 00:16	1,439,737
OOS_Report_2003_10_22	10/28/2003 00:16	1,092,241
OOS_Report_2003_10_23	10/28/2003 00:16	781,083
scans.031019	10/20/2003 05:10	90,539,828
scans.031020	10/21/2003 05:09	71,162,517
scans.031021	10/22/2003 05:10	115,010,609
scans.031022	10/23/2003 05:10	218,679,629
scans.031023	10/24/2003 05:09	295,165,707

Note the difference in “advertised” dates and the actual timestamps for the log files; there seems to be about a three-day difference. Also, the timestamps within the logs do not include the year, only month, day, and time. However, there were no other discrepancies between the dates within the logs and the files names; files with a name indicating they were for October 19th actually contained detects for that date.

Once the logs were downloaded and extracted, I was left with nearly 1 gigabyte’s worth of files filled with more than 12 million records ready for analysis. All of the log data was parsed and imported into a relational database to facilitate data-mining. A small number of corrupt records were discarded. The process of pulling the logs into the database receives attention in “ANALYSIS METHODOLOGY AND LEASSONS LEARNED”

HOST ROLES AND RELATIONSHIPS: Methodology for identifying hosts and their function on the network.

In order to conduct a comprehensive security log audit, it’s important to understand the makeup of the network. Having a detailed map of the network would have been ideal; however, one was not made available. Therefore, it was necessary to use other means to deduce the roles of the hosts on the network. To determine which hosts were of most interest, either due to the role played on the network or because of risk of service running, several methods were used.

As part of the preparation for analysis, the five days of logs were imported into a relational database. Details of how this was accomplished are discussed at the conclusion of this paper. Having all the logs in database tables, it was simply a matter of running a few queries to gather the information needed to provide a picture of the network; structured query language (SQL) commands were issued to the database to get the required results. In the interest of brevity, the actual SQL statements used will not be shown along with this commentary, but examples are included in the section “ANALYSIS PROCESS AND LESSONS LEARNED”.

First, I wanted to find out the number of hosts, to get an idea of the network’s size, so I queried the three sets of the alert tables (alerts, oos, and scans) for a distinct list of host addresses. A total of 8625 unique host addresses were seen in the logs; around 1100 of these belonging to the University.

Based on the source and destination port, along with the amount of traffic generated, it’s possible to speculate as to the function of given a machine. The IP address, host name, registration information, and alert messages can also provide some clues.

For example: A host receiving lots of traffic on port 80 is likely a web server (HTTP). Abundant traffic destined for port 25 may signify the source is an e-mail server (SMTP). A host doing a lot of talking on port 53 it could indicate it is providing Domain Name Service functionality (DNS).

Even the IP address can be helpful. Often, key machines are assigned IP addresses where the last octet is a low number; #.#.#.1 for the domain controller, #.#.#.2 for the backup domain controller, e-mail on #.#.#.3, for example. It doesn't always work this way, but having a closer look at alerts generated by machines with IP addresses that seem to "stick out" can aid in identifying the role on the network.

The name of a host can also drop hints. You can perform a "Reverse DNS Lookup" to obtain the name of select hosts. Often, the host's name will imply its occupation on the network. For instance, the IP address MY.NET.12.4, with its low final octet, turns out to have a host name of "mail.theuniversity.edu", while MY.NET.24.34, having engaged in plenty of port 80 traffic happens to be named www.theuniversity.edu. With names like these, it's not hard to guess what these machines are meant for. In the first case, the host name was obtained, based on a hunch that its assigned IP address was significant. Knowing the name included "mail" helped to pare down what to look for in the traffic to determine this host's true function, focus on SMTP-related clues to confirm or refute suspicions. In the second case, incoming port 80 traffic sure seems like HTTP; looking up the host's name helps to bolster the theory.

Another tip for reverse DNS lookups: This process queries the network's server designated for DNS, the name of the DNS server is usually displayed. So, our reverse DNS lookup for MY.NET.12.4 not only revealed its name, www.theuniversity.edu, we learned that one of the machines responsible for DNS is uni5.theuniversity.edu (MY.NET.1.5). Or, when performing a whois query, check the "NameServer" data displayed for the DNS servers registered along with the IP you queried for. Either way: Two birds, one stone.

Tip: A handy website for gathering this kind of information is www.dns.stuff.com

Finally, message text from alerts can be useful. An alert referring to "ICMP" or "helpdesk" would lead one to other conclusions; router or HelpDesk in this case.

All of these methods were put to use, correlating them with one another, to derive these tables of hosts with their presumed service. While information about many more hosts was gathered, only some of the "prime movers" are accounted for here.

DNS	**HTTP**	**FTP**
MY.NET.1.3	MY.NET.30.4	MY.NET.161.40
MY.NET.1.4	MY.NET.5.95	MY.NET.24.13
MY.NET.1.5	MY.NET.189.62	MY.NET.24.18
	MY.NET.5.15	MY.NET.24.19
SMTP	MY.NET.24.34	MY.NET.24.27
MY.NET.12.2	MY.NET.5.20	MY.NET.24.47
MY.NET.12.6	MY.NET.24.44	MY.NET.24.9
	MY.NET.100.165	MY.NET.30.3
HelpDesk	MY.NET.29.18	MY.NET.30.4
MY.NET.53.29	MY.NET.29.8	MY.NET.53.29
MY.NET.70.49		MY.NET.70.49

MY.NET.70.50 **Printers** MY.NET.60.14 MY.NET.24.44		MY.NET.70.50

In Hee So's GCIA Practical Assignment v3.0, which received honors, other methods for accomplishing the task of network mapping are discussed. This paper can be found here: http://www.sans.org/rr/papers/index.php?id=836.

EVENT IN-DEPTH ANALYSIS

In the previous section, we learned a bit about the network and the hosts residing on it. Just as it is important to understand which hosts are most important ("hosts with the most"), we need to have a cursory view of all events types so priorities can be set.

So, the next step in preparing for in-depth analysis was to research the alert messages to determine their meaning and locate, or create from scratch, rules similar to those in-place at the University. Without this knowledge, an analyst may be tempted to define their plan of action based solely on the number of events generated by each alert. This isn't always advisable. Obviously, an alert that may have occurred just a few times but is of high severity should be investigated first; it's more important than a large-scale port scan, for instance. A probable false positive on a very important machine should be given prompt attention before a large-scale probe is investigated - even if the first case probably will turn out to be nothing - if it's *not* nothing and you wait - you'll be sorry!

In this section, we'll cover a lot of ground. There were a total of 52 unique alerts generated for the 19 - 32 October timeframe. Rather than simply listing the alerts with some subtotals of "hits", we'll dig much deeper.

The purpose of each of these alerts, along with the associated vulnerabilities and exploits, will be explained. Examples of the rules generating the alerts are included. In cases were the probable rule could not be found from a reliable source, a sample rule was created. Detects featuring more in-depth analysis will be presented first, others are listed based on the number of events. Also, correlations are cited and recommendations offered.

There's a lot more information here than just how many times the alert fires; of course that's included, too. Additional in-depth analysis of many other events can be found elsewhere in this report, especially in "TOP 10 TALKERS" and "NOTEWORTHY EXTERNAL HOSTS".

CUSTOM ALERTS

There are a dozen or so Snort rules in-place that were written specifically for the University. Because most of these rules are related, they will be discussed collectively.

[UNI NIDS IRC Alert] IRC user /kill detected, possible trojan: This alert was generated 341 times by 49 distinct external hosts against 51 internal hosts. No internal hosts triggered this alert; all source IP addresses were external. This alert triggers on inbound traffic in established connection with source ports ranging from 6663-7000 where "user /kill" is included in the content. The "user /kill" command is used to terminate a client's IRC connection. This occurs when the client is no longer welcome; the "kill" command is issued by the IRC server. So, we have quite a few cases of IRC sessions being ended abruptly. What did the University's hosts do to wear out their welcome?

[UNI NIDS IRC Alert] XDCC client detected attempting to IRC: This is a case of an internal client trying to make a direct XDCC connection to an external host. XDCC is used for peer-to-peer (P2P) file sharing. Clients connect directly to one another, meaning not via an IRC server as with other XDCC-related. More importantly, these connections are made without proper network authentication; if someone can get out with XDCC, then someone could get in. The destination port tested for in this alert is 6667; this traffic on this port should be blocked from entering or leaving the network.

[UNI NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC: Another case of an attempt to make an IRC connection. In this case, "sdbot" is a trojan that allows remote control of a client.

[UNI NIDS IRC Alert] Possible drone command detected: This alert triggers on incoming traffic on port 6667 with a payload content indicating the external host is attempting to control the target host for use in a DDoS.

[UNI NIDS IRC Alert] Possible Incoming XDCC Send Request Detected: Here, a likely file transfer request has been received from an external source.

[UNI NIDS IRC Alert] K\:line'd user detected: This alert indicates that a user attempted to connect to an IRC server, but was automatically upon connection. This would happen if the user has been added to a "forbidden" list.

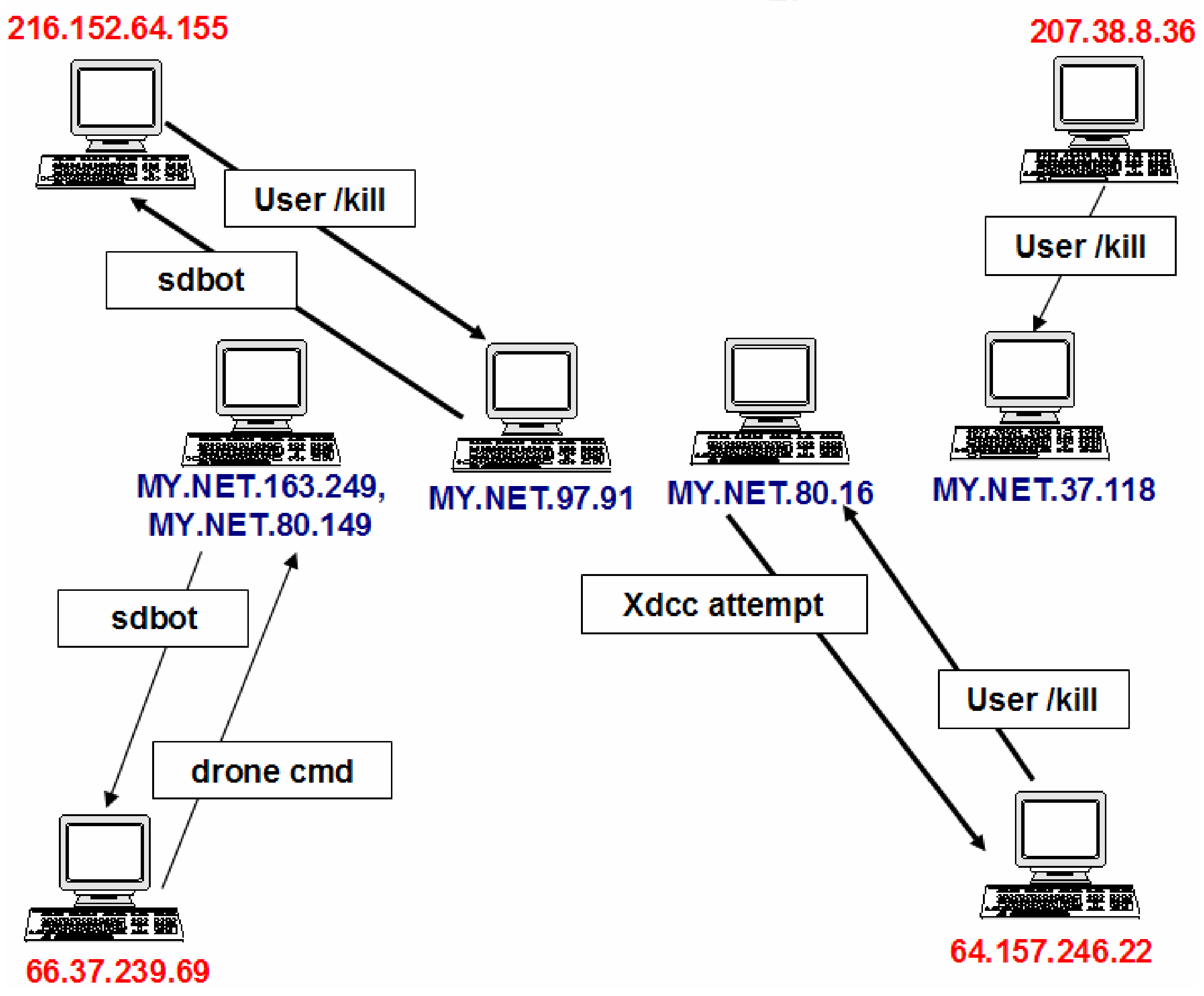
[UNI NIDS IRC Alert] Possible trojaned box detected attempting to IRC: This alert triggers when an internal host, that may be under remote control from an intruder, is attempting to make an IRC connection to an outside host.

IRC evil - running XDCC: Alert triggers on incoming traffic to port 6666. The actual rule in-place at the University may key on ports other than 6666; however, the only alert seen in the logs was to this port. Since there is only a single alert, it's difficult to deduce the actual stimulus for this alert.

[UNI NIDS] External MiMail alert: W32/Mimail.i@MM is a brand-new worm that related to credit card fraud. The worm attempts credit card theft from unwary recipients of an e-mail where the sender claims to be Paypal. The email states that the user's account will expire soon and that the “customer” must send their credit card information by filling in the form included in the e-mail.

[UNI NIDS] Internal MSBlast Infection Request : MSBlast, also known as Lovsan and others, exploits a vulnerability in the Windows operating system. This vulnerability was announced by Microsoft in September of 2003 and relates to RPC (Remote Procedure Call). RPC was intended to allow services on one machine to be run from another. Unfortunately this recently discovered weakness would allow an attacker pretty much do as they wish with the targeted system.

By creating customized rules to monitor traffic of this nature, the University is moving in the right direction. It is advisable to take this a step further to prevent this traffic from entering the network to begin with. To support the University’s new-found appreciation for the danger of these events, a link graph is presented to make the complex traffic patterns visible.



The link graph above illustrates the interaction between four internal and external hosts. These hosts were selected because they were the most active in terms of generating XDCC-related alerts and more importantly, with one exception; all of the external hosts were seen as both source and destination. Arrow direction indicates the direction of traffic flow that caused the alert the weight of the arrow indicates the relative amount of alerts generated. **MY.NET.163.249** and **MY.NET.80.149** caused the alert "[UNI NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC" with **66.37.239.69** as the destination; whose response triggered the '[UNI NIDS IRC Alert] Possible drone command detected.' alert. "sdbot" is a trojan that allows for remote control of a client. "drone" refers to a client that is under control by someone not wanted and they can use the machine to... The pairing of the sdbot and drone alerts could mean the internal host is being manipulated by an intruder, possibly to force the target to take part in a distributed denial of service (DDoS). All alerts generated by **MY.NET.97.91** were "[UNI NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC" and all of these triggered in response to outgoing traffic to **216.152.64.155**. In turn, nearly all of the alerts from 64.155 were "UNI NIDS IRC Alert] IRC user /kill detected, possible trojan.". **MY.NET.80.16** only caused alerts on traffic headed to **64.157.246.22**, "XDCC client detected attempting to IRC". The external host responded with "UNI NIDS IRC Alert] IRC user /kill detected, possible trojan." The alerts caused by **207.38.8.36** with **MY.NET.73.118** as the destination, "UNI NIDS IRC Alert] IRC user /kill detected, possible trojan." are notable because, unlike other examples, there were no alerts triggered to indicate the stimulus. Because the "UNI NIDS IRC Alert] IRC user /kill detected, possible trojan." would only be triggered by an established session, there must have been some stimulus that triggered the external host's response.

MY.NET.30.3 activity **MY.NET.30.4 activity**		Total Hits: **5726** Total Hits: **15604**

Alert explanation:

Triggers on all incoming traffic to MY.NET.30.3 or .4, except, perhaps, port 111.

Vulnerability/Exploit:

Alert on all inbound traffic to two key network assets. An attacker would consider these targets of high-value.

Snort rule sample:

Based on the Alert Description and Vulnerability/Exploit, I created the Snort rule below. The similarly-named rule for MY.NET.30.4 would be identical, except for the destination IP and the message text. This very simple rule triggers on inbound traffic using any protocol, source IP, or port to MY.NET.30.3 on any port except 111.

```
alert any $EXTERNAL_NET any -> MY.NET.30.3 !111 (msg: "MY.NET.30.3 Activity";)
```

Or, a rule similar to this could be in place:

```
alert any $EXTERNAL_NET any -> MY.NET.30.4 any (msg: “MY.NET.30.3 Activity”;)
```

This second rule fires on any destination port. Just because the actual rule never triggered on port 111 traffic, doesn’t mean the rule excludes it; there may have been no port 111 traffic at all. However, another rule did pickup on port 111 traffic to these hosts, so other rules must be in place that fired before “MY.NET.30.3 Activity” and “MY.NET.30.4 Activity”. Remember: Snort uses a “first match" policy when processing alerts.

Here’s a sampling of the alerts:

```
[**] MY.NET.30.3 activity [**]
10-19 00:01:24.412416 68.57.90.146:1032 -> MY.NET.30.3:524
[**] MY.NET.30.3 activity [**]
10-19 00:01:25.593280 68.57.90.146:1032 -> MY.NET.30.3:524
[**] MY.NET.30.3 activity [**]
10-19 00:01:25.593294 68.57.90.146:1032 -> MY.NET.30.3:524

        ...  trimmed for brevity ...

[**] MY.NET.30.3 activity [**]
10-23 23:46:26.915765 165.247.82.136:1466 -> MY.NET.30.3:524
[**] MY.NET.30.3 activity [**]
10-23 23:46:27.403726 165.247.82.136:1466 -> MY.NET.30.3:524
[**] MY.NET.30.3 activity [**]
10-23 23:48:00.524549 165.247.82.136:1466 -> MY.NET.30.3:524
```

Note: Although only port 524 traffic to MY.NET.30.3 is represented in the samples above, alerts were seen for MY.NET.30.4 and on many other ports as well.

What happened:

A total of 21330 alerts were generated for “MY.NET.30.3 activity” and “MY.NET.30.4 activity” combined. There were 493 unique external hosts involved, about 5% of these caused 80% of the alerts. The bulk of the alerts were caused by IPs in the range assigned to Comcast Cable Communications, an ISP based in New Jersey. Also, a single IP assigned to Verizon Internet Services, also an ISP based in Reston Virginia, was responsible for about 600 hits.

The alerts seen were on a slew of ports, 56 unique destination ports, listed below:

10, 21, 23, 80, 110, 135, 139, 443, 445, 524, 554, 631, 1002, 1006, 1054, 1135, 1187, 1243, 1257, 1265, 1290, 1368, 1374, 1433, 1524, 1542, 1552, 1598, 1642, 1650, 1651, 1662, 1803, 1809, 1903, 1930, 1964, 2000, 2301, 3018, 3019, 3128, 3297, 3389, 4000, 4128, 5128, 8000, 8080, 8081, 8888, 9090, 17300, 39706, 51443, 52080

Of all these, only three had enough traffic to at first appear significant; ports 80, 524, and 51443 (3918, 6817, and 10378 hits, respectively). We've already 'tagged' hosts MY.NET.30.3 and MY.NET.30.4 as being web servers, so seeing plenty of port 80 (HTTP) traffic should be no surprise. But what about ports 524 and 51443? (Note: other ports and their potential hazards are discussed along with the recommendations for this detect.)

Port 524 is an important port for Novell Netware networks. It's used for NCP (Netware Core Protocol) with both TCP and UDP. On TCP, port 524 is used for two server-client communications, while UDP port524 supports time synchronization between servers. When the network operating system is Netware, you're sure to see plenty of port 524 traffic. [14]

As for port 51443, it can also have a couple of uses. Port 51443 is assigned, by default, to HTTPS when Apache Web Server is installed on NetWare Enterprise server. Additionally, port 51443 can be used for another Novell product called "iFolder". It's a way for users to access important files when they are away from their usual workstation via an Internet browser. [15] If the user chooses to install the iFolder software on their home computer, laptop, etc, then their data can be synchronized, kept up-to-date, with the Novell server they normally connect to. It's possible some of the traffic was caused by legitimate users accessing their files at the University remotely.

Presuming MY.NET.30.3 and MY.NET.30.4 are both web servers, probably running Apache Web Server on Netware, the traffic we're seeing isn't that out of line from what we'd expect.

Recommendations:

So why create rules to generate these alerts? Are these alerts just checking for visitors? I'm inclined to answer "yes". It's probable these alerts are only for statistic gathering purposes. If this is the case, then no real harm has been done. However, there are better ways to accomplish this. If Snort is to be used to gather the stats, then change the rules so the data is logged, rather than raising an alert. This is easily accomplished by substituting the "alert" tag at the very beginning of the rule with "log", as shown in the example I created below.

```
        log any $EXTERNAL_NET any -> MY.NET.30.4 any (msg: "MY.NET.30.3
Activity";)
```

Some of the other ports involved do concern me a bit, however. Here's a partial list of the "slew" listed above and there possible association with more dangerous activity.

[14]Novell. "Netware 5 Features".
[15] Novell. "iFolder Features"

2000:	A-trojan, Fear, Force, GOTHIC Intruder, and many more
3128:	RingZero and Reverse WWW Tunnel Backdoor
4128:	RedShad (remote access)
5128:	Squid
8000:	session hijacking
8080:	RingZero
8888:	Dark IRC
9090:	Aphex (packet sniffer)

Obviously, we don't want to see anything like this on the University's network. Consider blocking these ports until further research on the hazards listed above can be completed.

Correlation:

I found confirmation of my theory on the probable use of port 542 in Steven Gamble's GCIA practical. [16]

SMB Name Wildcard		Total Hits: **199212**

Alert explanation:

This alert triggers when an external source queries an internal windows host for all shared resources.

Server Message Block (SMB), most-often seen on Windows hosts, is a shared resource protocol. The alert "SMB Name Wildcard" normally means that the initiating host wants a list of all shared resources (drives, files, printers, etc) from the destination host. Normal SMB traffic would be within the network and port 137 is used for both the source and destination ports.

Vulnerability explanation:

The source is checking for all resources that are shared by the targeted host. This is reconnaissance and often leads to more serious activity in the future. If the destination is found to be vulnerable and the attacker believes something of value exists within the shared resources of the target, then a compromise may be attempted.

Snort rule sample:

This example of what the actual rule may look like triggers on inbound traffic to UDP port 137 having content consistent with SMB.

[16] Gamble, Steven. "GCIA Practical".

```
alert udp $EXTERNAL any -> $INTERNAL 137 (msg:"SMB Name Wildcard";
content:"CKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA|0000|";)
```

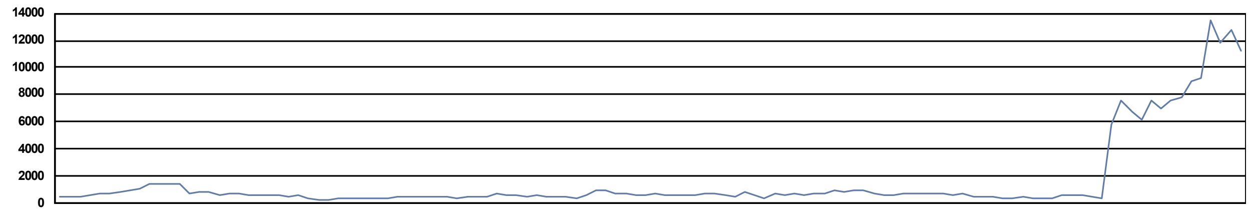
Alert snippets:

```
    [**] SMB Name Wildcard [**]
    10-19 00:00:24.716494 MY.NET.150.133:3117 -> 172.202.107.2:137
    [**] SMB Name Wildcard [**]
    10-19 00:00:24.716518 MY.NET.150.133:3117 -> 24.156.171.176:137
    [**] SMB Name Wildcard [**]
    10-19 00:00:28.635977 MY.NET.150.133:137 -> 200.24.182.77:137

            ...  trimmed for brevity ...

    [**] SMB Name Wildcard [**]
    10-23 23:48:45.712777 MY.NET.80.51:1035 -> 162.201.75.116:137
    [**] SMB Name Wildcard [**]
    10-23 23:48:45.967725 MY.NET.80.51:1036 -> 210.180.75.215:137
    [**] SMB Name Wildcard [**]
    10-23 23:48:46.012708 MY.NET.80.51:1035 -> 162.201.75.118:137
```

What happened:

We have something quite different from normal SMB traffic in this case. All of the traffic causing this alert is outbound, 99.9% is to port 137, 199026 hits. There are two hosts are responsible for nearly all of this traffic, MY.NET.80.51 and MY.NET.150.133. The source ports being used range from 1032 to 49613. A small portion of hits are ports 137/137, but it's still outbound traffic. **A reverse DNS lookup provides names for the two hosts, "ss-80-51.pooled.theuniversity.edu" and "yuna.lib.theuniversity.edu".** MY.NET.80.51's **name implies its address is part of a DHCP address pool and is probably a user workstation, rather than server or other high-value target.** MY.NET.150.133's name indicates it is used by the University's library and therefore should be considered an asset of more importance.

The line graph below shows the combined alert activity from MY.NET.80.51 and MY.NET.150.133 during the five day analysis period. Only "SMB Name Wildcard" alerts are included. Note that for much of the time, the activity is fairly low and consistent. At about 10 AM on the final day, October 23rd, we see a surge in the traffic; jumping from around 400 hits per hour to upwards of 14,000 hits per hour. Clearly, something's amiss.



Without the spike in activity, I may have suspected that this activity is normal for the network or, worse case, a couple of misconfigured hosts. Because this traffic went from a trickle to a gush, it's possible that these hosts have been compromised and are being used in a reconnaissance effort.

I could find no alerts to indicate when a compromise would have occurred.

A total of 115,610 unique destinations were "touched" by these two University hosts. Next, I wanted to know if any of the destination hosts had ever caused any alert as a source. Among the more than 100,000 external hosts, only one was ever seen as a source: 68.42.146.143. A whois query and reverse DNS lookup provide this host's name and the organization to which it is registered: "**bgp01138778bgs.ypwest01.mi.comcast.net**" of "Comcast Cable Communications"

68.42.146.143 triggered a single alert, "FTP passwd attempt" against MY.NET.24.47. Note that the destination is not one of those we were hoping to see (MY.NET.80.51 or MY.NET.150.133). A reverse DNS lookup for MY.NET.24.47 provides a name of "**mirrors.theuniversity.edu" for this host. So, efforts to gather information about** MY.NET.24.47 and 68.42.146.143 have lead us to a dead-end, in regards to this alert; there's just no relation. However, we did discover that there were possible attempts to compromise MY.NET.24.47 by 35 unique external hosts.

Back to the task at hand: What data was being transferred? Is it consistent with SMB? Since we only have the alerts on which to base our analysis, it's difficult to answer these questions. If we were able to examine the actual datagrams responsible for these alerts, then a more sound assessment could be offered. Even if the traffic turned out to be harmless, we could use that information to fine-tune the rule in order to trim down the number of alerts generated.

Recommendations:

These two hosts should be tracked down and inspected. If compromise or misconfiguration is evident, then further appropriate action should be taken. Once the issue is resolved with these two hosts, then the alerts generated by the remaining hosts should be monitored and investigated as needed. Also, because this exploit relates to a Windows-specific vulnerability, the attacker may have performed OS fingerprinting prior to this event.

As stated earlier, this alert can indicate normal traffic among windows hosts within the network, if on ports137/137. If it were inbound traffic to port 137, it could likely be probing for NETBIOS information or just noisy traffic. Either way, steps should be taken to limit this when/if it's caused by "noisy" traffic. Blocking incoming/outgoing traffic on port 137 and tuning the rule so it doesn't trigger on internal 137/137 traffic will cut down on the alerts an analyst must sift through.

Additionally, MY.NET.24.47 should be examined for possible compromise in regards to the alert "FTP passwd attempt", generated nearly 50 times for this host. Consider blocking suspicious external hosts causing this alert against MY.NET.24.47.

Correlations:

Based on the above, I suspected MY.NET.80.51 and MY.NET.150.133 were not properly configured, but felt I needed correlation from another analyst that had investigated similar traffic. I found the confirmation I was looking for here: http://www.sans.org/y2k/practical/Teri_Bidwell_GCIA.doc. In Teri Bidwell's GCIA Practical, a comparable case was analyzed and it was suggested that the bulk of the traffic was a due to a misconfigured Linux host. However, Teri did not offer any correlation to support this theory. [17] I concur that this may be a case of misconfiguration, but don't feel I have sufficient information to conclude that these must be Linux hosts.

SMB C access		Total Hits: **28546**

Alert explanation:

This alert relates to root-level file access attempts made by external hosts.

Vulnerability explanation:

If an intruder is successful in exploiting this vulnerability, the potential damage to the host is limited only by the attacker's imagination; information theft, file and configuration manipulation, introduction of malicious code, etc.

Snort rule sample:

The current Snort rule, 533, is shown below and was taken from www.snort.org (http://www.snort.org/snort-db/sid.html?sid=533).

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:"NETBIOS SMB C$ access"; flow:to_server,established; content: "|5c|C$|00 41 3a 00|";reference:arachnids,339; classtype:attempted-recon; sid:533; rev:5;)
```

Rule 533 triggers when an attempt is made, during an established session, to access the C: drive of the destination host. Note that the source must be external and the destination port 139. Also, the payload content is checked. This well-written rule generates few false positives.

Alert snippets:

```
[**] SMB C access [**]
```

[17] Bidwell, Teri. "GCIA Practical Assignment"

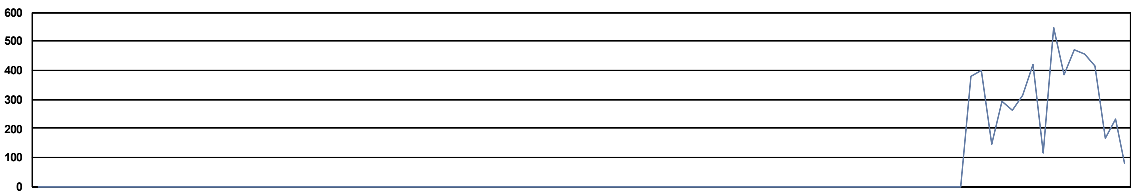
```
10-22 19:10:56.566786 65.66.17.86:4489 -> MY.NET.84.228:139
[**] SMB C access [**]
10-22 19:10:56.649510 65.66.17.86:4489 -> MY.NET.84.228:139
[**] SMB C access [**]
10-22 19:10:56.735481 65.66.17.86:4489 -> MY.NET.84.228:139

          ...  trimmed for brevity ...

[**] SMB C access [**]
10-23 10:22:06.721757 80.53.165.92:1796 -> MY.NET.84.228:139
[**] SMB C access [**]
10-23 10:22:06.854015 80.53.165.92:1796 -> MY.NET.84.228:139
[**] SMB C access [**]
10-23 10:22:07.641987 80.53.165.92:1796 -> MY.NET.84.228:139
```

What happened:

The name of this alert, "SMB C access", implies that an external host has attempted to access the primary hard disk of the destination host. Without having access to the rule or the payload that triggered it, it's difficult to say with certainty exactly what has happened. For now, we'll assume that the rule is well-written and is probably based on one provided by snort.org, although an older version.

I'll assume the Snort rule in place at the University, "SMB C access", is similar to the one above. Generally, c$ is the administrator share on Windows hosts (read "root-level access!"). More than 600 external hosts have made nearly 30,000 attempts to access about 1000 systems connected to the University's network. This is not good. Compromise attempts against most of these hosts are just a few times each. However, two destinations stand out; MY.NET.191.228 and MY.NET.84.228; 1146 and 5088 attempts, respectively.

I performed a reverse DNS lookup on both of the University's hosts. My search for MY.NET.191.228 turned up nothing. MY.NET.84.228 (MY.NET.84.228) resolved to "engr-84-228.pooled.theuniversity.edu"; possibly a dynamically assigned IP address set aside for the University's Engineering department. An important network asset would normally be assigned a static IP address, so I'll assume these hosts are not vital to the network and, while they may be the target of an attacker, these hosts probably wouldn't be considered highly desirable. Because the session must be established and the intruder would be expecting a return, it's not likely the source IP is spoofed.



I wondered what occurred with the two University hosts that lead up to this. 193.114.70.169 was the only other host causing alerts other than "SMC C Access"; two instances of "External RPC call" against MY.NET.84.228 shortly after 7 AM on the 23rd of October.

I found nothing in the OOS logs relating to the two internal hosts. Although there was some activity involving them in the scan logs, this did not appear to be noteworthy.

68 unique hosts made between 7 and 177 attempts each of this exploit against MY.NET.191.228 and MY.NET.84.228. A thorough scouring of the alerts, OOS, and scan logs revealed no other activity from these hosts; about half of them were destinations in "SMB Name Wildcard" alerts. The rule for "SMB C Access" requires a connection be established, so the IP addresses of the 68 external hosts would not be spoofed.

Recommendations:

Although I was unable to determine just how or when it happened, somehow the word got out to would-be attackers that MY.NET.191.228 and MY.NET.84.228 might be easy targets; nobody seemed to have any interest in them prior to the 22nd of October. These hosts should be checked to make sure they have not been compromised. For all servers and workstations, not only those affected by this alert, sharing should be disallowed either technically or by policy except when absolutely necessary. I recommend implementation of ingress filtering to block TCP port 139 traffic at the network's perimeter.

Correlations:

I found correlation for this alert in GCIA Practicals from Al Maslowski-Yerges and Hee So. [18] [19]

> http://www.giac.org/practical/GCIA/Al_Maslowski-Yerges_GCIA.pdf
>
> http://www.giac.org/practical/Hee_So_GCIA.doc

connect to 515 from inside		Total Hits: **7131**

Alert explanation:

Port 515 is assigned to LPRng (print spooling). Printer servers listen on port 515 (destination) for traffic using 721 as the initial source port.

Vulnerability explanation:

[18] Maslowski-Yerges, Al. "GIAC Certified Intrusion Analyst (GCIA) Practical Assignment Version 3.3"
[19] So, Hee, GCIA Practical Assignment v3.0, Feb 16, 2002

There is a known vulnerability in certain versions of LPRng which would allow an intruder to execute any commands they wish. This exploit is outlined in CVE-2000-0917 which can be found here: [20]

http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2000-0917.

Snort rule sample:

Below is a simple example of a Snort rule I created that would trigger on outbound TCP traffic to port 515 and having port 721 as the source.

```
alert tcp $HOME_NET 721 -> $EXTERNAL_NET 515 (msg:"connect to 515 from inside";)
```

A less specific rule, setting the source port to "any", for instance, could yield the same result. If payload content is known, then a more specific rule could be written that would be less prone to generate false positives.

What happened:

Practically all of this traffic is from MY.NET.162.41 to 128.183.110.242, always using source and destination ports of 721 and 515, respectively. Initially, what we're seeing here isn't terribly unusual. However, the alert message indicates the destination is external. This is not normal; printing should be to internal devices and port 721 should only be used as the source port for the first connection, other ports are used thereafter.

This anomalous traffic began mid-afternoon the second day of the five days that were analyzed and continued at least until late at night on the last day of logs analyzed, 23 October. I suspected the traffic may well have continued. No logs were posted at incidents.org for 24, 25, or 26 October. So, I decided to download and inspect October 27th's logs for this activity;, it was there for the entire 24-hour period. Next, I downloaded a couple more weeks of alert logs, decompressed them and grep'd for the text string "[**] connect to 515 from inside [**] MY.NET.162.41:721 -> 128.183.110.242:515" within the log files. Numerous instances were found in all of the logs up until 13 November. A non-exhaustive search revealed no other traffic involving MY.NET.162.41 or 128.183.110.242 within these additional logs. Closer inspection of the log file for the 13th indicated the traffic ceased at about 10:30 in the morning.

The first and final alerts are shown below.

```
10/20-14:04:06.54699  [**] connect to 515 from inside [**] MY.NET.162.41:721 ->
128.183.110.242:515

        (…24 days of alert logs trimmed for brevity …)
```

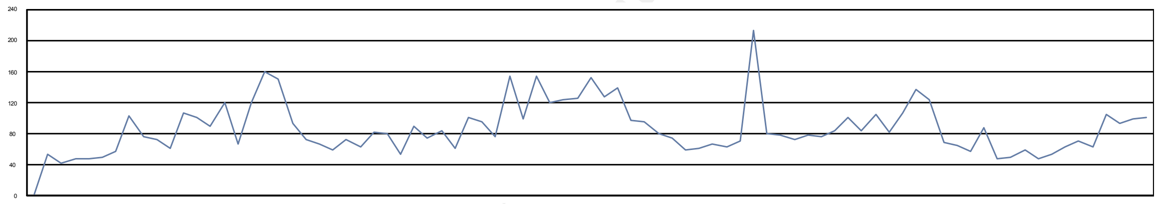
[20] Mitre. "CVE-2000.0917"

```
11/13-10:29:12.986400  [**] connect to 515 from inside [**] MY.NET.162.41:721 -
> 128.183.110.242:515
```

I performed a reverse DNS lookup to get more clues about the destination host. The IP address 128.183.110.242 resolves to tek924.gsfc.nasa.gov. A couple of google searches later, I learned that "GSFC" is NASA's Goddard Space Flight Center in Greenbelt, Maryland. I also found a link to a page posted by a couple of GSFC's network administrators that was also helpful. I'd already suspected that "tek" referred to the print brand name. GSFC names their printers based on location and, according to their list, they do use printers manufactured by Tek. However, "tek924" doesn't quite follow their naming convention and no locations are listed even close to "924" (room numbers are all between 30 and 70). [21]

I could find no alerts prior to the first "connect to 515 from inside" to explain the stimulus for this event. The internal host, MY.NET.162.41, is not known to support any special network functions (DNS, SMTP, HTTP, etc), so is not considered a high-value asset. A reverse DNS lookup for this host offers the name "**physics422-laptop.theuniversity.edu**".

To see if there was any sort of pattern to the traffic, I created a line chart, based on time, for the five day period.



As you can see, although it's not quite a "flat-liner", the activity is more or less consistent throughout the five days.

There are known vulnerabilities that could be exploited involving LPRng, mostly affecting Linux boxes. However, because I saw no indication of a compromise leading up to this event, I thought perhaps this could be caused by misconfiguration. I found an excellent how-to article about LPRng and RFC 1179 (Line Printer Daemon Protocol) posted among Massachusetts Institute of Technology's tech site. [22]

http://web.mit.edu/source/third/lprng/doc/LPRng-HOWTO-18.html

As stated earlier, normally port 721 is only used for the first connection. Usually, another port would be used after. However, according to web.mit.edu, there is a setting called "SO_REUSEADDR" that can be used with LPRng that allows port 721 to be used immediately for retry attempts; rather than waiting for a timeout and using a different

[21] Goddard Space Flight Center. "Reference Manual"
[22] Goddard Space Flight Center. "Reference Manual"

port. [23] But, this is only valid once the connection has been established. Or, since the expected response wasn't received, the source tried again and again; print jobs being sent to a nonexistent print spooler. So then, it's possible the "right" setting could cause this endless traffic. Nevertheless, it seems more likely that the NASA printer never even responded, there's certainly no evidence to support that it did.

Recommendations:

My conclusion is that we have problem with MY.NET.162.41 (no surprise); it was either misconfigured or broken. The alerts for this traffic weren't spaced-out evenly (no rhythm to them), but they did continue around the clock until the morning of 13 November. I saw no other alerts involving MY.NET.162.41 after that time, up until November 25th. Therefore, it's possible this problem has already been corrected. If a report is available concerning the resolution of this issue, I would be interested in seeing it.

Because there is a known related LPRng (port 515) exploit that affects the Linux OS, then MY.NET.162.41 should be investigated for compromise. Also, unless there is a very good reason to allow inbound or outbound traffic on port 515, then ingress and egress filtering should be employed to prevent this. Finally, perhaps most importantly, MY.NET.162.41 triggered several other alerts that could indicate that it has been targeted; "SMB Name Wildcard", "SMB C access", and "EXPLOIT x86 NOOP". This should be considered when examining MY.NET.162.41.

Correlations:

Although other GCIA candidates reported the same alert message, the actual traffic involved was different. For instance, Glenn Larratt addressed this alert in his GCIA practical. In Glenn's case, all of the traffic was between internal hosts only and he concluded the rule was for information gathering only.

Here is a list of the unique alerts, generated more than 100 times, collected during the five-day analysis period; sorted by the number of events generated. As noted earlier, the sections "TOP 10 TALKERS" and "NOTEWORTHY EXTERNAL HOSTS" contain in-depth analysis of many additional detects.

SMB Name Wildcard — Hits: 199.212

Query for all shared resources on windows host. This rule triggers on inbound traffic to UDP port 137 having content consistent with SMB.

```
[**] SMB Name Wildcard [**]
10-19 00:00:24.716494 MY.NET.150.133:3117 -> 172.202.107.2:137
    ...    ...    ...    ...    ...
[**] SMB Name Wildcard [**]
10-23 23:48:46.012708 MY.NET.80.51:1035 -> 162.201.75.118:137
```

[23] Massachusetts Institute of Technology. "LPRng How-To"

alert udp $EXTERNAL any -> $INTERNAL 137 (msg:"SMB Name Wildcard"; content:"CKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA|0000|";)

SMB C access **Hits: 28,546**

Attempted root-level file access.

```
[**] SMB C access [**]
10-19 05:23:32.390269 80.39.184.192:29065 -> MY.NET.190.101:139
    ...    ...    ...    ...    ...
[**] SMB C access [**]
10-23 15:10:37.694240 200.148.95.243:1250 -> MY.NET.190.97:139
```

alert tcp $EXTERNAL any -> $INTERNAL 139 (msg:"NETBIOS SMB C$ access"; flow:to_server,established; content: "|5c|C$|00 41 3a 00|";reference:arachnids,339; classtype:attempted-recon; sid:533; rev:5;)

Rule Source: http://www.snort.org/snort-db/sid.html?sid=533

MY.NET.30.4 activity **Hits: 15,606**

Triggers on all incoming traffic to MY.NET.30.4, except for port 111.

```
[**] MY.NET.30.4 activity [**]
10-19 00:11:09.671927 66.196.72.96:49268 -> MY.NET.30.4:80
    ...    ...    ...    ...    ...
[**] MY.NET.30.4 activity [**]
10-23 23:48:11.561445 68.55.62.79:3163 -> MY.NET.30.4:524
```

alert tcp EXTERNAL any -> MY.NET.30.4 !111 (msg: “MY.NET.30.4 Activity”;)

EXPLOIT x86 NOOP **Hits: 11,563**

Intel x86 internal host received "No Operation" string from external source, possible buffer overflow attempt.

```
[**] EXPLOIT x86 NOOP [**]
10-19 00:33:04.015753 63.232.32.9:3417 -> MY.NET.190.101:135
    ...    ...    ...    ...    ...
[**] EXPLOIT x86 NOOP [**]
10-23 23:35:39.199416 211.125.6.43:4883 -> MY.NET.190.102:135
```

alert tcp $EXTERNAL any -> $INTERNAL any (msg:"EXPLOIT x86 NOOP"; content: "|90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90|"; flags: A+; reference:arachnids,181;)

Rule Source: http://www.geocrawler.com/archives/3/5343/2001/4/50/5487319/

connect to 515 from inside **Hits: 7,131**

Internal host connection attempt on port 515, print spooler port, to external.

```
[**] connect to 515 from inside [**]
10-20 14:04:06.054669 MY.NET.162.41:721 -> 128.183.110.242:515
    ...    ...    ...    ...    ...
[**] connect to 515 from inside [**]
10-23 23:47:43.566468 MY.NET.162.41:721 -> 128.183.110.242:515
```

alert tcp $INTERNAL any -> $EXTERNAL 515 (msg:" connect to 515 from inside";)

MY.NET.30.3 activity Hits: 5,726

Triggers on all incoming traffic to MY.NET.30.3, except for port 111. Same as the similarly-named rule for MY.NET.30.4

```
[**] MY.NET.30.3 activity [**]
10-19 00:01:24.412416 68.57.90.146:1032 -> MY.NET.30.3:524
    ...    ...    ...    ...    ...
[**] MY.NET.30.3 activity [**]
10-23 23:48:00.524549 165.247.82.136:1466 -> MY.NET.30.3:524
```

alert tcp EXTERNAL any -> MY.NET.30.3 !111 (msg: “MY.NET.30.3 Activity”;)

TCP SRC and DST outside network Hits: 4,518

Neither source or destination IP registered on network. See "ICMP SRC and DST outside network" for additional information.

```
[**] TCP SRC and DST outside network [**]
10-19 00:25:00.429117 68.55.0.64:3512 -> 24.33.5.145:80
    ...    ...    ...    ...    ...
[**] TCP SRC and DST outside network [**]
10-23 22:45:08.789634 192.168.0.4:3003 -> 209.10.203.102:6301
```

alert tcp $EXTERNAL any -> $EXTERNAL any (msg:" TCP SRC and DST outside network"; flags:S12; tos:0x2;)

Rule Source: www.giac.org/practical/GCIA/Ashley_Thomas_GCIA.pdf

External RPC call Hits: 3,266

Connection from external host to port 111

```
[**] External RPC call [**]
10-22 06:22:23.515905 81.15.45.1:37255 -> MY.NET.5.5:111
    ...    ...    ...    ...    ...
[**] External RPC call [**]
10-23 10:41:31.967713 193.114.70.169:1666 -> MY.NET.6.15:111
```

alert tcp $EXTERNAL any -> $INTERNAL 111 (msg:“External RPC call”;)

High port 65535 tcp - possible Red Worm - traffic Hits: 3,172

Possible malicious code infection, and TCP traffic seen on port 65535, regardless of whether the traffic is entering, leaving, or staying within the network. 65535 could be the source or destination port, I've created a rule for each possibility. Red Worm is the original name for the worm "Adore"; which exploits a backdoor vulnerability on Linux hosts.

```
[**] High port 65535 tcp - possible Red Worm - traffic [**]10-19
01:28:43.923527 MY.NET.25.71:65535 -> 202.108.32.232:25    ...    ...
...    ...    ...[**] High port 65535 tcp - possible Red Worm - traffic
[**]10-23 23:41:18.490016 66.66.71.92:65535 -> MY.NET.153.94:1213
```

alert TCP any 65535 -> any any (msg:"High port 65535 tcp - possible Red Worm – traffic (source)";) … alert TCP any any -> any 65535 (msg:"High port 65535 tcp - possible Red Worm – traffic (destination)";)

Possible trojan server activity Hits: 2,009

Any inbound traffic to TCP port 27734, associated with SubSeven worm.

```
[**] Possible trojan server activity [**]
10-19 00:38:15.259460 MY.NET.60.17:27374 -> 195.206.49.9:113
    ...    ...    ...    ...    ...
[**] Possible trojan server activity [**]
10-23 21:41:54.963226 MY.NET.84.235:27374 -> 211.167.67.60:80
```

alert TCP $EXTERNAL any -> $INTERNAL 27734 (msg: "Possible trojan server activity"; flags: A+;)

Rule Source: www.giac.org/practical/GCIA/Andre_Cormier_GCIA.pdf

ICMP SRC and DST outside network Hits: 1,825

ICMP traffic seen between external hosts, internal host not involved; indicates possible host compromise or misconfiguration.

alert icmp $EXTERNAL any -> $EXTERNAL any (msg:"ICMP SRC and DST outside network";)

NMAP TCP ping! Hits: 752

Triggers on any inbound ICMP traffic with a data size of zero. Possibly, nmap is being used as a recon tool to gather a list of live hosts.

```
[**] NMAP TCP ping! [**]
10-19 00:44:08.886153 62.80.19.34:80 -> MY.NET.1.5:53
    ...    ...    ...    ...    ...
[**] NMAP TCP ping! [**]
10-23 23:34:12.695301 64.152.70.68:53 -> MY.NET.1.3:53
```

alert icmp $EXTERNAL any -> $INTERNAL any (msg:"ICMP PING NMAP"; dsize: 0; itype: 8; reference:arachnids,162; classtype:attempted-recon; sid:469; rev:1;)

Rule Source: http://www.snort.org/snort-db/sid.html?sid=469

SUNRPC highport access! **Hits: 494**

Any traffic directed internally destined to tcp port 32771.

```
[**] SUNRPC highport access! [**]
10-19 00:07:50.096399 66.218.84.150:80 -> MY.NET.97.76:32771
    ...    ...    ...    ...    ...
[**] SUNRPC highport access! [**]
10-23 23:09:45.273773 67.104.77.37:80 -> MY.NET.97.122:32771
```

alert tcp any any -> $INTERNAL 32771 (msg: "SUNRPC highportaccess!";)

Null scan! **Hits: 455**

Triggers if TCP traffic has no flags set.

```
[**] Null scan! [**]
10-19 00:04:33.349368 67.119.232.52:21260 -> MY.NET.12.4:110
    ...    ...    ...    ...    ...
[**] Null scan! [**]
10-23 23:21:47.909349 67.119.234.194:36109 -> MY.NET.12.4:110
```

alert tcp any any -> $INTERNAL any (flags: 0; msg:"NULL Scan";)

High port 65535 udp - possible Red Worm - traffic **Hits: 438**

Inbound UDP traffic using 65535 as source or destination port; possible BIND exploit attempt. See "High port 65535 tcp - possible Red Worm - traffic" for additional information.

```
[**] High port 65535 udp - possible Red Worm - traffic [**]
10-19 12:06:51.098507 212.50.160.100:65535 -> MY.NET.1.4:53
    ...    ...    ...    ...    ...
[**] High port 65535 udp - possible Red Worm - traffic [**]
10-23 23:45:58.688430 220.144.202.134:65535 -> MY.NET.130.157:8450
```

alert UDP $EXTERNAL any -> $INTERNAL 65535 (msg: " High port 65535 udp - possible Red Worm – traffic ";)

alert UDP $EXTERNAL 65535 -> $INTERNAL any (msg: " High port 65535 udp - possible Red Worm – traffic ";)

[UNI NIDS IRC Alert] IRC user /kill detected, **Hits: 342**

This alert triggers on inbound traffic in established connection with source ports ranging from 6663-7000 where "/kill" is included in the content, case is insensitive.

alert tcp $EXTERNAL 6663:7000 -> $INTERNAL any (msg: "IRC user /kill detected, possible trojan."; flow: established; content: "/kill"; nocase;)

[UNI NIDS IRC Alert] XDCC client detected attempting IRC Hits: 182

This alert warns of a peer-to-peer connection attempt from an internal host to and external host.

```
[**] [UNI NIDS IRC Alert] XDCC client detected attempting to IRC [**]
10-23 05:02:26.173975 MY.NET.29.2:1097 -> 69.55.239.14:6667
    ...    ...    ...    ...    ...
[**] [UNI NIDS IRC Alert] XDCC client detected attempting to IRC [**]
10-23 23:48:37.569757 MY.NET.80.16:1580 -> 64.157.246.22:6667
```

alert tcp $INTERNAL any -> $EXTERNAL 6667 (msg: "[UNI NIDS IRC Alert] XDCC client detected attempting to IRC"; flow: established;)

FTP passwd attempt   Hits: 150

Inbound TCP traffic to port 21 (FTP) having the "A+" flag set and "passwd" in the content.

```
[**] FTP passwd attempt [**]10-19 01:05:28.713131 199.243.85.90:51435 ->
MY.NET.24.47:21    ...    ...    ...    ...    ...[**] FTP passwd attempt
[**]10-23 22:20:49.699881 24.188.211.202:27810 -> MY.NET.24.47:21
```

alert tcp $EXTERNAL any -> $INTERNAL 21 \(msg:"FTP passwd attempt";flags: A+; content:"passwd";)

Rule Source: www.raid-symposium.org/raid2001/
papers/eckmann_raid2001.pdf

[UNI NIDS] External MiMail alert   Hits: 103

Internal host possibly infected with malicious code, e-mail address thievery and DoS possible. Rule looks for outbound TCP port 80 traffic to a specific destination address.

```
[**] [UNI NIDS] External MiMail alert [**]
10-19 05:04:18.457261 211.111.104.129:1506 -> MY.NET.12.6:25
    ...    ...    ...    ...    ...
[**] [UNI NIDS] External MiMail alert [**]
10-23 21:38:22.948247 68.48.163.134:1156 -> MY.NET.12.6:25
```

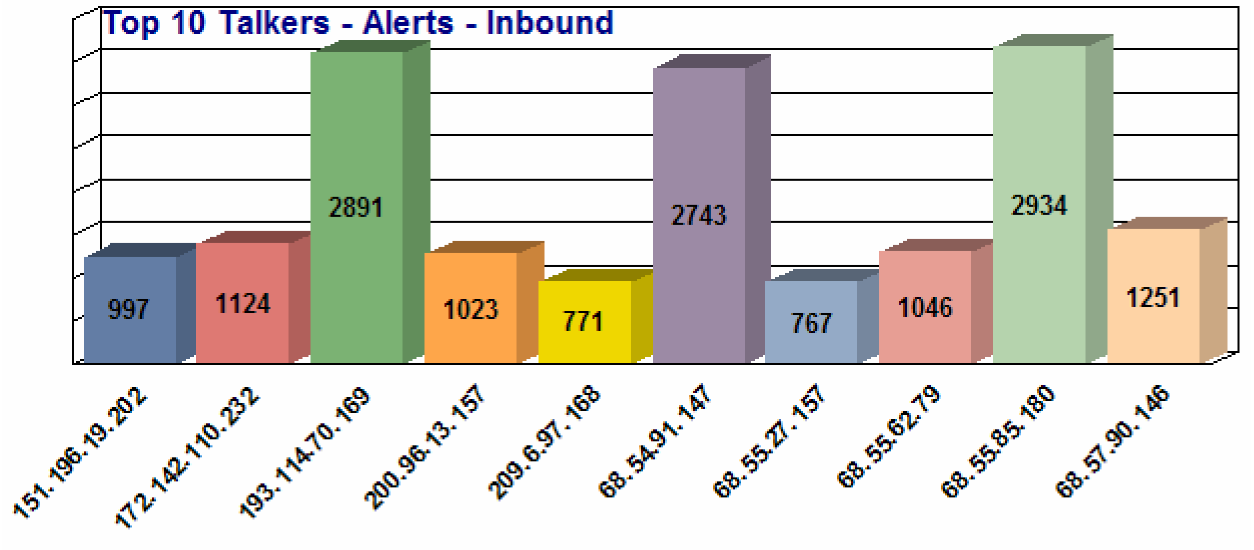
alert tcp any any -> 63.219.179.210 80 (msg:"Infected with new MiMail"; classtype:policy-violation; sid:10000003; rev:1;)

Rule Source:
http://sourceforge.net/mailarchive/message.php?msg_id=6439594 - virus info
from: http://vil.nai.com/vil/content/v_100795.htm

TOP 10 TALKERS: The Hosts with the Most

Presented in this section are bar charts displaying the 10 most active hosts, in terms of activity in the alert, oos, and scan logs. Analysis of the activity accompanies each chart along with recommendations for action where appropriate. Please refer to Section 4 for descriptions of alerts and additional statistics.

TOP 10 TALKERS - ALERTS - INBOUND



This chart shows the top 10 talkers in the alert logs and includes only external hosts entering the network. Most of the alerts raised by these hosts were "MY.NET.30.3 activity" and "MY.NET.30.4 activity". Three of theses hosts did generate more interesting alerts.

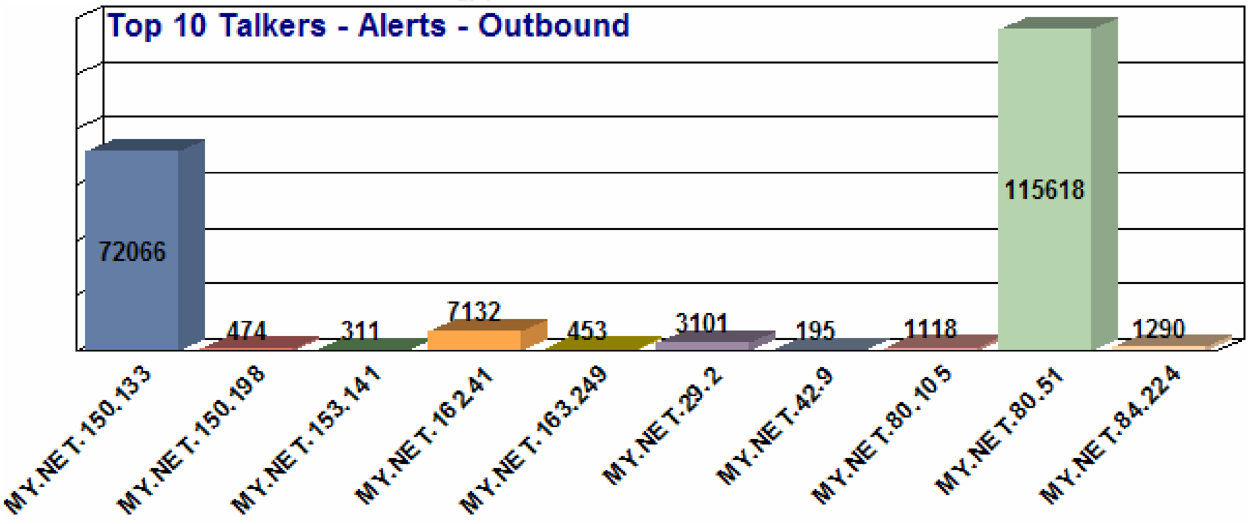
193.114.70.169, registered to "FIRST-PROCUREMENT-ASSOCIATES-LIMITED" in the United Kingdom, is performing reconnaissance and attempting to gain access to the University's network. Most of the alerts generated by this host are "External RPC call" and "NETBIOS NT NULL session". "External RPC call" means the source host is sending packets to TCP port 111 in an attempt to determine what RPC services are running, "touching" 1592 of the University's computers. The "NETBIOS NT NULL session" activity from 193.114.70.169 wasn't as far-reaching, but is much more suspect, only 10 hosts were targeted, one in particular. "MY.NET.111.226" (MY.NET.111.226) was hit the most and, according to a reverse DNS lookup, its host name is "sutherin-oracle.theuniversity.edu". The source host seems very interested in this machine, likely because they believe it is a database server. The other nine hosts are MY.NET.29.12 (cf1.theuniversity.edu), MY.NET.29.18 (educ1.theuniversity.edu), MY.NET.29.24 (db1.theuniversity.edu), MY.NET.29.5 (bb-db4.theuniversity.edu), MY.NET.29.8 (cms.theuniversity.edu), MY.NET.30.83 (apollo.theuniversity.edu), MY.NET.5.26

(umbbal01srv01.theuniversity.edu), MY.NET.5.34 (bb-mig.theuniversity.edu), and MY.NET.5.55 (kai.theuniversity.edu). Note that several of these have hosts names that would lead one to think they are database servers or are of some other high importance. Computers named after gods, "apollo" in this case, or with names like "bal01srv01"a (possibly a load-balancing server) are often high-value assets and valued targets by attackers.

Source host 200.96.13.157, registered to "Comite Gestor da Internet" in Brazil, is possibly attempting to pass along malicious code to MY.NET.80.105; all of the external host's traffic was to this host. The external host (200.96.13.157) as well as the MY.NET.80.105 triggered more than 1,000 instances each of the alert ""High port 65535 tcp - possible Red Worm - traffic"". "Red Worm" (aka "Adore") is a trojan that exploits a backdoor vulnerability. Based on these alerts, MY.NET.80.105 is surely infected with Code Red, presuming it is running the Linux operating system. External host 209.6.97.168, registered to "RCN Corporation", is responsible for triggering 764 alerts for "EXPLOIT x86 NOOP" involving half a dozen of the University's computers. The external host may be attempting to exploit a buffer overflow vulnerability. This vulnerability only affects hosts with the x86 (Intel) platform. I saw no previous fingerprinting generated by the source, so it is likely the attacker does not know whether or not their efforts will likely be successful. However, all of the traffic from 209.6.97.168 is directed at just a few hosts, so perhaps the attacker knows something about the targets.

Recommendation: Block all traffic from 193.114.70.169, 200.96.13.157, and 209.6.97.168 at the network's perimeter. MY.NET.80.105 should be taken off-line immediately and checked for the presence of malicious code.

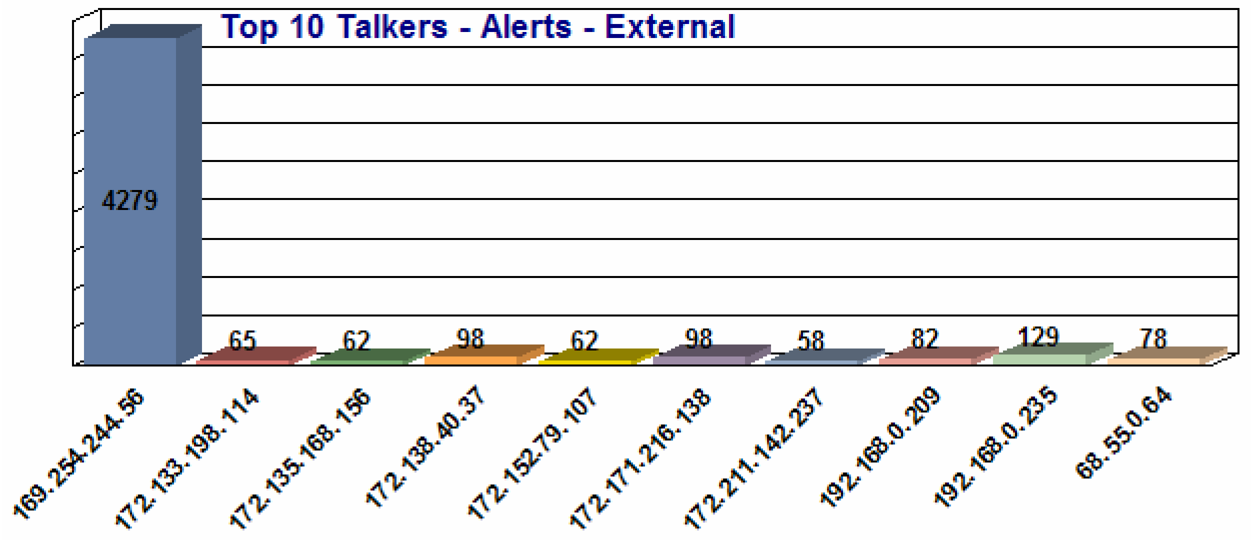
TOP 10 TALKERS - ALERTS - OUTBOUND



Here we have the top 10 talkers belonging to the University and their outbound traffic. MY.NET.150.133 (MY.NET.150.133) and MY.NET.80.51 (MY.NET.80.51) are responsible for triggering 187,684 instances of "SMB Name Wildcard" against 129,357 external hosts; querying for all shared resources. MY.NET.150.133' and MY.NET.80.51 addresses have been spoofed and/or they may have been comprised and are being used by an attacker to do their dirty work.

Recommendation: Traffic involving MY.NET.150.133 and MY.NET.80.51 deserves further scrutiny to determine the nature of their activity so appropriate action can be taken.

TOP 10 TALKERS - ALERTS - EXTERNAL



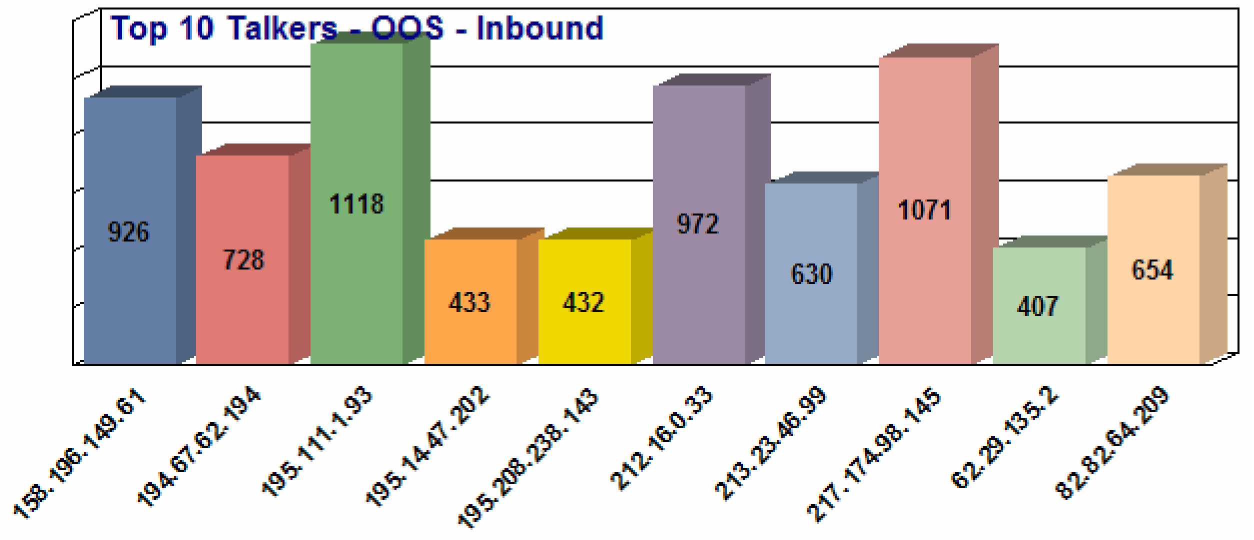
The ten source hosts above are employing IP addresses not within the range of the University's network and the destinations for their traffic is also outside the network. This is not normal and one of two things is happening: Network hardware has been misconfigured or an internal host has been compromised and is being used, along with a spoofed IP, to carry out whatever the intruder wishes. All of these hosts generated only two distinct alerts: "TCP SRC and DST outside network" and "ICMP SRC and DST outside network".

A whois query for 169.254.244.65, the most active of those shown, reveals that it is assigned to blackhole-2.iana.org. This means that this IP is within the range of addresses reserved for networking hardware for consumer use. It's a valid IP, but should only be seen within someone's home network. Several other hosts have IP addresses in the "black hole" range as well.

No other alerts were triggered by any of the IPs that generated the "TCP SRC and DST outside network" and "ICMP SRC and DST outside network". This doesn't mean that nothing else suspicious occurred, it just means we have no record of it, there's a difference; because of Snort's "first match" policy, rules listed after these two would not be processed. The traffic with IPs within the "black hole" range could be due to router misconfiguration. However, an attacker may choose one of these addresses because it can be harder to trace; a whois query or reverse DNS lookup will only lead to one of IANA's black hole servers. The offending hosts that do not have IPs in the black hole range are no less suspect.

Recommendation: Block all outbound traffic with IP addresses not belonging to the University. Inspect routers for misconfiguration. Determine the actual hosts involved; if the MAC address can be obtained for each host, then they can be tracked down this way. Reprocessing the logs with Snort using the -X option will yield the MAC address. Configuration Management databases or an older set of logs will help in determining the actual IP address of the host. Once these machines are found, they should be checked for compromise; this should be done for all hosts causing this alert, not just the ten listed above.

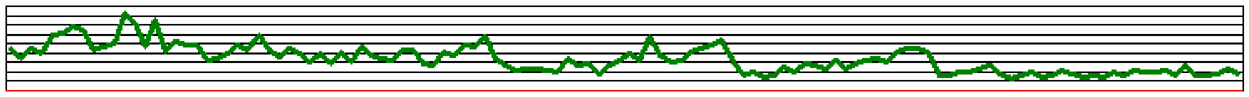
TOP 10 TALKERS - OOS - INBOUND



These are the top 10 hosts responsible for sending out of specification (OOS) packets to the University's network; no internal hosts transmitted OOS packets to the outside. OOS infers that the combination of TCP flags and/or options is invalid or at least unusual. This occurs for two reasons: 1) The packets were corrupted in transit, which is not unheard of, but fairly rare. 2) The packets were crafted, using a tool such as nmap, and sent in order to elicit a response from the targeted host. Various operating systems respond to these oddball packets in their own way. So, based on the content of a target's reply, an attacker gain make suppositions about the OS. This activity is known as "operating system fingerprinting".

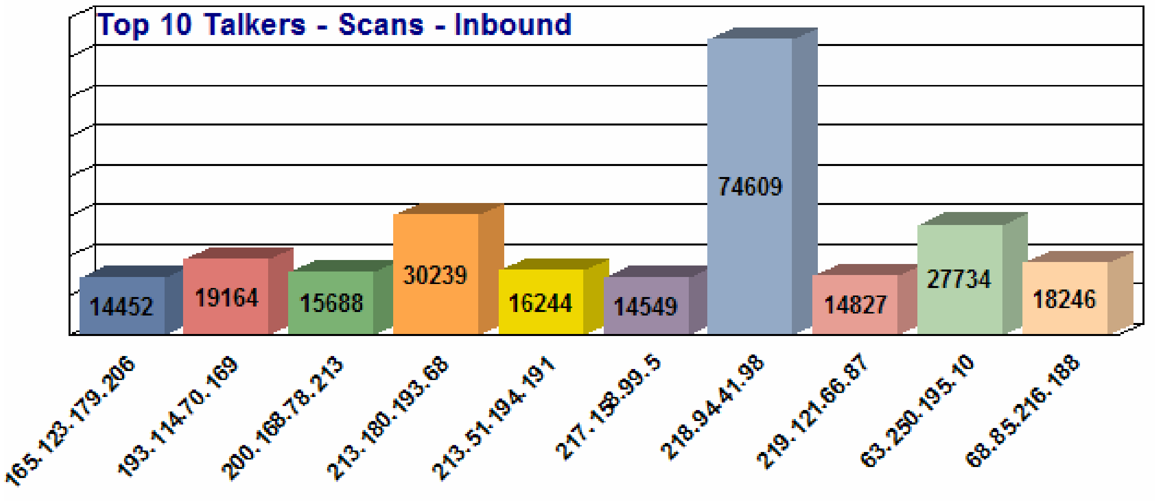
There's a very informative article relating to OS fingerprinting posted here: http://www.insecure.org/nmap/nmap-fingerprinting-article.html. The article was written by "Fyodor", who had a hand in the creation of nmap.

OOS packets occur naturally on the Internet, but without any set pattern or time cycle. OOS packets were spread out throughout the five day period of analysis, so I don't believe this traffic represents what should be accepted as normal; we're seeing a fairly steady flow of OOS traffic, as illustrated in the line graph below.



Recommendation: Consider blocking traffic from any hosts that have been known to send OOS packets. Consider blocking all incoming OOS traffic.

TOP 10 TALKERS - SCANS - INBOUND



Finally, we have the top 10 talkers in terms of scans directed against the University's network. Among these, 218.94.41.98 is by far the most prolific. A whois query reveals this IP is assigned to "CHINANET-JS". All of this host's activity occurred within a very short time span at around 6 AM on October 22nd. In addition to the scanning activity, this host generated about 30 alerts, all of them "MY.NET.30.3 activity" and "MY.NET.30.4 activity", but generated no OOS entries.

218.94.41.98 probed about 50 different ports; eight of them received the bulk of the traffic, about 8300 hits each to ports 2000, 3128, 4128, 5128, 8000, 8080, 8888, and 9090. These ports are associated with a number of Trojans and backdoors, very undesirable things:

2000:	A-trojan, Fear, Force, GOTHIC Intruder, and many more
3128:	RingZero and Reverse WWW Tunnel Backdoor
4128:	RedShad (remote access)
5128:	Squid
8000:	session hijacking
8080:	RingZero
8888:	Dark IRC
9090:	Aphex (packet sniffer)

And, this is just the first few port's the source was targeting; hardly seems innocuous.

Recommendations: 218.94.41.98 covered a lot of ground in a very short time. Chances are they got what they were looking for, but will be back to cause more damage later once they've had an opportunity to decide which of the University's assets to attack. 218.94.41.98 should be blocked at the network's perimeter to prevent its return.

FIVE NOTEWORTHY EXTERNAL HOSTS: Registration information and in-depth analysis of hosts seen "across the board".

SUMMARY

These five hosts were selected because they are special. Among the 8628 unique hosts accounted for, these stand out. These five, and only these, appear in all 3 types of logs files analyzed, alerts, oos, and scans, during the five days' of logs analyzed, 19-23 October.

In the alert logs, none of the IPs were seen in more than one day's traffic. Only one was seen on the 19th, three on the 20th, and another on the final day, the 23rd. There was no alert activity from any of the selected IPs on the 21st or 22nd. Also, all of these hosts triggered their own distinct set of alerts, meaning none of the hosts caused any alerts that were also triggered by another. 68.85.216.188 caused three different alerts, while the other four hosts only trigged one distinct alert each.

In the OOS logs, only a single host, 195.101.94.208, appeared on all five days. The other hosts were seen in no more than two days of OOS logs.

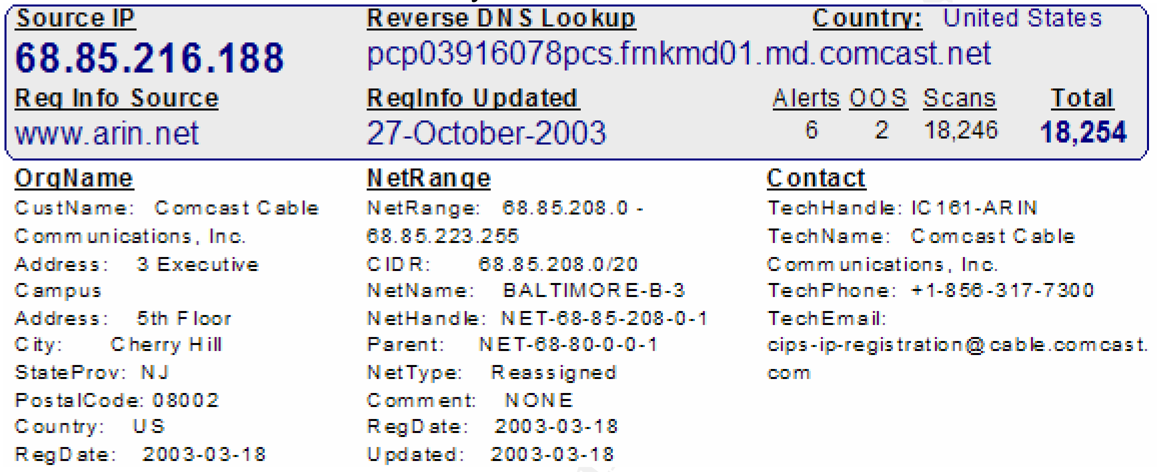
In the scan logs, 195.101.94.208 is of interest because it's the only host involved in scan activity on every day analyzed; as it was in the OOS logs. 213.186.35.9 was seen on three consecutive days. Other hosts, of these five, were seen on no more than two days.

The registration information for the five selected external source IP addresses, along with the source for the registration info and the date it was updated, are covered in this section. Also included are reverse DNS lookup, country of origin, and the activity for each of these hosts as recorded in the alerts, oos, and scan logs. The information for each host is preceded by a summary of its activity and recommendations for action.

68.85.216.188: With more than 18,000 "hits", most of it scanning, this host was by far the busiest of all five reviewed. 68.85.216.188 generated three distinct alerts, "DDOS mstream client to handler", "High port 65535 tcp - possible Red Worm - traffic", and "SUNRPC highport access!"; one instance of the first two, four for the last. Important: 100% of 68.85.216.188's efforts are directed to a single internal host, MY.NET.24.34 (MY.NET.24.34); already identified as the University's web server, www.theuniversity.edu. Having performed a reverse DNS lookup on the external host, I did an Internet search via www.google.com for "pcp03916078pcs.frnkmd01.md.comcast.net", but this search proved fruitless.

However, a search for “frnkmd01.md.comcast.net” returned a large number of results, many referencing blacklisting of a host.

Recommendations: I recommend blocking this external host’s traffic from entering the network. Also, the University’s web server (MY.NET.24.34) should be checked for any possible malicious activity, including intrusion and viruses and to make sure the latest security patches have been applied. Clearly, www.theuniversity.edu has been targeted by 68.85.216.188. Step must be taken to keep this external host away from the network and to assure MY.NET.24.34’s safety.

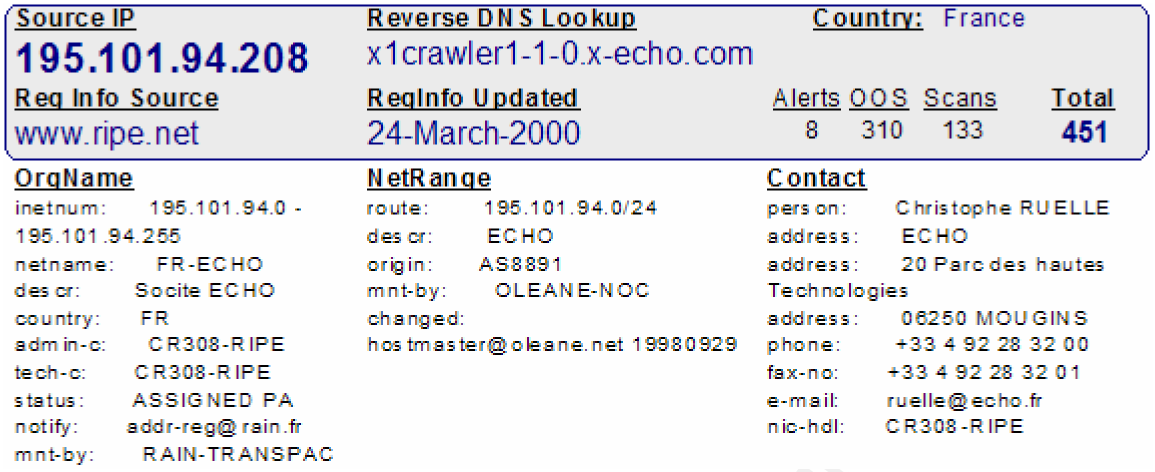


Source IP	Reverse DNS Lookup	Country: United States			
68.85.216.188	pcp03916078pcs.frnkmd01.md.comcast.net				
Reg Info Source	**RegInfo Updated**	Alerts	OOS	Scans	**Total**
www.arin.net	27-October-2003	6	2	18,246	**18,254**

OrgName	NetRange	Contact
CustName: Comcast Cable Communications, Inc. Address: 3 Executive Campus Address: 5th Floor City: Cherry Hill StateProv: NJ PostalCode: 08002 Country: US RegDate: 2003-03-18	NetRange: 68.85.208.0 - 68.85.223.255 CIDR: 68.85.208.0/20 NetName: BALTIMORE-B-3 NetHandle: NET-68-85-208-0-1 Parent: NET-68-80-0-0-1 NetType: Reassigned Comment: NONE RegDate: 2003-03-18 Updated: 2003-03-18	TechHandle: IC161-ARIN TechName: Comcast Cable Communications, Inc. TechPhone: +1-856-317-7300 TechEmail: cips-ip-registration@cable.comcast.com

195.101.94.208: This host scanned the University’s network and was responsible for OOS traffic on all five days analyzed. As mentioned, this is the only host of the five detailed in this section with such consistent activity. So, I was looking forward to seeing what this host was up to. 195.101.94.208 caused only a single alert a total of 8 times. The alert was “MY.NET.30.4 activity”. Since the alert has already been identified as probably being for informational use only, we have no evidence in the alert logs that 195.101.94.208 was up to anything malicious. I must admit, I was a bit disappointed. I thought, surely, with the previous days’ activity, something of note would have turned up.

Recommendation: It may be wise to add 195.101.94.208 to the University’s “watch list” for future activity.

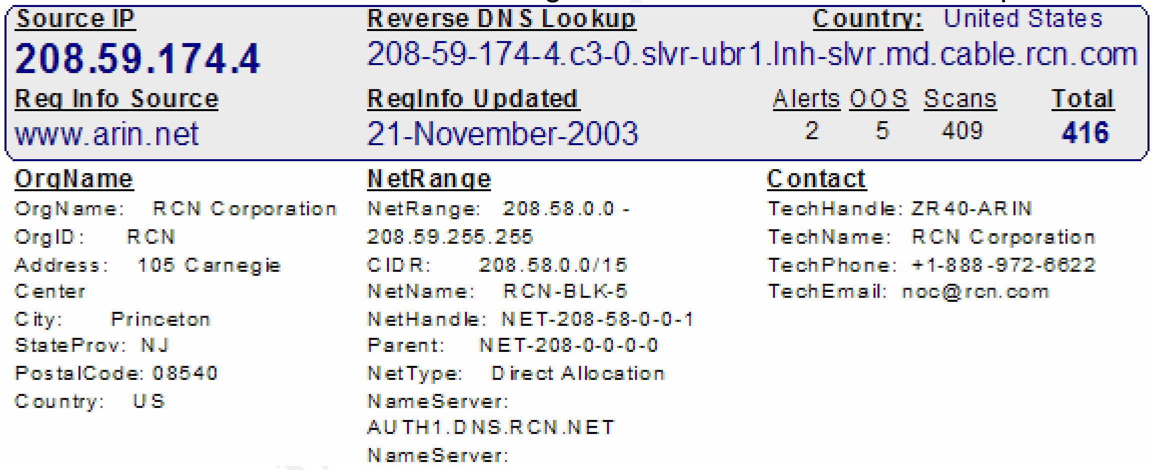


Source IP	Reverse DNS Lookup	Country: France			
195.101.94.208	x1crawler1-1-0.x-echo.com				
Req Info Source	ReqInfo Updated	Alerts	OOS	Scans	Total
www.ripe.net	24-March-2000	8	310	133	451

OrgName	NetRange	Contact
inetnum: 195.101.94.0 - 195.101.94.255	route: 195.101.94.0/24	person: Christophe RUELLE
netname: FR-ECHO	descr: ECHO	address: ECHO
descr: Socite ECHO	origin: AS8891	address: 20 Parc des hautes Technologies
country: FR	mnt-by: OLEANE-NOC	address: 06250 MOUGINS
admin-c: CR308-RIPE	changed: hostmaster@oleane.net 19980929	phone: +33 4 92 28 32 00
tech-c: CR308-RIPE		fax-no: +33 4 92 28 32 01
status: ASSIGNED PA		e-mail: ruelle@echo.fr
notify: addr-reg@rain.fr		nic-hdl: CR308-RIPE
mnt-by: RAIN-TRANSPAC		

208.59.174.4: Along with more than 400 scan entries, this external host generated alert "NMAP TCP ping!" twice. Usually this alert indicates the source host is attempting to see if the destination is "alive". Similar to 68.85.216.188, all of this host's efforts were directed toward the University's web server.

Recommendation: I recommend blocking 208.59.174.4 at the network's perimeter.

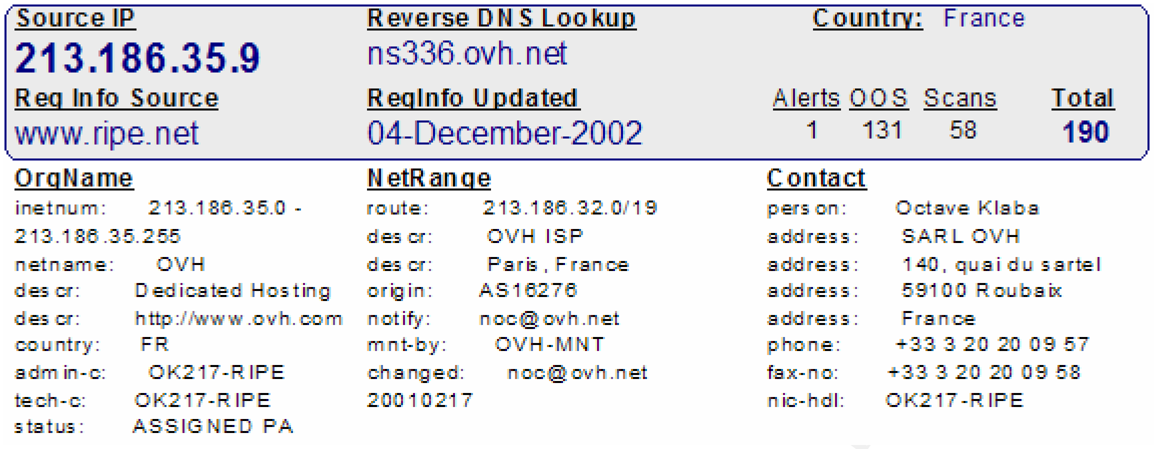


Source IP	Reverse DNS Lookup	Country: United States			
208.59.174.4	208-59-174-4.c3-0.slvr-ubr1.lnh-slvr.md.cable.rcn.com				
Req Info Source	ReqInfo Updated	Alerts	OOS	Scans	Total
www.arin.net	21-November-2003	2	5	409	416

OrgName	NetRange	Contact
OrgName: RCN Corporation	NetRange: 208.58.0.0 - 208.59.255.255	TechHandle: ZR40-ARIN
OrgID: RCN	CIDR: 208.58.0.0/15	TechName: RCN Corporation
Address: 105 Carnegie Center	NetName: RCN-BLK-5	TechPhone: +1-888-972-6622
City: Princeton	NetHandle: NET-208-58-0-0-1	TechEmail: noc@rcn.com
StateProv: NJ	Parent: NET-208-0-0-0-0	
PostalCode: 08540	NetType: Direct Allocation	
Country: US	NameServer: AUTH1.DNS.RCN.NET	
	NameServer:	

213.186.35.9: This external host was responsible for a low level of scanning and generated one instance of the alert "[UNI NIDS IRC Alert] IRC user /kill detected". This alert triggers on inbound TCP traffic to ports 6660-7000. None of this needs to be of great concern at this time.

Recommendation: I recommend no further action regarding this external host, with the exception of possibly adding it to the watch list.

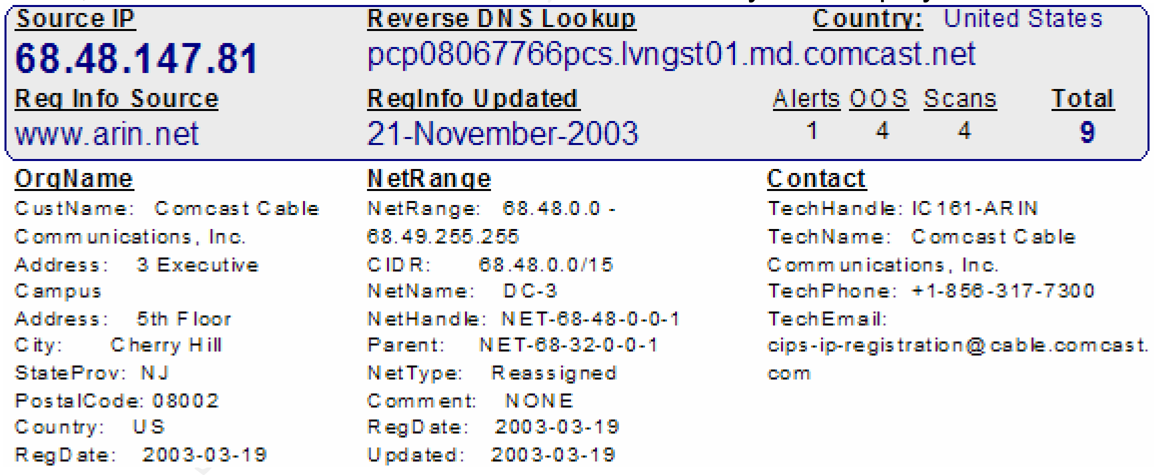


Source IP	Reverse DNS Lookup	Country: France			
213.186.35.9	ns336.ovh.net				
Req Info Source	**RegInfo Updated**	**Alerts**	**OOS**	**Scans**	**Total**
www.ripe.net	04-December-2002	1	131	58	190

OrgName	NetRange	Contact
inetnum: 213.186.35.0 - 213.186.35.255	route: 213.186.32.0/19	person: Octave Klaba
netname: OVH	descr: OVH ISP	address: SARL OVH
descr: Dedicated Hosting	descr: Paris, France	address: 140, quai du sartel
descr: http://www.ovh.com	origin: AS16276	address: 59100 Roubaix
country: FR	notify: noc@ovh.net	address: France
admin-c: OK217-RIPE	mnt-by: OVH-MNT	phone: +33 3 20 20 09 57
tech-c: OK217-RIPE	changed: noc@ovh.net 20010217	fax-no: +33 3 20 20 09 58
status: ASSIGNED PA		nic-hdl: OK217-RIPE

68.48.147.81: This host was the least active of the five, generating only 9 hits. OOS traffic was seen against MY.NET.24.34 (www.theuniversity.edu) and MY.NET.29.3 (bb-app4.theuniversity.edu). A single instance of the alert "Probable NMAP fingerprint attempt" involved internal host MY.NET.24.33; my.theuniversity.edu based on a reverse DNS lookup. While 68.48.147.81's activity doesn't seem overly dangerous, I am concerned about activity directed by several external hosts toward MY.NET.24.33 (my.theuniversity.edu). Eighteen alerts were generated with my.theuniversity.edu as the target, all involving malicious code.

Recommendation: I recommend no action for the external host, 68.48.147.81. However, MY.NET.24.33 should be checked immediately for foul play.



Source IP	Reverse DNS Lookup	Country: United States			
68.48.147.81	pcp08067766pcs.lvngst01.md.comcast.net				
Req Info Source	**RegInfo Updated**	**Alerts**	**OOS**	**Scans**	**Total**
www.arin.net	21-November-2003	1	4	4	9

OrgName	NetRange	Contact
CustName: Comcast Cable Communications, Inc.	NetRange: 68.48.0.0 - 68.49.255.255	TechHandle: IC161-ARIN
Address: 3 Executive Campus	CIDR: 68.48.0.0/15	TechName: Comcast Cable Communications, Inc.
Address: 5th Floor	NetName: DC-3	TechPhone: +1-856-317-7300
City: Cherry Hill	NetHandle: NET-68-48-0-0-1	TechEmail: cips-ip-registration@cable.comcast.com
StateProv: NJ	Parent: NET-68-32-0-0-1	
PostalCode: 08002	NetType: Reassigned	
Country: US	Comment: NONE	
RegDate: 2003-03-19	RegDate: 2003-03-19	
	Updated: 2003-03-19	

DEFENSIVE RECOMMENDATIONS:

OVERVIEW

The University has employed an Intrusion Detection System (IDS) for some time now. However, more attention needs to be given to the care and feeding of the IDS. Snort is currently in use, but the version seems to be outdated. Upgrading to the most-recent stable version of Snort should be high on the list of priorities. Also, the Snort ruleset in-place is also out of date and many rules could use fine-tuning.

There's a lot of traffic triggering alerts that should never be allowed to cross the network's perimeter. The subject of ingress/egress filtering has been touched upon several times in this report; it's important and the effort expended to implement it is worthwhile.

Additionally, most of the alerts generated seem to be for events not related to security, but rather for reporting. Gathering statistics can be very important to any organizations operation. However, this should not be done to the detriment of the network's defense. Other means of collecting statistical data should be explored. In the meantime, any Snort rule not purely related to safeguarding the network should be modified to push the needed information to logs; rather than generating alerts.

Plenty of alerts are being generated at a fast pace. So much data can burden an analyst. After all, "data" is not "information". It's up to the analyst to grind through the data in order to get to the information. There are a number of tools that can make the work of the analyst easier. Snort can be integrated with MySQL; alert logs are pushed directly to a relational database for analysis instead of plain text logs files that must be scoured by eye. Joining Snort with even the most basic datamining tool can greatly improve the analyst's efficiency. The benefits are covered in more detail in "ANALYSIS METHODOLOGY AND LESSONS LEARNED".

The University's current network configuration management program (CM) could use improvement. Things such as network diagrams and host configuration should be readily available, as they are essential to any security plan. The effort of performing a security log audit would be eased greatly by a comprehensive CM database. You can't know if what you have is secure if you don't know what you have to begin with.

ACTION ITEMS

INTERNAL HOSTS

Numerous internal hosts were identified earlier in this report as needing attention. These hosts are listed below along with the type of action needed.

These hosts should be investigated for possible compromise: MY.NET.162.41, MY.NET.191.228, MY.NET.84.228, and MY.NET.24.47.

These hosts should be check for misconfiguration, malware, or tampering: MY.NET.162.41, MY.NET.24.47, 68.42.146.143, and MY.NET.80.105.

Monitor the following hosts for unauthorized activity: MY.NET.150.133 and MY.NET.80.51.

Resource sharing should be disallowed either technically or by policy except when absolutely necessary.

EXTERNAL HOSTS

These external hosts listed below have brought harm to the University's network and should be blocked at the perimeter: 218.94.41.98, 193.114.70.169, 200.96.13.157, and 209.6.97.168.

NETWORK PERIMETER: Things that can be done at the network's perimeter to improve security:

Block traffic from any hosts known to consistently send OOS packets. Consider blocking all incoming OOS traffic.

Block external hosts that regularly cause the alert "FTP passwd attempt".

Block all outbound traffic with IP addresses not belonging to the University.

Consider blocking these ports until further research on the hazards listed above can be completed: 2000, 3128, 4128, 5128, 8000, 8080, 8888, and 9090.

Employ ingress and egress filtering of port 515, unless required.

Implement ingress filtering to block TCP port 139 traffic at the network's perimeter.

Block incoming/outgoing traffic on port 137.

Tune Snort rules relating to port 137 so they do not trigger on internal traffic.

Change Snort rules intended for informative purposes from "alert" to "log".

Inspect routers for misconfiguration.

MISCELLANEOUS

Finally: Alert, OOS, and scan logs file do not all use the same date format and the year is not included. Using a consistent timestamp format across all types of logs would ease the effort of correlating events. As for the exclusion of the year in the timestamp: Without it, some confusion could be caused when reviewing logs and it's more difficult to combine logs from different years for analysis. Also, problems could arise in regards to forensics and other investigations; if the logs don't record the year of an event, they may not be admissible in court or similar proceedings.

ANALYSIS METHODOLOGY AND LESSONS LEARNED

The means used to perform the analysis for this security audit are well-covered throughout this the paper. Rather than recapping here, this space will be used to describe some of the difficulties which presented themselves and how these challenges were met, along with lessons learned during the process.

The greatest obstacle of this endeavor proved to be the sheer magnitude of the data requiring analysis. I was fully prepared to use my past experience, along with knowledge gained from completing the GCIA Intrusion Detection In-Depth course materials, to perform a comprehensive assessment of the University's network defense. However, I wasn't ready for logs files with a combined weight of a gigabyte containing more than 12 million events of potential interest. Clearly, just sifting through logs of this size would be unfeasible. This called for getting "relational".

The best way to organize logs of such volume is to bring them a relational database. Because I'd already setup a database server to support Part 1 of this assignment, using this same arrangement seemed logical. So, Microsoft SQL Server 2000 would be the datamining tool of choice. Crystal Reports version 9, from Crystal Decisions, was used as a front-end to MS-SQL; to render information to the screen, printer, file, or web. Visio 2000, another Microsoft product, proved useful for creating data-driven network diagrams and a link graph.

Actually getting all of the logs into the database would prove to be a formidable task unto itself. Aside from being huge, the logs aren't in a format that will readily import into a database. And, each log type is in a slightly different format.

So, I'd need to do some clever string manipulation to parse the logs into something useable. The tool I chose is TextTools, a windows-based text editor with powerful string manipulation features that can be automated, has a good GUI, and can be run from a command line. This software can be downloaded from http://www.FireflySoftware.Com and is recommended in the GCIA study guide which can be found here: http://www.giac.org/gcia_study_guide_v33.pdf.

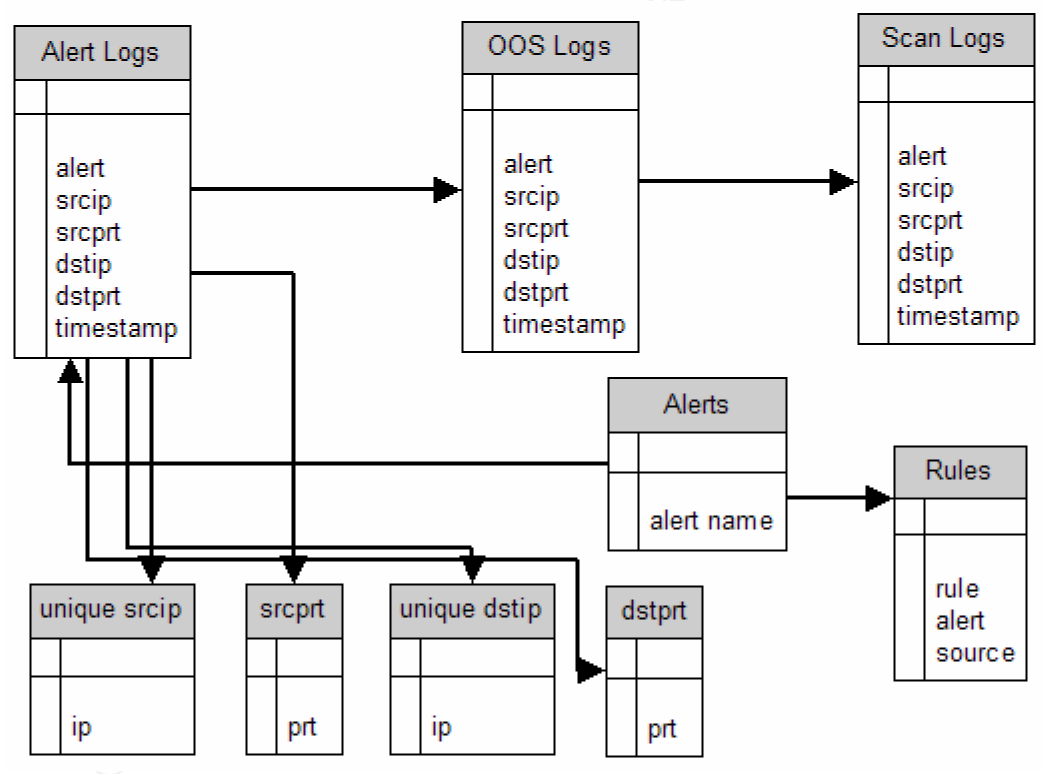
Using TextTools' scripting language, I was able to convert the alert and OOS logs into a comma-delimited format; supported by most any database, including MS-SQL. I wasn't as lucky with the scan logs; their huge size caused my computer to choke. Sounds like a job for perl. Rather than create a perl script from scratch, I was sure another analyst must have done this already. I found just what I needed in GCIA practicals from Les Gordon and Gary Morris; with just a little fine-tuning, their scripts worked great. Next, I simply imported the files I'd processed into my database.

Once all of the log data was parsed and imported, a bit of cleaning up was needing to be done; internal host IP addresses were changed from "MY.NET" to ###.## for cross-comparing/analyzing across the alert, oos, and scan logs. Also, a small number of corrupt records were discarded.

Now that I was on a roll, I decided to not stop at just having the logs in the database. During the course of this project, every piece of relevant information was added to the database. Alert definitions, Snort rules, results from whois queries and reverse DNS lookups, related CERT Advisories, websites, links to GCIA student practicals, and much more, were all integrated into the database.

The result was a powerful datamining tool. Even a simple spreadsheet will help you answer a question like "How many hits did we have per alert?". That's useful; no security audit would be complete without it. However, I wanted to be able to make much more complex - and useful - demands. Example: "Give me a list of external hosts that caused alerts to be triggered that have recently probed the network. And, show me what the rule looks like and a link to a website that discusses the vulnerability." A tall order for sure, but the system I'd setup would make this easy.

The simplified schema for this database would look something like this:



I created an array of SQL queries and scripts to let me dig as deep as I needed to provide the in-depth analysis required to complete this project. The database came in handy for one the thing I never anticipated; how could I easily get snippets of the alerts to include in this report without having to search through those lengthy files manually and copy and paste? The answer: Create an SQL script to reassemble the data back into the format needed for the examples. I'll leave you with this script.

```
-- creating the 'look' of a Snort alert from the log tables
--   The goal is to convert what you've parsed and imported into a database BACK into what
--      you're used to seeing for alerts

--just set this value to whatever you like, an alert name, source IP address, destination port, etc
declare @wear as varchar(255) set @wear = "MY.NET.30.3 Activity"

--  first, drop the "temp" table we'll use for the bottom half of the results
if exists (select * from sysobjects where id = object_id("alerts_bottom_few") and sysstat & 0xf = 3)
            drop table alerts_bottom_few

-- next, grab the last X records and put them in a new table called "alerts_bottom_few"
select top 3 * into alerts_bottom_few  from alerts  where alert = @wear
 order by cast(mn as varchar(2)) + "-" + cast(da as varchar(2)) + " " + cast(tm as varchar(8)) + "." +  ms
desc

--  gather the results from the alerts and alerts_bottom_few table, properly formatted
SET NOCOUNT ON -- results only, no record count should be displayed
select top 3 "      [**] " + alert + " [**] " + "  " + cast(mn as varchar(2))  + "-" +  cast(da as varchar(2))  + " " +
cast(tm as varchar(8))  + "." +  ms + " " +  + srcip + ":" + srcprt  + " -> "  + dstip + ":" + dstprt as " "
 from alerts where alert = @wear and ( " [**] " + alert + " [**] " + " " + cast(mn as varchar(2))  + "-" +
cast(da as varchar(2))  + " " +  cast(tm as varchar(8))  + "." +  ms + " " +  + srcip + ":" + srcprt  + " -> "  +
dstip + ":" + dstprt   is not null )
order by   cast(mn as varchar(2))  + "-" +  cast(da as varchar(2)) + " " + cast(tm as varchar(8))  + "." + ms
asc

print "                 ...  trimmed for brevity ... "  -- puts this line between the log snippets

select top 3 "      [**] " + alert + " [**] " + "  " +  cast(mn as varchar(2))  + "-" +  cast(da as varchar(2))  + " " +
cast(tm as varchar(8)) + "." + ms + " " + + srcip + ":" + srcprt + " -> " + dstip + ":" + dstprt as " "
from alerts_bottom_few
order by   cast(mn as varchar(2))  + "-" +  cast(da as varchar(2))  + " " +  cast(tm as varchar(8))  + "." + ms
asc
```

REFERENCES

Bidwell, Teri. "GCIA Practical Assignment".
URL: http://www.sans.org/y2k/practical/Teri_Bidwell_GCIA.doc

Bowling, Joe. "RE: GCIAC GCIA Version 3.4 Practical Detect #3 - Eric Evans".
intrusions@incidents.org (17 November 2003)
URL: http://cert.uni-stuttgart.de/archive/intrusions/2003/11/msg00126.html (17 November 2003)

Carnegie Mellon University's CERT Coordination Center:
URL: http://www.cert.org/incident_notes/IN-2001-08.html

CERT.ORG. "CA-1997-28"
URL: http://www.cert.org/advisories/CA-1997-28.html

CERT.ORG. "CA-2003-04".
URL: http://www.cert.org/advisories/CA-2003-04.html

Cisco Systems. "How CGMP Leave Processing Functions".
URL: http://www.cisco.com/warp/public/cc/pd/iosw/prodlit/cgmp_an.htm

Cisco Systems. "Multicast Documentation".
URL:
http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123tcr/123tip3r/ip3_m1gt.htm

Cisco Systems. "SAFE SQL Slammer Worm Attack Mitigation".
URL: http://www.cisco.com/warp/public/cc/so/neso/sqso/worm_wp.htm

Coffer, Jason. "MAC Address Lookup and Search".
URL: http://www.coffer.com/mac_find/

COMPNETWORKING. "Port 0".
URL: http://compnetworking.about.com/library/ports/blports_0.htm

Computer Associates. "CID 39147".
URL: http://www3.ca.com/solutions/collateral.asp?CT=27081&CID=39147

Cooperative Association for Internet Data Analysis. "Analysis of the Sapphire Worm".
URL: http://www.caida.org/analysis/security/sapphire/

Delany, Kalen. Inside Microsoft SQL Server 2000. 2001.

Evans, Eric. "LOGS: GCIAC GCIA Version 3.4 Practical Detect #3 - Eric Evans".
intrusions@incidents.org (17 November 2003)
URL: http://cert.uni-stuttgart.de/archive/intrusions/2003/11/msg00123.html (17 November 2003)

GIAC.ORG. "GIAC Certification ADMINISTRIVIA for All Assignments (2.7)".
(November 2003).
URL: http://www.giac.org/administrivia.php (4 December 2003)

GIAC.ORG. "GCIA Practical Assignment (3.4) (Intrusion Detection in Depth)". (October 2003)
URL: http://www.giac.org/GCIA_assignment.php (24 September 2003)

Holland, Jeff and French, Jamie and Tan, Koon Yaw. "GCIA Practical Study and Planning Guide 3.3"
URL: http://www.giac.org/gcia_study_guide_v33.pdf (4 December 2003)

F-Secure: "F-Secure Virus Descriptions : Lovsan"
URL: http://www.f-secure.com/v-descs/msblast.shtml

F-Secure. "MS-SQL Security".

URL: http://www.f-secure.com/v-descs/mssqlm.shtml

FAQ.ORG. “RFC 1179 - Line printer daemon protocol”
URL: http://www.faqs.org/rfcs/rfc1179.html

Goddard Space Flight Center. “Reference Manual”
URL: http://hires.gsfc.nasa.gov/stis/docs/ref_manual/sec1.html
URL: www.gsfc.nasa.gov

Gamble, Steven. “GCIA Practical”.
URL: www.giac.org/practical/GCIA/Steven_Gamble_GCIA.pdf

Gordon, Les. Intrusion Analysis - The Director’s Cut!”, GCIA Practical v3.2 November 22, 2002.
URL: http://www.giac.org/practical/GCIA/Les_Gordon_GCIA.doc

Hayden, Chris. “SQL Slammer Worm”. GCIH practical.
URL: http://www.giac.org/practical/GCIH/Chris_Hayden_GCIH.pdf

Hping. download, manpages
URL: http://www.hping.org/download.html
URL: http://www.hping.org/manpage.html

I-Gateway. “How do I setup a proxy server to work with SubSpace?” (Jan. 4, 2003)
URL: http://games.igateway.net/subspace/proxy.html.

IANA. “Blackhole Server FAQ”
URL: http://www.iana.org/faqs/abuse-faq.htm#FAQonBlackholeServers

Incidents.org Raw Log Readme.
URL: http://www.incidents.org/logs/raw/README (4 December 2003)

Information Sciences Institute. “RFC 793, PROTOCOL SPECIFICATION”. (September 1981)
URL: http://www.ibiblio.org/pub/docs/rfc/rfc793.txt

Insecure.Org. “Nmap Fingerprinting”.
URL: http://www.insecure.org/nmap/nmap-fingerprinting-article.html

Kesavamatham, Sai Prasad. GIAC GCIA Practical (version 3.3). (7 July 2003)
URL: www.giac.org/practical/GCIA/ SaiPrasad_Kesavamatham_GCIA.pdf

Larrat, Glenn. “GCIA Practical Assignment”
URL: http://is.rice.edu/~glratt/practical/Glenn_Larratt_GCIA.html

Litchfield, David. “Advanced Windows Shellcode”

URL: http://www.security-corporation.com/exploits-20030822-000.html
URL: http://www.nextgenss.com/advisories/mssql-udp.txt

Maslowski-Yerges, Al. "GIAC Certified Intrusion Analyst (GCIA) Practical Assignment Version 3.3"
URL: http://www.giac.org/practical/GCIA/Al_Maslowski-Yerges_GCIA.pdf

Massachusetts Institute of Technology. "LPRng How-To". RFC 1179 - Line Printer Daemon Protocol
URL: http://web.mit.edu/source/third/lprng/doc/LPRng-HOWTO-18.html

McAffee. W32/Mimail.i@MM
URL:
http://us.mcafee.com/virusInfo/default.asp?id=helpCenter&hcName=mimail&cid=8954

Microsoft. "Code Response".
URL:
http://www.microsoft.com/sql/techinfo/administration/2000/security/coderesponse.asp

Microsoft. "MS02-039".
URL:
http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS02-039.asp

Microsoft. "MS02-043".
URL: http://www.microsoft.com/technet/treeview/?url=/technet/security/bulletin/MS02-043.asp

Microsoft. "MSDE Applications".
URL: http://www.microsoft.com/technet/treeview/?url=/technet/security/MSDEapps.asp

Microsoft. "Security Market Bulletin".
URL:
http://www.microsoft.com/sql/techinfo/administration/2000/security/SecurityMarketBulletin.doc

Microsoft. "Slammer".
URL: http://www.microsoft.com/sql/techinfo/administration/2000/security/slammer.asp

Microsoft. "Slammer Bulletin".
URL:
http://www.microsoft.com/sql/techinfo/administration/2000/security/slammerbulletin.asp

Microsoft. "Slammer Virus Announcement".
URL: http://www.microsoft.com/presspass/press/2003/jan03/01-25virus.asp

Microsoft. “Slammer Webcast”.
URL:
http://www.microsoft.com/sql/techinfo/administration/2000/security/slammerwebcast.asp

Mitre. “CVE-1999-016”.
URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0016

Mitre. “CAN-1999-0061”.
URL: (http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0061)

Mitre. “CVE-2000.0917”.
URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2000-0917

Mitre. “CVE-2001-0154”.
URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0154

Mitre. “CAN-2002-0649:.
URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0649

Mitre. “CVE-2001-0500”
URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0500

Mitre. “CVE-CAN-2002-0650”
URL: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0650

Morris, Gary. Contemporary Intrusion Detection and Analysis. GCIA Practical (version 3.3)Submitted: October 17, 2002
URL: http://www.giac.org/practical/GCIA/Gary_Morris_GCIA.doc

Network Associates Technology's Virus Information Library:
URL: http://vil.nai.com/vil/content/v_99142.htm

Network Associates:
URL: http://vil.nai.com/vil/content/v_99992.htm

NetworkSorcery.Com. “IGMP, Internet Group Management Protocol”
URL: http://www.networksorcery.com/enp/protocol/igmp.htm
NGSSoftware. “Windows .NET Server (RC1) and MSDE”.
URL: www.nextgenss.com/advisories/dotnet-msde.txt

Novell. “iFolder Features”.
URL: https://128.172.22.25:51443/iFolder/

Novell. “Netware 5 Features”.
URL: http://www.novell.com/coolsolutions/netware/features/a_ports_nw5_nw.html

Packet Storm Security. "Microsoft SQL Slammer Worm Propagation".
URL: http://packetstormsecurity.nl/advisories/iss/iss.slammer.worm.txt

Pilker, Joanne. "MS SQL Slammer/Sapphire Worm". GSEC v1.4b ()
http://www.giac.org/practical/GSEC/Joanne_Pilker_GSEC.pdf (4 December 2003)

Roesch, Martin and Green, Chris. "Writing Snort Rules". Snort Users Manual. 2003.
URL: http://www.snort.org/docs/writing_rules/chap2.html#tth_chAp2 (17 November 2003).

Security Focus. "BID 5310"
URL: http://www.securityfocus.com/bid/5310

Security Focus. "BID 5311"
URL: http://www.securityfocus.com/bid/5311

Security Corporation. "Exploits".
URL: http://www.security-corporation.com/exploits-20030822-000.html

Snort.Org. "About Snort"
URL: www.snort.org/about.html

Snort.Org. "SID 524"
URL: http://www.snort.org/snort-db/sid.html?sid=524

Snort. "SID 527".
URL: http://www.snort.org/snort-db/sid.html?sid=527

Snort. "SID 533".
URL: http://www.snort.org/snort-db/sid.html?sid=533

Snort SID 1322
URL: http://www.snort.org/snort-db/sid.html?sid=1322

Snort.Org. "SID 2003"
URL: http://www.snort.org/snort-db/sid.html?sid=2003

Snort.Org. "SID 2004"
http://www.snort.org/snort-db/sid.html?sid=2004

So, Hee, GCIA Practical Assignment v3.0, Feb 16, 2002
URL: http://www.giac.org/practical/Hee_So_GCIA.doc

Sorensen, Stan. "SQL 2000 Security"

URL:
http://www.microsoft.com/sql/techinfo/administration/2000/security/StanSorensen2_06.ppt

SQLSecurity.com
URL: http://www.sqlsecurity.com/DesktopDefault.aspx

Thompson, Jason, "LOGS: GIAC GCIA Version 3.3 Practical Detect - Repost". (17 July 2003)
URL: http://cert.uni-stuttgart.de/archive/intrusions/2003/07/msg00183.html (4 December 2003)

Vyas Kondreddi, Narayana. SQL Server Best Practices
URL: http://vyaskn.tripod.com/sql_server_security_best_practices.htm

Wesemann, Daniel. "Practical Detect #1 -- Suspicious IGMP Packets". Practical Version 3.3 (24 March 2003)
URL: www.giac.org/practical/GCIA/Daniel_Wesemann_GCIA.pdf (17 November 2003)

Whitehats.com. "IDS 456".
URL: (http://www.whitehats.com/info/IDS456)

Whitehats.com. "IDS 457".
URL: (http://www.whitehats.com/info/IDS457)