



Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permitted without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Network Monitoring and Threat Detection In-Depth (Security 503)"
at <http://www.giac.org/registration/gcia>

Intrusion Detection: In-Depth Analysis

© SANS Institute 2003, Author retains full rights.

Johnny Calhoun
GIAC GCIA Practical version 3.3
Submitted: January 8, 2003

Assignment #1: Describe the State of Intrusion Detection.....	4
Revealing the Insecurities of Wireless Networks	4
Introduction	4
Background	5
Identifying the Problem	5
Wireless Intrusion Detection.....	5
Countermeasures	6
References for this Assignment.....	7
Assignment #2: Three Network Detects.....	8
Detect #1: Banner Grabbing	8
Source of Trace	8
Detect was Generated by	8
Probability the Source Address was Spoofed	9
Description of the Attack	9
Attacking Mechanism	9
Correlations	10
Evidence of Active Targeting	11
Severity	11
Defensive Recommendation	11
Multiple Choice Question	12
Detect #2 Apache Chunked Encoding Worm	13
Source of Trace	13
Detect was Generated by	13
Probability the Source Address was Spoofed	14
Description of Attack.....	14
Attack Mechanism.....	15
Correlations	16
Evidence of Active Targeting	17
Severity	17
Defensive Recommendation	18
Multiple Choice Question	18
Detect #3 The XMAS Scan.....	19
Source of Trace	19
Detect was Generated By.....	19
Probability Source Address was Spoofed	19
Description of Attack.....	19
Attack Mechanism.....	19
Correlations	20
Evidence of Active Targeting	21
Severity	21
Defensive Recommendation	21
Multiple Choice Test Question	21
References for this Assignment.....	22
Assignment #3 Analyze This	23
Executive Summary	23

List of Analyzed Files.....	23
Prioritized Detects	24
Incomplete Packet Fragments Discarded.....	25
spp_http_decode: IIS Unicode attack detected	26
SMB Name Wildcard	27
SUNRPC highport access!.....	27
TFTP – External UDP Connection to Internal TFTP Server.....	28
Registration Information for 24.90.124.187	29
Registration Information for 212.113.174.194.....	29
CGI Null Byte Attack Detected.....	31
Relational Analysis Process	31
Top Talkers	33
Registration information for 213.115.19.12:	34
Registration Information for 130.161.220.212:	36
Out of Spec.....	36
Insights about Internal Machines	37
Registration Information for 128.121.97.106:	37
Defensive Recommendations.....	39
References for this Assignment.....	40

© SANS Institute 2003, Author retains full rights.

Assignment #1: Describe the State of Intrusion Detection

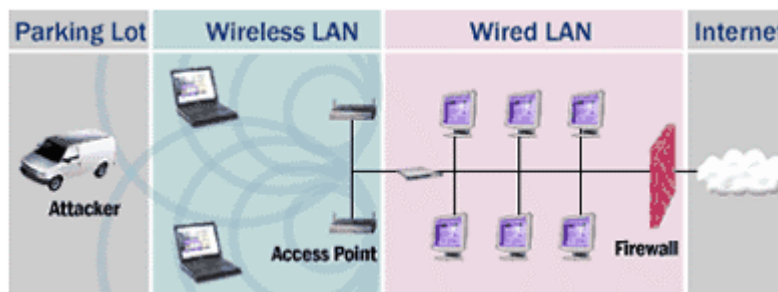
Revealing the Insecurities of Wireless Networks

Introduction

With the ever-increasing popularity of wireless technologies, new threats have emerged in the information security arena. Wireless Local Area Networks, also known as WLANs, are beginning to become more and more widespread. This can be attributed to ease of installation and maintenance as compared to a wired LAN, no more long cable runs or cutting holes in walls. This ease comes at a price; WLANs pose a serious security risk. Carelessly configured access points and wireless routers can become a “welcome-all” gateway into a wired network, and for this reason has become the latest hacker playground. A misconfigured WLAN can easily become a springboard for a larger attack and that is what I intend to show.

Hackers can determine where WLANs are physically located and how they are configured via a technique known as “wardriving.” Wardriving consists of driving around in an automobile while using a laptop equipped with a wireless card to detect any wireless access points in the surrounding area. Depending on the type of wardriving tool used, the two most common, Kismet for Linux users or Netstumbler for Windows users, it is possible to gather information about your network from afar. The very aspect of a “wireless” network makes it a danger from a security standpoint; this is due to the fact that no physical access is needed for an intruder to access the network, thus making the job of sniffing packets or capturing sensitive data much easier for an intruder. Normally, with a wired network access to layer 1 is protected by locked doors, and limited by a physical cable, but with wireless networks there is no such protection.

The location of the access point can also escalate the risk involved, if the access point is located behind the firewall, then the very presence of a firewall becomes non-existent, thus giving an attacker instant access via wireless connection to the internal network, which is the last place we want an unauthorized user. Consider the following diagram:



In the above diagram, from www.airdefense.net, we see the danger that improperly placed access points can have on a network. The access points are connected directly to the internal network and may possibly be granting unauthorized access to an intruder positioned outside in the parking lot.

Background

Access Points are not the only threat, peer-to-peer WLANs known as “ad hoc” networks and rogue access points can also affect the overall security of the network. Ad hoc networks are composed only of WLAN cards and do not require any wireless access point or authentication scheme in order to establish a connection. Rogue access points are access points that are put into use without authorization. These are not the only problems faced when deploying a wireless LAN, many access points come with a preset default configuration that is insecure, and so configurations must also be considered potentially harmful. Most are set up without WEP (Wired Equivalent Privacy) enabled, which means there is no encryption. Default passwords, default SSIDs and allowing open broadcast SSIDs are also common insecurities. The default configuration alone can allow any user with a wireless card to access the wireless network without any authentication, and also grants them the ability to sniff network traffic using tools like tcpdump and ethereal. At times the only authentication in place is MAC address based filters, which is weak because MAC addresses can be spoofed easily. This can lead to identity theft and connection hijacking where an attacker takes control over a pre-established connection without authentication, and this can be done even if security measures such as IPSEC and WEP are already in place.

Identifying the Problem

The obvious problem is that an intruder, via an incorrectly configured wireless access point, can easily gain instant remote access to a network. Access to a wireless network can be abused in many ways. One such abuse is unauthorized access to a private Internet connection. An attacker could also use the insecure WLAN as a springboard to launch other attacks. There are several tools that can aid an attacker in gathering information and possibly compromise hosts within that network. There are tools available that can enable a wireless card to act as mobile access point, which can lead to a malicious user posing a valid access point by spoofing the MAC address of the real access point and also sending disassociations to the access points.

Wireless Intrusion Detection

The current approach to IDS in wireless LANs is two tiered – looking for wireless

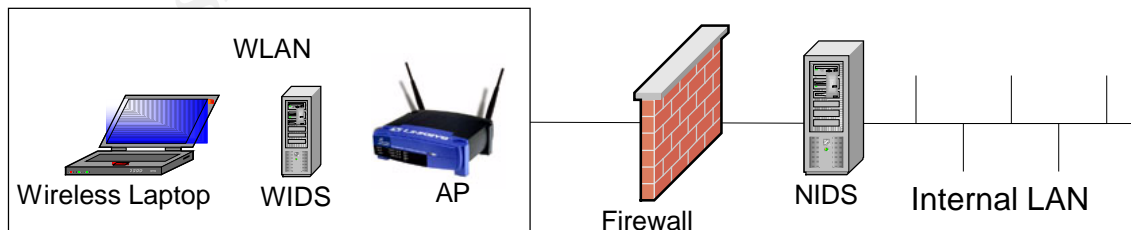
attacks and looking for IP based attacks. The wireless IDS focuses primarily on wireless attacks and does not perform IP-based intrusion detection. If we want to watch for IP-based attacks, then we simply put a NIDS at the wireless AP choke point. That will take care of most attacks, the ones your IDS has signatures for, but does not protect against wireless attacks. The NIDS cannot detect wireless attacks, so a wireless NIDS implementation is therefore needed. One such implementation is by Airdefense (www.airdefense.net).

A wireless network will require both IDS technologies to provide proper visibility and coverage. The wired NIDS cannot detect any wireless based attacks or wireless threats including: rogue access points, soft access points, ad hoc networks, sniffers, netstumbler probes or kismet users to name a few. Basically, a wired NIDS is useless against wireless attacks, but can detect wireless born IP based attacks once it hits the wire.

The wireless IDS can detect the above mentioned attacks as well as provide minimal Intrusion Prevention, such as trapping a signal, and forcing a disassociation. The Airdefense solution also provides for health monitoring of wireless devices as an added benefit.

Countermeasures

There are various ways of defeating wireless security measures, so a layered approach must be taken in order to properly secure a WLAN. WLANs create an interesting problem in that as security professionals we have to deal with the normal security threats in addition to all the new threats that wireless technologies bring to the table. We must remain vigilant on both fronts and monitor each with precision and accuracy. We must also aggregate events from both wired NIDS and wireless NIDS and correlate them to ensure maximum visibility, and with both in place we can begin to build a strong security infrastructure. I have designed a diagram below of what a secure wireless infrastructure might look like, and if properly implemented can become a secure foundation for a wireless network environment.



There are also other measures that can be put into place to further increase the security of the wireless network. A technique known as RF signal shaping can be used to “directionalize” the RF signals emitted from the access point.

One way this can be accomplished is to use directional antennas. Also, because signals can bleed over beyond your perimeter, consider reducing the power of the access point, as to weaken the signal so that it spans a smaller distance. Physical security is sometimes just as important.

Although WEP can be broken using tools such as aircrack-ng and WEPCrack, it is still a good idea to have it in place. The reason for this is because it takes a lot of packets to be captured before the crack can be performed, an attacker is more likely to move on to the next target if he sees a WEP enabled network. This is similar to a burglar moving on to the next house where the owners don't lock the doors.

Another commonplace security measure is to place a VPN over the wireless link and to use strong mutual authentication. While the management and control frames can still be seen using this solution, if WEP is broken, the IP contents will not be readable due to encryption. This creates one more level of security and causes an intruder a lot more work in order to compromise the network. With the widespread use of wireless these days, an attacker is prone to ignore your network if you have WEP, and IPSEC enabled, and is likely to move on to the next network down the street.

References for this Assignment

Airdefense Wireless Security

www.airdefense.net

Linux Security Article

www.linuxsecurity.com/features/stories/wireless-kismet.html

Kismet Wardriving Tool

www.kismetwireless.net

Aircrack-ng WEP cracking Tool

aircrack-ng.org

Netstumbler Wardriving Tool

www.netstumbler.com

Google Search Engine

www.google.com

HostAP

hostap.epitest.fi

Assignment #2: Three Network Detects

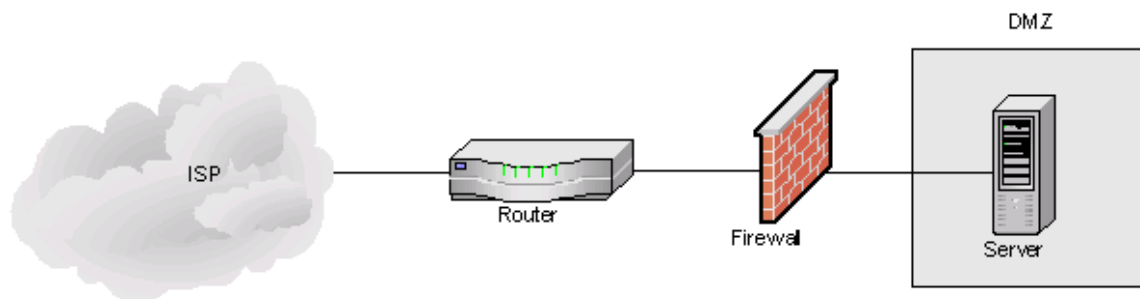
For this assignment I have analyzed three detects. The first detect was one I discovered on the incidents mailing list and responded to, it is a simple detect, but is still worthy of an in-depth analysis. The second detect was captured in real-time during an actual attack on a web server. The third detect was taken from the incidents.org raw log files.

Detect #1: Banner Grabbing

Most attacks and exploits are platform dependent, meaning they only work on a select group of platforms and servers. Attackers can find out information about a server in plain view, such as visiting the website of a victim and looking for slogans such as, "powered by Redhat Linux" or by sending a simple crafted "GET" request to a web server on port 80. The technique of sending crafted packets to a served port in an effort to extract information is known as "Banner Grabbing." Telnetting to a web server on port 80 and submitting "GET x HTTP/1.0" can reveal lots of useful information to an attacker, such as the operating system, type of web server and version numbers. Some worms, such as the sadmind worm, take advantage of this feature when identifying vulnerable hosts. Below is such a detect generated by snort:

Source of Trace

The following trace was taken from the incidents mailing list. An administrator reported seeing strange log entries and was curious to what they were and if anyone else had seen the same type of activity. Although the layout of the network is unknown, we can safely presume that this machine is a web server in the DMZ.



Detect was Generated by

This detect looks to be generated by an Apache web server due to the similarities in the logging format:

```
213.165.144.xxx - - [12/Oct/2002:05:40:01 -0500] "GET /sumthin HTTP/1.0" 404 1086 "-" "-"
213.165.144.xxx - - [12/Oct/2002:05:40:01 -0500] "GET /sumthin HTTP/1.0" 404 1086 "-" "-"
213.165.144.xxx - - [12/Oct/2002:05:40:01 -0500] "GET /sumthin HTTP/1.0" 404 1086 "-" "-"
213.165.144.xxx - - [12/Oct/2002:05:40:01 -0500] "GET /sumthin HTTP/1.0" 404 1086 "-" "-"
213.165.144.xxx - - [12/Oct/2002:05:40:01 -0500] "GET /sumthin HTTP/1.0" 404 1086 "-" "
```

The “GET” request was logged due to the fact that an error was returned. In this particular case a 404 error was returned, which is generally a sign that a page cannot be found, but it also returns headers with valuable information back to the requestor. The format of the log is as follows:

<Source Address> <Date><Time> <Details>

Probability the Source Address was Spoofed

It is highly unlikely that the source address was spoofed. This is because the attack is soliciting some type of response from the victim host, and also because HTTP communicates over TCP, which requires that the three-way handshake be completed in order for a connection to be established and communications to take place. So if this address were spoofed, the results would be returned to the spoofed host and not the original sender. It is possible to hijack TCP sessions, but unlikely in this instance because the level of sophistication involved is out of context with this simple attack. The nature of this attack does not indicate spoofing activity.

Description of the Attack

This is an information gathering attempt against a web server. The technique used is known as banner grabbing and the purpose of this attack is to gain information about the server that may be used in a future attack. While this is a very quiet probe, it is still important that attention is paid to it, because it can be a tip-off to an up and coming larger and more threatening attack.

Attacking Mechanism

In this attack an attacker is simply trying to gain information about a web server, such as the type of web server (Apache, IIS. Etc.) and Operating System type. The attack is very simple and is launched by merely formatting a URL such that an error is spawned, or to just telnet the web server on port 80 (HTTP) and craft a GET request that will return HTTP header information.

As a case study of how this attack works, let’s take a look at an excerpt from a recent post to the incidents mailing list that I found and responded to that deals with Banner Grabbing.

Upon analyzing this post I recalled several similar events that I have seen previously that used banner grabbing as a pre-attack scan before launching some sort of attack. Being the nice guy that I am, I figured I should post to the

list describing what was occurring. The full post can be found here:
<http://online.securityfocus.com/archive/75/295887/2002-10-18/2002-10-24/2>

Post follows:

"This looks to be a banner grabbing attempt on your web servers. A lot of scanners/worms will do this in an attempt to find out what type of web server you are running and compare it against a list of vulnerable servers for some particular exploit. The /sumthin is placed within the GET command to trigger a 404 error, which in turn reveals valuable information about your server back the requestor. If the information returned by your server is useful to the scanner/worm you may see other exploits in the near future targeted towards your box. For a more practical example, consider the sadmind worm which issues the following request for this purpose: "GET x HTTP/1.0." If you want to see what is returned by your box, simply telnet to your server on port 80 and issue the same request and hit ENTER twice. You should see something similar to:

```
[root@webserver root]# telnet 127.0.0.1 80
Trying 127.0.0.1 ...
Connected to 127.0.0.1.
Escape character is '^]'.
GET /sumthin HTTP/1.0
```

```
HTTP/1.1 404 Not Found
Date: Thu, 17 Oct 2002 11:21:35 GMT
Server: Apache/2.0.40 (Unix)
Content-Length: 286
Connection: close
Content-Type: text/html; charset=iso-8859-1
```

. . . and probably some 404 HTML error code too

Notice how it revealed the Web Server type, Version and OS it runs on.

I would consider this type of activity as an information gathering attempt . . .

So as you can see, there is a lot of information to be gathered easily from a web server by simply crafting a GET request.

Correlations

As stated in the quote above, a correlation was made between the detect used for this analysis and a previous snort alert I have dealt with that was triggered by the sadmind worm. As a comparison let's now take a look at a sample Snort alert that triggers on the sadmind worm:

Sadmin Worm Probe - SNORT IDS

[**] [1:1375:5] WEB-MISC sadmin worm access [**]

[Classification: Attempted Information Leak] [Priority: 2]

10/03-22:32:07.164902 x.95.120.252:59768 -> x.18.0.10:80

TCP TTL:236 TOS:0x0 ID:40385 IpLen:20 DgmLen:58 DF

AP Seq: 0xCECED6A7 Ack: 0x97721E37 Win: 0x25BC TcpLen: 20

[Xref => <http://www.cert.org/advisories/CA-2001-11.html>]

GET x HTTP/1.0

This alert is very similar to the detect being analyzed in several ways. First of all, both destination or victim machines are web servers and the targeted port is 80. Also, the GET requests essentially perform the same function, they trigger an error that returns valuable information.

Evidence of Active Targeting

There is not enough evidence to support the idea of host based active targeting. A scan for an open port 80 was probably launched across a large number of Internet addresses prior to this detect being triggered. Then after the scan returned the results, the scope of targeting was narrowed.

Severity

We can calculate severity using the following formula:

$$(\textit{Target's Criticality} + \textit{Lethality of Attack}) - (\textit{System Defense} + \textit{Network Defense})$$

Criticality	4	This was directed towards a web server.
Lethality	1	Information Gathering in nature, not an exploit
System Defense	4	OS assumed up to date with relevant patches
Network Defense	1	This is web traffic firewalls and routers do not block

Now that we have assigned values to the four aspects of severity, let us now calculate the severity of this attack: $\textit{Severity} = (4 + 1) - (4 + 1) = 0$

Due to the lack of lethality of this attack, the severity is low. This is more of an information gathering attempt than an actual exploit driven attack.

Defensive Recommendation

The best way to combat an attack that gathers valuable information about your host is to turn off the mechanism that reveals such information. In the particular case we can adjust our web server so that its footprint is not displayed. On an Apache web server the way to stop this is to add the following line to the apache.conf file:

ServerSignature Off

By adding the above to your configuration, less information is freely given to an attacker.

Multiple Choice Question

```
213.165.144.xxx - - [12/Oct/2002:05:40:01 -0500] "GET /sumthin HTTP/1.0" 404 1086 "-" "-"
213.165.144.xxx - - [12/Oct/2002:05:40:01 -0500] "GET /sumthin HTTP/1.0" 404 1086 "-" "-"
213.165.144.xxx - - [12/Oct/2002:05:40:01 -0500] "GET /sumthin HTTP/1.0" 404 1086 "-" "-"
213.165.144.xxx - - [12/Oct/2002:05:40:01 -0500] "GET /sumthin HTTP/1.0" 404 1086 "-" "-"
213.165.144.xxx - - [12/Oct/2002:05:40:01 -0500] "GET /sumthin HTTP/1.0" 404 1086 "-" "
```

Based on the above detect, what is the “404” indicative of:

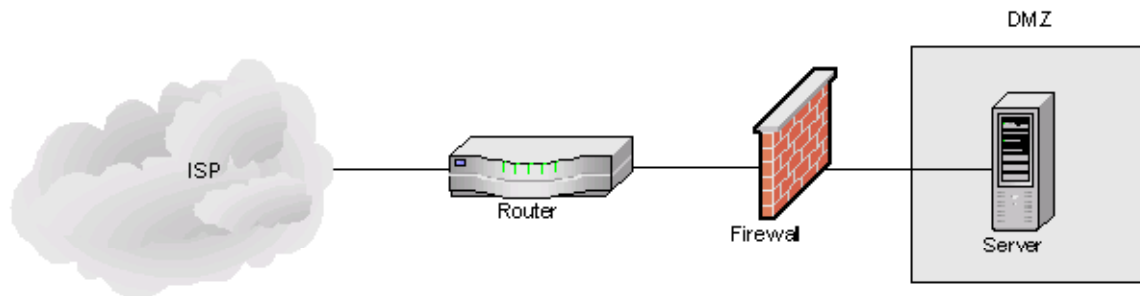
- a) 404 connection attempts
- b) type 404 connection
- c) HTTP error code
- d) IP ID number

The correct answer is c.

Error code 404 is used to signify an error in processing an HTTP request, in this particular case, to report that a page was not found. Code 404 is not the only error code there are many others, and they can be signs that someone is doing something bad on your web server.

© SANS Institute 2003, Author retains full rights.

Source of Trace



Port Intrusion Detection

This detect was generated by the Snort Intrusion Detection System:

[illegible]

As we begin our analysis of this alert we see that the attacker has found a web server and has attempted to compromise this host via a known vulnerability within the Apache implementation. The format of a snort alert is as follows, and

describes the connection in detail:

```
Timestamp      Source:Port -> Destination:Port
<Protocol> <TimeToLive> <TypeOfService> <IP ID> <IPHeaderLength> <PacketLength> <[options]>
<flags> <SequenceNumber> <AckValue> <WindowSize> <TcpHeaderLength>
<TCP Options>
```

Below is the Snort signature that alerted on this event:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-MISC Apache
Chunked-Encoding worm attempt"; flow:to_server,established; content:"CCCCCCC\
AAAAAAAAAAAAAAAAAAAA"; nocase; classtype:web-application-attack; reference:bugtraq,4474;
reference:cve,CAN-2002-0079;reference:bugtraq,5033; reference:cve,CAN-2002-0392; sid:1809; rev:2;)
```

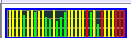
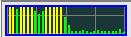
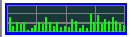
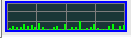
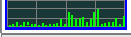
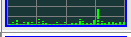

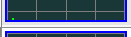
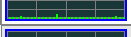
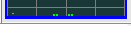
This signature alerts on *established* (Ack or more flags - A+) ingress traffic destined for a web server on a web port, in this case port 80, with the content of: "CCCCCCC:AAAAAAAAAAAAAAAAAAAAAA."

Probability the Source Address was Spoofed

It is unlikely that the source was spoofed in this attack, due to the fact that a response is expected. HTTP connections are established via the TCP three-way handshake, therefore making it extremely difficult for an attacker to pull off a successful spoofing attack. Also, by examining proof-of-concept code^{*}, there does not seem to be any mechanism for spoofing the source.

Description of Attack

The foundation of all websites is the web server. Each day web servers all over the world are constantly scanned for vulnerable services, applications, backdoors, and not to mention the lethal worms that probe massive numbers of hosts looking for various types of insecurities and try to exploit them. It is for this reason much time must be spent in aggregating and correlating web server logs with NIDS logs. See the chart below for relational comparisons of port 80 scans compared to other massive scans that sweep across cyberspace.

Service Name	Port Number	Activity Past Month	Explanation
http	80		HTTP Web server
nothing-no	137		
ms sql s	1433		Microsoft SQL Server
netbios-ssn	139		Windows File Sharing Probe
ftp	21		FTP servers typically run on this port
smtp	25		Mail server listens on this port.
sunrpc	111		RPC, vulnerable on many Linux systems. Can get root
???	3969		
https	443	
???.....	4885		eDonkey P2P software

^{*} <http://online.securityfocus.com/bid/5033/exploit/>

As you can see in the above chart, from dshield.org, port 80 is the most common port scanned on the Internet today. Although the graph only represents one day of scans, repeated visits to dshield.org reveals that port 80 scans are the most prevalent over a longer time span, though at times the list is topped by other ports that correspond to the latest vulnerability. Many worms target web servers, including CodeRed, Nimda, sadmind, and the Apache Chunked Encoding worm.

The following excerpt is from http://www.iss.net/security_center/static/9249.php and describes the vulnerability in greater detail:

“Apache HTTP Server versions 1.2.2 and later, 1.3 up to and including 1.3.24, and 2.0 up to and including 2.0.36 are vulnerable to a heap buffer overflow in the mechanism that calculates the size of “chunked” encoding. Chunked encoding is a process by which a client generates a variable sized “chunk” of data and notifies the Web server of the data’s size before transferring it, so that the Web server can allocate a buffer of the correct size. The Apache HTTP Server has a software flaw that misinterprets the size of incoming data chunks. A remote attacker can use this vulnerability to overflow a buffer and execute arbitrary code or cause a denial of service against the affected Web server.”

The following, taken from <http://online.securityfocus.com/bid/5033/discussion>, gives us a few more details:

“When processing requests coded with the ‘Chunked Encoding’ mechanism, Apache fails to properly calculate required buffer sizes. This is believed to be due to improper (signed) interpretation of an unsigned integer value. Consequently, several conditions may occur that have security implications. It has been reported that a buffer overrun and signal race condition occur. Exploitation of these conditions may result in the execution of arbitrary code.”

This attack comes in the form of a worm that scans the Internet for vulnerable web servers, then it proceeds to exploit the server via the chunked encoding vulnerability. If the exploit attempt is successful, then the worm will upload a copy of itself in the form of a uuencoded file named “.uua” to the victim. With the new file in place, located in the /tmp directory, the file is uudecoded into a binary file called “.a.” The new file can then be executed to begin searching for more hosts to infect.

Attack Mechanism

The lifecycle of this attack is: scan, exploit, transfer and listen. The worm begins by scanning the Internet for web servers. This is accomplished by sending an legit HTTP request to the web server, and observing the header that comes back. If the header matches the list of vulnerable apache implementations, notice implementation because this attack is only known affect certain operating systems, FreeBSD in particular. When vulnerable servers are found they are stored for later use. Next, it attempts to exploit the web servers that were found

by the scan via the chunked encoding vulnerability. Then, the worm transfers a uuencoded copy of itself (.uua) to the /tmp directory infecting of the victim. Lastly, it listens on UDP port 2100 for control packets to ok the launch of the new worm. A graphical diagram of this activity can be found at <http://www.idefense.com/Intell/CI063002.html>.

Correlations

The following alert was triggered by Snort in the midst of a broad scan that was focusing on Apache web servers.

```
[**] [1:1807:1] WEB-MISC Transfer-Encoding: chunked [**]
[Classification: Web Application Attack] [Priority: 1]
09/19-17:29:36.001811 x.129.81.40:3838 -> x.244.39.70:80
TCP TTL:49 TOS:0x0 ID:34088 IpLen:20 DgmLen:510 DF
***AP*** Seq: 0xA84BA748 Ack: 0x9608144F Win: 0x8218 TcpLen: 32
TCP Options (3) => NOP NOP TS: 1353816331 2264210427
[Xref => http://www.securityfocus.com/bid/4474]
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0079]
[Xref => http://www.securityfocus.com/bid/5033]
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0392]

.....X-AAAA
: .....
X-AAAA: .....
.....X-AAAA: .....
.....X-AAAA: .....
.....X-AAAA: .....
.....Transfer-Encoding: chunked....5..BBBBB..
fffff6e..
```

Below are logs that were taken from a router that alerted on scans across multiple hosts on port 80 from the same host as the snort alert:

```
Sep 19 20:17:43 xxx.117.106.65 2096595: %SEC-6-IPACCESSLOGP: list 101 denied tcp
61.129.81.40(3791) -> xxx.244.39.75(80), 1 packet
Sep 19 20:17:43 xxx.117.106.65 2096596: %SEC-6-IPACCESSLOGP: list 101 denied tcp
61.129.81.40(3792) -> xxx.244.39.76(80), 1 packet
Sep 19 20:17:43 xxx.117.106.65 2096597: %SEC-6-IPACCESSLOGP: list 101 denied tcp
61.129.81.40(3793) -> xxx.244.39.77(80), 1 packet
Sep 19 20:17:43 xxx.117.106.65 2096598: %SEC-6-IPACCESSLOGP: list 101 denied tcp
61.129.81.40(3794) -> xxx.244.39.78(80), 1 packet
Sep 19 20:17:43 xxx.117.106.65 2096599: %SEC-6-IPACCESSLOGP: list 101 denied tcp
61.129.81.40(3795) -> xxx.244.39.79(80), 1 packet
```

In the above log entries, which correlates with the Snort IDS detect, we can determine several things about this scan by analyzing it as a whole and then breaking it down piece by piece. As we look at the whole trace we see that the

attacker is scanning the Internet for victim hosts, which means that this is probably the initial recon phase of an attack. I say “probably” because the probes are getting blocked by the screening router *list 101 denied tcp*, because of this we only know that there was an attempted connection, we do not know what the *intent* of the connection was. The destination ports are all the same, which tells me that this attacker is scanning for hosts that accept connections on port 80, in other words a web server. I also see that the destination IP addresses are incrementing with each scan, as well as incrementing source port numbers, which is indicative of host scanning activity. By correlating these logs and alerts we can begin to see how the attack was structured. First a large number of hosts were scanned for open port 80. Then, once it was determined which of these hosts were web servers, the attack was launched. It should also be noted that since this is an attack that only affects apache web servers, more reconnaissance probably took place before the attack to determine which type of web server was running on this host. This information could have easily been determined by using the banner grabbing technique discussed in detect #1.

Evidence of Active Targeting

This scenario does not display evidence of active targeting. As stated before, a broad range of IP addresses were scanned prior to this attack being launched, so on the second go round this was just a host that happened to fall within a predetermined criteria for exploitation, in other words it was a potentially vulnerable web server. Also because this exploit is primarily worm driven, there is no human element involved to focus this attack, but I am sure that this could be performed with altered or derivative code.

Severity

We can calculate severity using the following formula:

$$(\text{Target's Criticality} + \text{Lethality of Attack}) - (\text{System Defense} + \text{Network Defense})$$

Criticality	4	This was an attack on a web server. While a successful attack against a web server can be embarrassing, it does not get a 5 because it does not affect the network as bad as a firewall or DNS hack.
Lethality	4	It may be possible to execute arbitrary code or DOS, thereby altering the contents of the webpage or bringing the box down completely
System Defense	4	Operating System and patches were up to date, this platform was not vulnerable to this attack
Network Defense	1	Due to nature of attack, and the server being in the DMZ, firewalls and routers do not help much because web traffic is allowed to pass.

Now that we have assigned values to the four aspects of severity, let us now

calculate the severity of this attack: $Severity = (4 + 4) - (5 + 1) = 2$

Defensive Recommendation

The best counter measure for this attack is to patch any vulnerable systems referred to in the advisories. Available patches should be applied as soon as possible. Also due to the fact that headers are used in this attack, it would be wise to disable this mechanism. Which should be an easy change to the apache configuration file, and was noted in the previous detect. If the host is already infected, to eradicate the worm simply remove the .uua and .a files from the tmp directory and kill the running worm process.

Multiple Choice Question

Which port does the Apache Chunked Encoding Worm Listen for Control Packets?

- a) UDP 2100
- b) TCP 2100
- c) UDP 2001
- d) TCP 2001

The correct answer is a. A newly infected machine will send out a single UDP packet to its infector, and will listen on UDP port 2100 for two UDP "control" packets before it begins to scan for other hosts.

© SANS Institute 2003, Author retains full rights.

Detect #3 The XMAS Scan

Source of Trace

The following detect was taken from the raw logs located at <http://www.incidents.org/logs/Raw/>. The logs are in a tcpdump binary format. The particular log that the detect originated from was 2002.8.23. The logs below were taken from the Snort alert file that was produced by the 2002.8.23 raw log through snort:

```
[**] [1:1228:1] SCAN nmap XMAS [**]  
[Classification: Attempted Information Leak] [Priority: 2]  
09/23-14:38:46.316507 xxx.74.249.65:61621 -> xxx.61.16.19:601  
TCP TTL:50 TOS:0x0 ID:55961 IpLen:20 DgmLen:60  
**U*P**F Seq: 0x417A1598 Ack: 0x0 Win: 0x800 TcpLen: 40 UrgPtr: 0x0  
TCP Options (5) => WS: 10 NOP MSS: 265 TS: 1061109567 0 EOL  
[Xref => arachnids 30]
```

Detect was Generated By

The detect was generated by the Snort Intrusion Detection System version 1.9.0 build 209. The alerts were produced by using Snort to read the binary logs and write the alerts to a specified log directory using the FULL alert format. The following command was issued, *snort -c /root/rules/snort.conf -r 2002.8.23 -l log -N -A full*, and the alert file was generated.

Probability Source Address was Spoofed

The packets contained in this detect could easily be spoofed because there is no initial connection or TCP three way handshake involved. But, because the attack is information gathering in nature, and a response is needed it is unlikely that the source address has been spoofed.

Description of Attack

This is an information gathering attack that probes a victim host in an effort to solicit a response that can identify various details about that host. This is often a pre-attack warning that someone is targeting a specific type of host. One such detail includes the type of operating system running on the host, which is most often the case. The reason for identifying the operating system from the attackers point of view is to find out if that particular host is vulnerable to a specific attack. For example, an IIS exploit does not work on an Apache machine, and a Unix attack doesn't work on a Windows host, so identifying the OS of the target is a critical step in compromising a host.

Attack Mechanism

The attacking mechanism in this detect is to set the PUSH, FIN, and URG flags in the TCP header in an attempt to gather a response from the victim host. Now

this is harmless to the host, but it does help to discriminate between the different OS TCP stacks. A Solaris TCP stack may respond in one manner, while an OpenBSD host will respond completely different or may not respond at all. This technique is known as OS fingerprinting and the tool most commonly used is nmap*, the Network Mapper. Often an XMAS scan, characterized by the UPF flags being set will often be accompanied by other nmap scans such as a TCP scan, that attempts connections across multiple ports to determine the state of that port, whether it be open, closed, or filtered. If we take a look at a tcpdump capture we can see the three flags set that indicate this is an XMAS tree scan:

```
[root@laptop log]# tcpdump -Xn -r 2002.8.23 host xxx.74.249.65 and dst port 601
14:38:46.316507 xxx.74.249.65.61621 > xxx.61.16.19.601: FP 1098519960:1098519960(0) win 2048 urg
0 <wscale 10,nop,mss 265,timestamp 1061109567 0,eol>
0x0000 4500 003c da99 0000 3206 fad8 734a f941 E.<....2...sJ.A
0x0010 c63d 1013 f0b5 0259 417a 1598 0000 0000 .=....YAz.....
0x0020 a029 0800 c3a2 0000 0303 0a01 0204 0109 ).
0x0030 080a 3f3f 3f3f 0000 0000 0000 ..???.....
```

0xA029 = 10100000000101001b

If we analyze the hex output in the above traffic dump we can see the URG, PSH, and FIN flags set:

4 bit Header	Reserved Bits (6)	URG	ACK	PSH	RST	SYN	FIN
1010	000000	1	0	1	0	0	1

It is the combination of conflicting flags that cause each operating system to return differently. The FIN and PSH flags should never be set at the same time, and with the addition of the URG flag the stack is sure to be confused.

Correlations

While analyzing the alert file produced by snort, I saw the following detect which has the same source address as our attacker. This attacker seems to be gathering information about several hosts on this network.

```
[**] [1:628:1] SCAN nmap TCP [**]
[Classification: Attempted Information Leak] [Priority: 2]
09/23-14:38:54.356507 xxx.74.249.65:61618 -> xxx.61.16.19:21
TCP TTL:50 TOS:0x0 ID:42919 IpLen:20 DgmLen:60
***A**** Seq: 0x4AC38CDD Ack: 0x0 Win: 0x800 TcpLen: 40
TCP Options (5) => WS: 10 NOP MSS: 265 TS: 1061109567 0 EOL
[Xref => arachnids 28]
```

I also posted my analysis to the intrusions@incidents.org mailing list but I did not

* <http://www.insecure.org/nmap/>

receive any replies as of the date of the submission of this paper. My post to the list can be found here:

<http://cert.uni-stuttgart.de/archive/intrusions/2003/01/msg00026.html>

Evidence of Active Targeting

There seems to be evidence of active targeting due to the fact the attacker has focused on a particular host and is further probing the host in an effort to identify open ports and the Operating System type.

Severity

We can calculate severity using the following formula:

$(\text{Target's Criticality} + \text{Lethality of Attack}) - (\text{System Defense} + \text{Network Defense})$

Criticality	1	The type of host is unknown
Lethality	1	This is only an attempt to gather information, it does not affect the host, or attempt to exploit any weakness
System Defense	1	Not much can be done to defend against this attack since the target is essentially the TCP stack, which is written into the OS code
Network Defense	2	Filtering Routers, Firewalls

When we plug these values into the severity formula we get a result of -1 :

$\text{Severity} = (1 + 1) - (1 + 2) = -1$

We have a severity of -1 , so this attack does not seem appear to be a big threat, only a simple probe to gather information, but should be recognized as a red flag that an attack may be up and coming in the near future.

Defensive Recommendation

The best defense for this attack is to have a stateful Firewall in place, such as Checkpoint FW1. A stateful firewall is one that is enabled with stateful packet analysis capabilities; each connection is tracked and any packet that does not belong to an established connection is dropped by the firewall. So with a stateful firewall in place, an XMAS scan will be dropped, and will not reach any machine protected by the firewall.

Multiple Choice Test Question

Which TCP flags will be set in a XMAS Scan Packet?

- a) SYN and FIN
- b) SYN, ACK, PSH
- c) ACK, PSH, and URG
- d) URG, PSH, and FIN

The answer is d. The URG, PSH, and FIN packets are the indication of an nmap XMAS scan.

References for this Assignment

SANS Institute
Courseware: "IDS Signatures and Analysis"

Incidents Mailing List
<http://online.securityfocus.com/archive/75/>

CERT[®] Advisory CA-2001-11 sadmind/IIS Worm:
<http://www.cert.org/advisories/CA-2001-11.html>

CERT[®] Advisory CA-2001-11 sadmind/IIS Worm
www.cert.org/advisories/CA-2002-17.html

Apache HTTP Server chunked encoding heap buffer overflow
http://www.iss.net/security_center/static/9249.php

SecurityFocus Chunked Encoding Information
<http://online.securityfocus.com/bid/5033/discussion/>

iDefense analysis of chunked encoding worm
<http://www.iddefense.com/Intell/CI063002.html>

My Post to Incidents
<http://online.securityfocus.com/archive/75/295887/2002-10-18/2002-10-24/2/>

Dshield
<http://www.dshield.org>

Snort SID Lookup Utility
<http://www.snort.org/snort-db/sid.html>

Google Search Engine
<http://www.google.com>

Assignment #3 Analyze This

Executive Summary

Our job as analysts is ultimately to find the proverbial needle in the haystack. Throughout this assignment many false positives will arise. The false positives can be thought of as the hay. The needle or needles are the events that interest us, such as alerts that indicate possible compromise or worms, Trojans, and other malicious activity.

Part of the analysis process is to weed out the false positives, so that a better analysis can be made of what is left behind. Most of the traffic in this assignment can be redirected to /dev/null. The snort configuration, if properly configured can help us in attaining this goal. By disabling signatures of known false positives we can "lighten the load."

Although some traffic seems to be malicious at first, by applying the simple analysis method, introduced in the Relational Analysis Process section, we can determine quickly and efficiently if this is the case. Some of this traffic would never be seen if certain security measures were put into place, these are be discussed in the Defensive Recommendations section.

The logs files were massive, in part to single alerts that are logged many times. A small group of hosts are responsible for most of the alerts and can be found in the Top Talkers Section. As we begin to dive into the analysis, remember, "Needle in the Haystack!"

List of Analyzed Files

For this assignment I chose to analyze the log files from December 4th through December 8th 2002. Below is a list of those files:

alert.021204
alert.021205
alert.021206
alert.021207
alert.021208

For simplicity I chose to concatenate all these alert files into one complete file, which I named *alert.all*.

scans.021204
scans.021205
scans.021206
scans.021207
scans.021208

I also chose to concatenate these files into one file, which I named *scans.all*.

OOS_Report_2002_12_04_19685
OOS_Report_2002_12_05_32638
OOS_Report_2002_12_06_552
OOS_Report_2002_12_07_22540
OOS_Report_2002_12_08_23488

Just like the other two sets of files, I also chose to concatenate all the Out of Spec files into one file as well, unambiguously named *OOS.all*.

It should be noted that the Snort Portscan Preprocessor events were removed from the alert files, due to the fact that the University supplied me with the *scans* files. Tod Beardsley also did the same in his paper, which can be found here: http://www.giac.org/practical/Tod_Beardsley_GCIA.doc

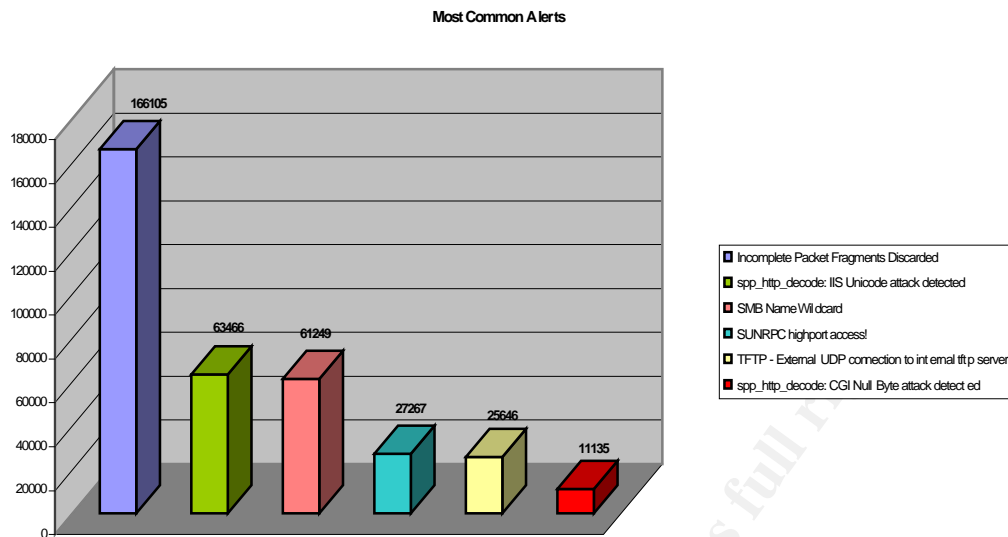
Prioritized Detects

In this section I will be analyzing events that have appeared in the alert files more than 10000 times. When I aggregated the summaries of the events into one file and summed* the unique ones, I found which events were being triggered the most. Ten thousand occurrences seemed to be a good cut-off point when I looked at the output, due to the fact that there was a gap of around five thousand to the next noisiest in line. So, if the squeaky wheel gets the grease, then the noisy alerts will be getting the analysis. Lets take a look at the top alerts and number of occurrences from the previously mentioned output:

Incomplete Packet Fragments Discarded	166105
spp_http_decode: IIS Unicode attack detected	63466
SMB Name Wildcard	61249
SUNRPC highport access!	27267
TFTP - External UDP connection to internal tftp server	25646
spp_http_decode: CGI Null Byte attack detected	11135

Below is what these alerts look like relationally when compared graphically:

* cat alert.all | awk -F'[\\|*\\|*\\|\\|]' '{print \$2}' | sort | uniq -c | sort -rn



Incomplete Packet Fragments Discarded

Reported: 166105 times

```
12/05-05:32:35.349925 [**] Incomplete Packet Fragments Discarded [**] MY.NET.190.100:0 -> 209.81.41.149:0
12/05-05:32:35.458140 [**] Incomplete Packet Fragments Discarded [**] MY.NET.190.100:0 -> 209.81.41.149:0
12/05-05:32:35.509138 [**] Incomplete Packet Fragments Discarded [**] MY.NET.190.100:0 -> 209.81.41.149:0
12/05-05:32:35.625318 [**] Incomplete Packet Fragments Discarded [**] MY.NET.190.100:0 -> 209.81.41.149:0
12/05-05:32:35.733063 [**] Incomplete Packet Fragments Discarded [**] MY.NET.190.100:0 -> 209.81.41.149:0
```

Summary:

This event was by far the noisiest of all the alerts, and is triggered because packet fragments were detected, but not all the packets arrived, therefore the stream could not be reassembled. After an exhaustive search through the log files for a stimulus for this activity, none was found, but I did notice that each connection that triggered this alert had both a source and destination port of 0. This activity could be due to several things, possibly a misconfiguration or a router corrupting packets. But it could also be crafted packets designed for a DOS since obviously the OS stacks were not designed to accept connections on this port or to create a connection with 0 as the source port. Obviously there is a problem with connections that utilize port 0, either as a source or a destination, which is not specified in the TCP RFC.*

I think this alert is just noise and the signature should be tuned or disabled. It comprised the majority of all alerts combined and was mostly triggered by internal host MY.NET.190.100. It seems as if this has been an ongoing issue because the alerts showed up across all five days worth of data. If this activity

* <http://www.rfc-editor.org/rfc/rfc793.txt>

continues to occur I would investigate this host further to see why this is occurring.

Correlations:

David Jenkins mentions this in his paper^{**} but only briefly.

spp_http_decode: IIS Unicode attack detected

Reported: 63466 times

```
spp_http_decode: IIS Unicode attack detected [**] MY.NET.53.60:4720 -> 210.219.197.27:80
spp_http_decode: IIS Unicode attack detected [**] MY.NET.53.60:4720 -> 210.219.197.27:80
spp_http_decode: IIS Unicode attack detected [**] MY.NET.53.60:4720 -> 210.219.197.27:80
spp_http_decode: IIS Unicode attack detected [**] MY.NET.53.60:4720 -> 210.219.197.27:80
spp_http_decode: IIS Unicode attack detected [**] MY.NET.53.60:4720 -> 210.219.197.27:80
spp_http_decode: IIS Unicode attack detected [**] MY.NET.53.60:4720 -> 210.219.197.27:80
```

Summary:

According to John Berkers^{***}: "The http_decode preprocessor normaliz[es] any unicode representations of characters and then passes them back to snort for matching against rules. If a particular pattern of unicode characters is detected the ISS Unicode attack event is logged, (no, that's not a spelling error, it doesn't only affect MS IIS, the vuln was first discovered by ISS guys). You can turn them off by specifying -unicode and -cginull after the http_decode thusly:

preprocessor http_decode: 80 -unicode -cginull

These events are sometimes triggered by visiting sites that use multi-byte characters such as Simplified Chinese etc. "

This was the second most reported attack, and can be lethal on an unpatched system. This is just one of the many known path traversal type attacks that exploit a system via path traversal vulnerability in the IIS server. After examining the log files a little closer I noticed that a lot of different hosts were generating this event, which means one of two things, either these hosts are all operated by malicious users or these events are just false positives. I chose the latter. It looks like web traffic to a particular site that has content that the preprocessor doesn't like and is alerting on it. Also, there seems to be no pattern or previous scans to indicate that these alerts are malicious. I would consider tuning this alert or disabling it altogether as it generates only noise at this time. One of the other noise makers, the CGI Null Byte attack also seems to be a false positive. I would consider tuning that alert as well for the same reason as tuning the Unicode alert.

^{**} http://www.giac.org/practical/David_Jenkins_GCIA.doc

^{***} <http://archives.neohapsis.com/archives/snort/2001-08/0075.html>

Correlations:

Matthew Richards analyzes the IIS Unicode attack in his GCIA practical located here: http://www.giac.org/practical/matthew_richard_gcia.doc.

SMB Name Wildcard

Reported: 61249 times

Summary: SMB stands for Server Message Block and is a protocol used for sharing. SMB provides for the sharing of files, printers and other communications, especially on Windows machines, and if abused it can be dangerous from a security standpoint. In the following series of alerts, someone is attempting to gain NETBIOS information about hosts on the inside:

12/05-01:45:30.756903	[**]	SMB Name Wildcard	[**]	65.66.16.120:1029 -> MY.NET.133.205:137
12/05-01:45:30.909202	[**]	SMB Name Wildcard	[**]	65.66.16.120:1029 -> MY.NET.133.206:137
12/05-01:45:31.058159	[**]	SMB Name Wildcard	[**]	65.66.16.120:1029 -> MY.NET.133.207:137
12/05-01:45:31.207485	[**]	SMB Name Wildcard	[**]	65.66.16.120:1029 -> MY.NET.133.208:137
12/05-01:45:31.372144	[**]	SMB Name Wildcard	[**]	65.66.16.120:1029 -> MY.NET.133.209:137

Correlations:

Toshi Iijima mentions this detect briefly in his GCIA practical located here: http://www.giac.org/practical/Toshi_Iijima_GCIA.doc. Toshi describes this as “a query for netbios information” and that it should be considered as a recon attempt from external sources. I agree with this assessment, and would like to add that any NETBIOS or Windows type file sharing originating outside of the home network should be considered malicious. When analyzing the traffic patterns in the alert file I noticed that most of this traffic is originating from the outside. This traffic is either the result of a misconfiguration on the part of the source or malicious users scanning for open shares.

SUNRPC highport access!

Reported: 27267 times

Summary:

This alert is triggered when there is an attempted connection to port 32771. Generally this is an RPC port on a Solaris system. But can also be the source port of a connection. This event was mostly triggered by scans to this port from external machines and Instant Messaging Clients such as Yahoo Messenger and AOL instant messenger. Some events were actually triggered when a user checked his Yahoo mail account and the source port was 32771.

The RPC services do not have a good reputation for being secure, and are prone to scans such as the ones present in the log files.

```
12/05-11:14:30.666987 [**] SUNRPC highport access! [**] 213.115.19.12:80 -> MY.NET.100.10:32771
12/05-11:14:30.671145 [**] SUNRPC highport access! [**] 213.115.19.12:80 -> MY.NET.100.10:32771
12/05-11:14:31.312650 [**] SUNRPC highport access! [**] 213.115.19.12:80 -> MY.NET.100.10:32771
12/05-11:14:31.312779 [**] SUNRPC highport access! [**] 213.115.19.12:80 -> MY.NET.100.10:32771
12/05-11:14:31.484090 [**] SUNRPC highport access! [**] 213.115.19.12:80 -> MY.NET.100.10:32771
12/05-11:14:31.487358 [**] SUNRPC highport access! [**] 213.115.19.12:80 -> MY.NET.100.10:32771
```

The above trace is alarming to me. Usually when a scan is launched, it is across multiple hosts, but this one is different. The source and destination remain the same, as if a connection existed between the two. Also notice that the source port is 80, which is a stealth mechanism to get by the firewall and screening routers as if this packet was in response to a web page request. That is not the case because 213.115.19.12 is the stimulus in this connection. I confirm this by connecting to this host and finding that no web server exists on this host. When I attempt a connection to port 80, no connection is made, which means there is no server listening on that port. So that means that this detect is the work of a low source port scan*. I would investigate the destination host to see if offered any RPC services, and if so, then shut them down.

TFTP – External UDP Connection to Internal TFTP Server

Reported: 25646 times

```
TFTP - External UDP connection to internal tftp server [**] 63.250.205.15:16883 -> MY.NET.153.137:69
TFTP - External UDP connection to internal tftp server [**] 63.250.205.10:16883 -> MY.NET.153.165:69
TFTP - External UDP connection to internal tftp server [**] 63.250.205.10:16883 -> MY.NET.153.165:69
TFTP - External UDP connection to internal tftp server [**] 212.113.174.194:16883 -> MY.NET.84.198:69
TFTP - External UDP connection to internal tftp server [**] 24.90.124.187:4739 -> MY.NET.177.52:69
```

Summary:

The above log entries are the only External connections to an internal machine. At first glance it seems that there are TFTP servers running on several internal machines but I do not believe this is the case. When I perform a DNS lookup on the addresses I see why. 63.250.205.15 and 63.250.205.10 actually turn out to be Yahoo media servers. The lookup returns as wmcontent30.bcst.yahoo.com and wmcontent13.bcst.yahoo.com respectively. When I connect to that address via browser, an asf video is attempted to be opened by the browser, but the connection fails.

The other two addresses resolved to a Road Runner cable modem user (24-90-124-187.nyc.rr.com) and the other is a host in Portugal(a212-113-174-194.netcabo.pt). This doesn't seem normal so I dig deeper by looking up the registration information for these hosts, and it is listed below:

* A scan by which low source ports are used in an effort to bypass access control lists and firewall rules; these can appear to be established connections, but are crafted packets.

Registration Information for 24.90.124.187

OrgName: ROADRUNNER-NYC
OrgID: RRYN

NetRange: 24.90.0.0 - 24.90.255.255
CIDR: 24.90.0.0/16
NetName: ROADRUNNER-NYC-2
NetHandle: NET-24-90-0-0-1
Parent: NET-24-0-0-0-0
NetType: Direct Allocation
NameServer: DNS1.RR.COM
NameServer: DNS2.RR.COM
NameServer: DNS3.RR.COM
NameServer: DNS4.RR.COM
Comment: ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
RegDate: 2001-07-12
Updated: 2002-04-08

TechHandle: ZS30-ARIN
TechName: ServiceCo LLC
TechPhone: +1-703-345-3416
TechEmail: abuse@rr.com

OrgAbuseHandle: ABUSE10-ARIN
OrgAbuseName: Abuse
OrgAbusePhone: +1-703-345-3416
OrgAbuseEmail: abuse@rr.com

OrgTechHandle: IPTEC-ARIN
OrgTechName: IP Tech
OrgTechPhone: +1-703-345-3416
OrgTechEmail: abuse@rr.com

OrgTechHandle: IPCON-ARIN
OrgTechName: IPControl
OrgTechPhone: +1-703-345-3416
OrgTechEmail: tconley@va.rr.com

ARIN Whois database, last updated 2002-12-28 20:00
Enter ? for additional hints on searching ARIN's Whois database.

Registration Information for 212.113.174.194

% This is the RIPE Whois server.
% The objects are in RPSL format.
%
% Rights restricted by copyright.
% See <http://www.ripe.net/ripencb/pub-services/db/copyright.html>

inetnum: 212.113.174.0 - 212.113.178.255
netname: TVCABO
descr: TVCABO-Portugal HDI-Datacenter Network
country: PT
admin-c: TVCA1-RIPE
tech-c: TVCT1-RIPE

status: ASSIGNED PA
remarks: ABUSE REPORTS MUST BE SEND TO ABUSE@TVCABO.PT
notify: tvcabo.adm@tvcabo.pt
mnt-by: ID414-MNT
changed: rfonseca@tvcabo.pt 20020710
source: RIPE

route: 212.113.160.0/19
descr: TVCABO-Portugal
origin: AS12542
notify: rfonseca@tvcabo.pt
mnt-by: ID414-MNT
changed: id@tvcabo.pt 19990823
changed: rfonseca@tvcabo.pt 20020507
source: RIPE

role: TvCabo Admin Contact
address: Avenida 5 de Outubro, 208
address: Edifício Santa Maria
address: 9 andar
address: 1069-203 Lisboa
phone: + 351 217824760
phone: + 351 217914800
fax-no: + 351 217824896
e-mail: tvcabo.adm@tvcabo.pt
trouble: Abuse Reports abuse@tvcabo.pt
trouble: Network Issues tvcabo.tech@tvcabo.pt
admin-c: TVCA1-RIPE
tech-c: TVCT1-RIPE
nic-hdl: TVCA1-RIPE
remarks: TvCabo Administrative Contact
notify: tvcabo.adm@tvcabo.pt
mnt-by: ID414-MNT
changed: rfonseca@tvcabo.pt 20011119
source: RIPE

role: TvCabo Tech Contact
address: Avenida 5 de Outubro, 208
address: Edifício Santa Maria
address: 9 andar
address: 1069-203 Lisboa
phone: + 351 217824760
phone: + 351 217914800
fax-no: + 351 217824896
e-mail: tvcabo.tech@tvcabo.pt
trouble: Abuse Reports abuse@tvcabo.pt
trouble: Network Issues tvcabo.tech@tvcabo.pt
admin-c: TVCA1-RIPE
tech-c: TVCT1-RIPE
nic-hdl: TVCT1-RIPE
remarks: TvCabo Technical Contact
notify: tvcabo.tech@tvcabo.pt
mnt-by: ID414-MNT
changed: rfonseca@tvcabo.pt 20011119
source: RIPE

Aside from the hosts listed in the previous detect log, the majority of these alerts

were caused by a connection from an internal machine to a 192.168.0.253 address. While I am unaware of the IP addressing scheme used by the university, this may be an actual internal to internal connection or the 192.168 address may be a spoofed address. TFTP servers can be dangerous because no authentication is required. If it is necessary to have tftp servers for some reason their use should be limited, and have access control lists in place to help. If there are no tftp servers on the inside and there are pre-existing rules to block this type of activity, I would consider removing this rule from the signature set.

CGI Null Byte Attack Detected

Reported: 11135 times

Summary:

This detect is produced by the snort decode preprocessor.

```
spp_http_decode: CGI Null Byte attack detected [**] MY.NET.153.176:2858 -> 66.129.106.116:80
spp_http_decode: CGI Null Byte attack detected [**] MY.NET.153.176:2858 -> 66.129.106.116:80
spp_http_decode: CGI Null Byte attack detected [**] MY.NET.153.176:2861 -> 66.129.106.116:80
spp_http_decode: CGI Null Byte attack detected [**] MY.NET.153.176:2861 -> 66.129.106.116:80
spp_http_decode: CGI Null Byte attack detected [**] MY.NET.153.176:2861 -> 66.129.106.116:80
spp_http_decode: CGI Null Byte attack detected [**] MY.NET.153.176:2861 -> 66.129.106.116:80
spp_http_decode: CGI Null Byte attack detected [**] MY.NET.153.176:2861 -> 66.129.106.116:80
spp_http_decode: CGI Null Byte attack detected [**] MY.NET.153.176:2861 -> 66.129.106.116:80
```

An alert is produced whenever %00 is contained within a CGI form. The decode preprocessor sees %00 and decodes it to the NULL character, which can be used for IDS evasion, since the %00 skews the normal signature. This can be used in conjunction with path traversal attacks to exploit a host without being detected by the IDS. Joe Ellis mentions this detect in his practical located here: http://www.giac.org/practical/Joe_Ellis_GCIA.doc.

According to Joe Ellis, "This alert can trigger many false positives, and can be turned off by adding the "-cginull" option to the line "preprocessor http_decode:" in Snort's alert.ids file."

Disabling this alert would further help to cut down on the number of false positives, making the analyst(s) job much less painful. Remember, we are looking for the needle and not the hay!

Relational Analysis Process

Naturally some events grab our attention more than others such as the "Possible Trojan Server activity" alerts. It is with this group of alerts I would like to describe the process by which I determined which events were possible threats and which were possible false positives. By going through a process of elimination I started with 4845 of these alerts and narrowed it down to only 2 alerts that needed more analysis. First we begin with the elbow grease: `grep` for all the alerts with

* `grep "Possible Trojan Server" alert.all`

“Possible Trojan Server” as the summary, this will aggregate all similar events. This generates 4845 alerts, which is too many to deal with at once so next we look for false positives and throw those out first. When I begin the analysis of the output produced by the grep command, I notice that all of these events are triggered with 27374 as either a source or destination port. Source port 27374 is associated with the Ramen Worm and destination port 27374 is associated with the Subseven Trojan.

```
Possible trojan server activity [**] 80.62.74.110:27374 -> MY.NET.185.48:6346
Possible trojan server activity [**] 80.62.74.110:27374 -> MY.NET.185.48:6346
Possible trojan server activity [**] MY.NET.185.48:6346 -> 80.62.74.110:27374
Possible trojan server activity [**] MY.NET.185.48:6346 -> 80.62.74.110:27374
```

To begin weeding out the false positives, I take a look at the above alerts and notice that 80.62.74.110 seems to be the stimulus for this alert. The Ramen uses source port 27374, but this isn't a worm, a worm usually tries to spread across multiple hosts, but the source is only communicating with one internal host. The exchange between these two hosts generates 4623 of all “Possible Trojan” events. Also, notice the destination port of 6346. It appears as if MY.NET.185.48 is a member of the Gnutella file-sharing network, and is actively sharing files with other members. That takes care of the majority of the false positives, so we no longer need see this exchange. To weed out this exchange I simply add “ | grep -v :6346 ” to the end of my previous command, this will ignore the Gnutella connections.

With the number of alerts dramatically reduced, I now begin to look for other false positives and I find them: ports 80, 1214, and 4662. The alerts with port 80 seemed to be web traffic with 27374 as the source port, the port 1214 alerts were caused by the Kazza / Morpheus file-sharing utility, and 4662 was eDonkey2000, another file-sharing application. By parsing out the known activity we reduce the number of alerts down to 2. This makes the analysis process much easier now:

```
Possible trojan server activity [**] 65.88.96.76:1044 -> MY.NET.135.84:27374
Possible trojan server activity [**] 63.161.29.66:3677 -> MY.NET.135.190:27374
```

Now I search for ports 1044 and 3677 using a port lookup utility** and a search engine to determine if the ports are associated with any known service, and I don't turn up anything. Port 1044 seems suspicious because the OS usually begins to select source ports at 1024 and increments with each connection, so it would not take many connections to reach 1044, which can indicate a direct connection to destination port 27374 on one of the internal machines. But because there is no response to either of these probes, it is assumed that the probe was blocked or ignored by the host. So all is well with the “Possible Trojan

** Treachery Unlimited Port Lookup Utility - http://www.treachery.net/security_tools/ports/

Server” alerts except for the file-sharing-bandwidth-bandits.

So the analysis process looks like this:

- 1: Aggregate and Correlate similar activity
- 2: Identify the false positives and filter out with `grep -v`
- 3: Group the remaining alerts by port, by source then destination
- 4: Identify the remaining traffic – port lookup and search engine
- 5: Determine if connection is a stimulus or a response
- 6: Determine if malicious

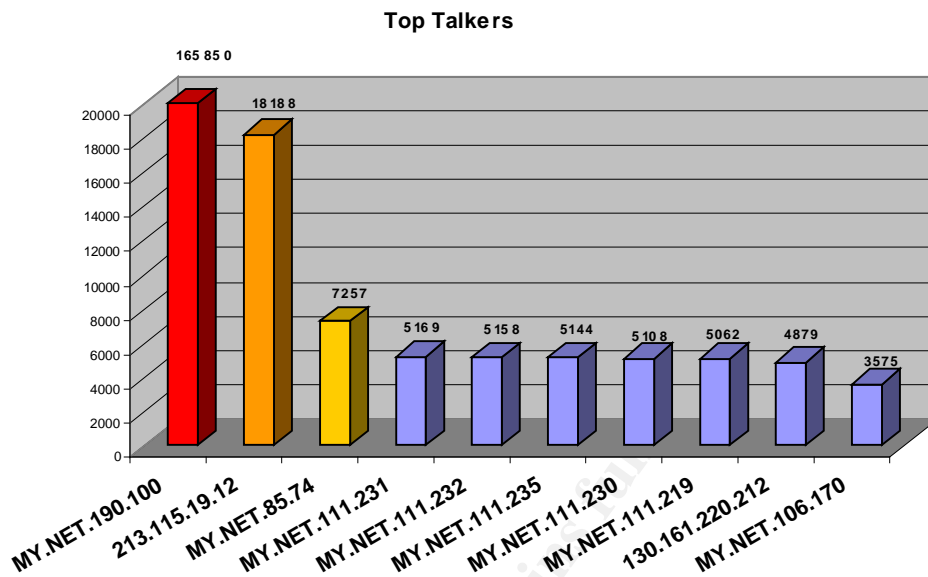
Following the above steps helped to reduce the number of alerts analyzed and helped identify the noise early on.

Top Talkers

Below is a list of the top talkers chosen by the number of alerts generated in the alerts files over the five day period:

MY.NET.190.100	165850
213.115.19.12	18188
MY.NET.85.74	7257
MY.NET.111.231	5169
MY.NET.111.232	5158
MY.NET.111.235	5144
MY.NET.111.230	5108
MY.NET.111.219	5062
130.161.220.212	4879
MY.NET.106.170	3575

The Top Talkers would look like this graphically:



Notice that the ceiling of the graph has been adjusted, due to the fact that MY.NET.190.100 generated so many alerts, this was done so that a visual comparison could be made among the top talkers.

We have two external addresses that have managed to make their way onto the Top Talkers list, so let's see who they belong to:

Registration information for 213.115.19.12:

Final results obtained from whois.ripe.net.

Results:

% This is the RIPE Whois server.

% The objects are in RPSL format.

%

% Rights restricted by copyright.

% See <http://www.ripe.net/ripenncc/pub-services/db/copyright.html>

inetnum: [213.112.0.0](http://whois.ripe.net/213.112.0.0) - [213.115.255.255](http://whois.ripe.net/213.115.255.255)

netname: SE-CYBER-20000314

descr: Provider Local Registry

country: SE

admin-c: ELO2-RIPE

tech-c: BR3045-RIPE

status: ALLOCATED PA

mnt-by: RIPE-NCC-HM-MNT

mnt-lower: B2-MNT

mnt-routes: B2-MNT

changed: hostmaster@ripe.net 20000314

changed: hostmaster@ripe.net 20000315

changed: hostmaster@ripe.net 20000316
changed: hostmaster@ripe.net 20001215
changed: lir-help@ripe.net 20011214
source: RIPE

route: [213.112.0.0/14](#)
descr: Broadband Customers in Scandinavia
descr: Please report improper use to abuse@bredband.com
origin: AS8642
notify: noc@bredband.com
mnt-by: B2-MNT
changed: [anton.gunnarsson@bredband.com](#) 20001215
changed: [tommy.nilsson@bredband.com](#) 20020408
source: RIPE

role: Bredbandsbolaget Rouingregistry
address: Stockholm, Sweden
e-mail: noc@bredband.com
trouble: Abuse related issues is reported
trouble: to abuse@bredband.com
trouble: phone +46 586 65485
admin-c: TN2809-RIPE
tech-c: TN2809-RIPE
admin-c: JN1883-RIPE
tech-c: JN1883-RIPE
admin-c: EB78-RIPE
tech-c: EB78-RIPE
admin-c: NE102-RIPE
tech-c: NE102-RIPE
nic-hdl: BR3045-RIPE
mnt-by: B2-MNT
notify: noc@bredband.com
changed: [jonas.nylund@bredband.com](#) 20020418
changed: [jonas.nylund@bredband.com](#) 20020425
changed: [nicklas.eriksson@bredband.com](#) 20021004
source: RIPE

person: Anders Elo
address: Bredbandsbolaget AB
address: Ingenjorsv. 3
address: S-11743 Stockholm
address: Sweden
phone: +46 8 55632500
e-mail: [anders.elo@bredband.com](#)
nic-hdl: ELO2-RIPE
remarks: Please report all abuse related issues to our
remarks: abuse-department, abuse@bredband.com
remarks: +46 586 65485
notify: [anders.elo@bredband.com](#)
changed: [anders.elo@bredband.com](#) 20010201
source: RIPE

Registration Information for 130.161.220.212:

Final results obtained from whois.arin.net.

Results:

OrgName: Technische Universiteit Delft

OrgID: TUD-1

NetRange: 130.161.0.0 - 130.161.255.255

CIDR: 130.161.0.0/16

NetName: DUNET

NetHandle: NET-130-161-0-0-1

Parent: NET-130-0-0-0-0

NetType: Direct Assignment

NameServer: NS1.TUDELFT.NL

NameServer: NS2.TUDELFT.NL

NameServer: NS1.SURFNET.NL

NameServer: NS1.ET.TUDELFT.NL

Comment:

RegDate: 1988-08-26

Updated: 2000-11-10

TechHandle: FD18-ARIN

TechName: Kruijf, Freek

TechPhone: +31 15 2783226

TechEmail: SSC@tudelft.nl

Host 213.115.19.12 seems to be familiar, we seen this host back in the analysis of the SUNRPC scans.

Out of Spec

As I analyzed the OOS.all file I noticed that three particular out of spec combinations of flags were present:

12****S*	5457
****P***	1236
*****	1208

These three alone account for 7901 or 98% of the total 8047 OOS packets. The most common combination had both of the reserved bits set as well as SYN bit. The second one only has the PSH bit set, but it does not have the ACK bit. The third one doesn't have any flags set at all.

Most of the 12S and P packets were caused by file-sharing applications. Two of the common ones were Kazaa and Morpheus, both of which utilize the Gnutella file-sharing network. The protocols are very similar in these utilities and the flag settings do not adhere to the specifications set forth in the TCP RFC.

Gnutella also contributed to the third combination, but only minimal. This time

the culprit was destination port 37 or the time protocol. Time updates accounted for 92% of the total 1208 alerts to this out of spec combination.

Common applications are to blame for most of the OOS alerts. In particular Gnutella clients and time updates. The remainder of the OOS packets were mostly one time occurrences and can be attributed to packet corruption.

Insights about Internal Machines

Host 130.85.190.100 interests me for several reasons. When I correlate the top talkers list with the scans list 130.85.190.100 stands out in both. This host appeared on the top talkers list due to incomplete packet fragments, which seemed to have something to do with source port 0 to destination port 0 traffic. That was suspicious enough, and if it were my host I would look into the cause of that traffic, but when I began to analyze the scans file I noticed that this host was scanning external hosts for open ports 445 and 139, which are NETBIOS related ports, which means it was scanning for hosts that had sharing enabled. See logs below:

```
Dec 7 06:34:18 130.85.190.100:4269 -> 128.121.97.106:445 SYN *****S*
Dec 7 06:34:18 130.85.190.100:4270 -> 128.121.97.106:139 SYN *****S*
Dec 7 06:34:20 130.85.190.100:4319 -> 128.121.97.108:445 SYN *****S*
Dec 7 06:34:20 130.85.190.100:4320 -> 128.121.97.108:139 SYN *****S*
Dec 7 06:34:20 130.85.190.100:4323 -> 128.121.97.109:445 SYN *****S*
Dec 7 06:34:20 130.85.190.100:4324 -> 128.121.97.109:139 SYN *****S*
```

Notice how it scans each host for port 445 then for port 139. Just to analyze a little deeper, it seems that this may be the work of the Opaserv worm, in any case, let's take a look at the destination network to see if it would be a worthwhile target for this scanner's hard work:

Registration Information for 128.121.97.106:

```
OrgName:      Verio, Inc.
OrgID:        VRIO

NetRange:     128.121.0.0 - 128.121.255.255
CIDR:         128.121.0.0/16
NetName:      VRIO-128-121
NetHandle:    NET-128-121-0-0-1
Parent:       NET-128-0-0-0-0
NetType:      Direct Allocation
NameServer:   NS0.VERIO.NET
NameServer:   NS1.VERIO.NET
NameServer:   NS2.VERIO.NET
Comment:      *****
               Reassignment information for this block is
               available at rwhois.verio.net port 4321
               *****

RegDate:      2000-07-11
Updated:      2001-09-26
```

TechHandle: VIA4-ORG-ARIN
TechName: Verio, Inc.
TechPhone: +1-303-645-1900
TechEmail: vipar@verio.net

OrgAbuseHandle: VAC5-ARIN
OrgAbuseName: Verio Abuse Contact
OrgAbusePhone: +1-800-551-1630
OrgAbuseEmail: abuse@verio.net

OrgNOCHandle: VSC-ARIN
OrgNOCName: Verio Support Contact
OrgNOCPhone: +1-800-551-1630
OrgNOCEmail: support@verio.net

OrgTechHandle: VIA4-ORG-ARIN
OrgTechName: Verio, Inc.
OrgTechPhone: +1-303-645-1900
OrgTechEmail: vipar@verio.net

ARIN Whois database, last updated 2002-12-23 20:00
Enter ? for additional hints on searching ARIN's Whois database.

Rwhois server data:

%rwhois V-1.5:0078b6:00 rwhois.verio.net (Vipar 0.1a. Comments to
vipar@verio.net)
network:Class-Name:network
network:Auth-Area:128.121.64.0/18
network:ID:NETBLK-W042-128-121-97.127.0.0.1/32
network:Handle:NETBLK-W042-128-121-97
network:Network-Name:W042-128-121-97
network:IP-Network:128.121.97.0/24
network:In-Addr-Server;I:NS8629-HST.127.0.0.1/32
network:In-Addr-Server;I:NS8630-HST.127.0.0.1/32
network:IP-Network-Block:128.121.97.0 - 128.121.97.255
network:Org-Name:**Verio Web Hosting** - San Jose
network:Street-Address:250 Stockton Ave
network:City:San Jose
network:State:CA
network:Postal-Code:95126
network:Country-Code:US
network:Tech-Contact;I:WA577-VRIO.127.0.0.1/32
network:Created:2001-10-05 21:06:45+00
network:Updated:2002-07-17 20:59:03+00

network:Class-Name:network
network:Auth-Area:128.121.64.0/18
network:ID:NETBLK-VRIO-128-121-064.127.0.0.1/32
network:Handle:NETBLK-VRIO-128-121-064
network:Network-Name:VRIO-128-121-064
network:IP-Network:128.121.64.0/18
network:In-Addr-Server;I:NS8629-HST.127.0.0.1/32
network:In-Addr-Server;I:NS8630-HST.127.0.0.1/32
network:IP-Network-Block:128.121.64.0 - 128.121.127.255
network:Org-Name:**Verio Web Hosting** - San Jose

```
network:Street-Address:250 Stockton Ave
network:City:San Jose
network:State:CA
network:Postal-Code:95126
network:Country-Code:US
network:Tech-Contact;I:WA577-VRIO.127.0.0.1/32
network:Created:2001-02-02 20:51:05+00
network:Updated:2002-07-17 20:57:19+00
```

The scanned host in this example is very interesting, it is the Verio Web Hosting Company. This company has a big network, an entire class B, with most of those being active web servers. So the scanning activity does not seem to be benign at all. This is bad. This host does not seem to be functioning normally in two cases. It is the source of bad traffic, and is scanning external hosts. This host should be investigated for compromise.

Defensive Recommendations

Based on the analysis of the university's log files and with all the noise that is present, I would consider tuning filters so that more threatening attacks and probes stand out. The sole purpose in intrusion detection is to find the needle in the haystack, and the more sensors we have the more haystacks we have. So to effectively monitor a network we need to cut down the size of each haystack to a manageable size. This can be done through signature tuning as well as disabling particular signatures that are known to be noisy or that trigger only false positives.

I would also recommend that ingress and egress filtering be put into place for all Windows specific communications, such as file-sharing and domain controller activity. This can prevent worms that originate on the Internet from infecting internal hosts. The opaserv worm has been known to do this, and by filtering the egress traffic, the spread of any worm can also be stopped before reaching other computers on the outside. This would also block the traffic originating from the suspicious internal host 130.85.190.100 from reaching its destination.

Assuming that the Snort sensor is inside the firewall, it appears as though crafted packets are entering the network and getting by the firewall. I would suggest that stateful packet filtering firewall be put into place. This would drop any packets that not belong to a connection, or otherwise do not correspond to the specifications set forth in the TCP RFC, such as crafted packets, and would help prevent against nmap scans.

In addition to IDS monitoring, the log files for any public servers, such as web, ftp, dns, mail, etc. should also be aggregated along with the IDS alerts so that a correlation can be made between the devices to offer more visibility to the intrusion detection analyst.

References for this Assignment

Tod Beardsley's GCIA Practical

http://www.giac.org/practical/Tod_Beardsley_GCIA.doc

Neophasis Archives - John Berkers Unicode Comments

<http://archives.neohapsis.com/archives/snort/2001-08/0075.html>

Geektools

www.geektools.com

The TCP Request For Comment

<http://www.rfc-editor.org/rfc/rfc793.txt>

David Jenkins GCIA practical

http://www.giac.org/practical/David_Jenkins_GCIA.doc

Treachery Unlimited Port Lookup Utility

http://www.treachery.net/security_tools/ports/

NMAP Port Scanner

<http://www.insecure.org/nmap/>

Matthew Richards GCIA Practical

http://www.giac.org/practical/matthew_richard_gcia.doc

Toshi Iijima GCIA Practical

http://www.giac.org/practical/Toshi_Iijima_GCIA.doc

Joe Ellis GCIA Practical

http://www.giac.org/practical/Joe_Ellis_GCIA.doc

© SANS Institute 2003. Author retains full rights.