Global Information Assurance Certification Paper

GIAC Certified Intrusion Analyst (GCIA)

Practical Assignment
Version 4.1

Kris Wicks
January 2, 2005

Executive Summary

As part of an ongoing project to update security practices at the university, a third-party audit of logs from the Snort Intrusion Detection System (IDS) from the previous year was requested by the faculty, for the purpose of security analysis. The analyst was instructed to determine whether any intrusions had taken place during a three day period, and to explain in-depth the three most critical events in the logs. The analyst was also requested to explain how such attacks could have been avoided, and to provide overall technical recommendations for security enhancement.

Critical Events

Due to the limited information made available to the analyst, a number of interesting detects could not be substantiated. Alert logs with more in-depth packet information, or provision of raw packet dumps, along with access to system logs for specific servers or routers on the university network, would have allowed for much more detailed analysis.

- The first attack listed was launched against student computers running their own web servers. The attack was probably against an optional service known as WebDAV, running along with Internet Information Server on these Windows systems, which would allow full access for the attacker to run malicious applications or damage the system if successfully compromised.

- The second attack was a group of attackers searching for Windows computers with open access to the C drive. A number of systems appear to have been successfully compromised with this attack, which is due to a configuration error on some Windows systems, rather than a software vulnerability.

- The last attack detailed was a system within the network which was attempting cause a Denial of Service attack on a remote system. This attack would disrupt communication on a system from another network by sending connection requests from a number of false hosts, to elicit a communication attempt between the targeted system and the false host, thus tying up resources. Enough false requests would prevent the targeted system from responding to valid communication requests. The number of false requests coming from the university network would probably not be enough to cause a DoS on the targeted system, so the university system had probably been previously compromised, and was being used by an offsite attacker as part of a larger attack.

Recommendations

Successful attacks were not detected against the university servers, although these systems were scanned many times. Rather, most successful attacks were perpetrated

against student systems running poorly patched or misconfigured versions of the Windows operating system. After a detailed analysis of the logs provided, the following determinations have been made:

- Encourage updates of all student systems through weekly email bulletins to all students.
- For ease of access, all critical patches should be downloaded and indexed by OS in a central share for simple retrieval.
- Separate administration and faculty systems from student systems by a firewall or filtering router.
- Implement IDSs within the campus network to monitor internal traffic. Define rules to detect attempts to exploit critical and protected systems from other systems on the internal network which may be compromised.
- Filter outgoing packets with falsified source IP addresses, at the internet router.
- Filter incoming and outgoing requests to known ports often used for malicious purposes (such as Trojans on 31337 or 27374), or ports whose services should not be utilized outside the campus network (such as Windows file sharing ports 137, 138, 139, and 445).

Analysis

Network Setup

It can be inferred that the more scans a host receives from unique source addresses, the more likely it is that the host's address has been published somewhere, as a public web or mail server, for example. This method can also often show us which students are running P2P apps or hosting IRC or games servers.

In the scan logs, almost all scan alerts to destination port UDP 53 originated from either MY.NET.1.3 or MY.NET.1.4, with destinations to known DNS servers. MY.NET.1.3 and MY.NET.1.4 are clearly the university's DNS servers. It's pretty common to have basic infrastructure on subnet 1, so the main router is probably also on that subnet, although no direct evidence of this is found in the logs.

A large number of packets going to port 25 on MY.NET.24.21 – 23 which were flagged as scans appear to be legitimate traffic, marking these systems as campus SMTP servers. The alerts and scans to port 80 on MY.NET.24.44 indicate that it is probably the school's public web server, and the alerts and scans to port 21 on MY.NET.24.47 indicate that this is the school's public FTP server, so MY.NET.24.x is probably a screened subnet for public-facing systems.

Server MY.NET.100.165 gets a large number of "CS Web server" alerts in the alert logs. This alert is triggered numerous times for both port 80 and port 21, indicating that this is a web server and FTP server for the CS department. There aren't enough scans to any interesting ports on any other IPs in this subnet to determine if this entire subnet is the CS department or if this is another screened subnet, although the alerts that external traffic is hitting the system are very telling.

I saw very few scans for proxies, and no alerts, which lead me to believe that internet access most likely is not proxied on this network.

Due to the complete lack of any alerts or scans from internal source IP to internal destination IP addresses, it appears that either the IDS is on the outside of the firewall or DMZ. While this does make it more difficult to determine which attacks actually reached their intended destination, a permissive firewall is common at most universities, and as such, it is assumed for the purposes of this report that alerts caught by Snort were not blocked at the firewall.

Files Utilized

The following files were downloaded from isc.sans.org for this paper. They are all from the same IDS, and no logs from any other systems on this network were available for

analysis. Relational log analysis was attempted between the three log types, but the detects mentioned here were not found in either the scan logs or the OOS logs.

Alerts: Alert.030606, Alert.030607, Alert.030608.
Scans: Scans.030606, Scans.030607, Scans.030608.
OOS Logs: OOS_Report_2003_06_06_14006, OOS_Report_2003_06_07_30098, OOS_Report_2003_06_08_22596.

Detects Identified

The following is a list of all detects found in the alert logs from June 6-8 2003, along with the total number of alerts triggered.

AlertType	Total
SMB Name Wildcard	395028
CS WEBSERVER - external web traffic	38610
External RPC call	10187
[UMBC NIDS IRC Alert] XDCC client detected attempting to IRC	8837
spp_http_decode: IIS Unicode attack detected	8386
MY.NET.30.4 activity	6536
[UMBC NIDS IRC Alert] IRC user /kill detected- possible trojan.	6013
High port 65535 tcp - possible Red Worm – traffic	4371
EXPLOIT x86 NOOP	3156
spp_http_decode: CGI Null Byte attack detected	3150
Queso fingerprint	3078
High port 65535 udp - possible Red Worm – traffic	2780
TCP SRC and DST outside network	1911
CS WEBSERVER - external ftp traffic	1467
connect to 515 from outside	1458
MY.NET.30.3 activity	1231
Possible trojan server activity	1141
IDS552/web-iis_IIS ISAPI Overflow ida nosize	802
Null scan!	552
Incomplete Packet Fragments Discarded	541
SNMP public access	448
[UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected.	316
SUNRPC highport access!	273
NMAP TCP ping!	162
Notify Brian B. 3.54 tcp	60
IRC evil - running XDCC	58
Notify Brian B. 3.56 tcp	54
SMB C access	40
EXPLOIT x86 setuid 0	29
FTP passwd attempt	27
EXPLOIT x86 setgid 0	20
EXPLOIT x86 stealth noop	15
RFB - Possible WinVNC - 010708-1	7

EXPLOIT x86 NOPS	6
External POP to HelpDesk MY.NET.70.49	4
TFTP - Internal TCP connection to external tftp server	4
Probable NMAP fingerprint attempt	4
External FTP to HelpDesk MY.NET.70.49	3
NETBIOS NT NULL session	3
External FTP to HelpDesk MY.NET.70.50	3
TFTP - Internal UDP connection to external tftp server	3
NIMDA - Attempt to execute cmd from campus host	2
EXPLOIT NTPDX buffer overflow	2
External POP to HelpDesk MY.NET.70.50	2
[UMBC NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC	2
[UMBC NIDS IRC Alert] User joining Warez channel detected. Possible XDCC bot	2
Fragmentation Overflow Attack	2

Determination of Important Detects

Snort's "Quick Alert" format, and the scan log format, did not provide very much information to determine whether the packets were crafted, or which OS the host was running. Additionally, since responses to scans were not captured, it not possible from the scan logs to tell what services are actually running on these systems, making real threat analysis much more difficult.

What was already a difficult request was made much more so by the fact that so many interesting alerts were caused by custom rules for which the specific triggers couldn't be viewed, and the lack of raw scan data to view the actual packet. For example, an overwhelming number of entries for the second most common alert (IRC user /kill detected) were from one user attempting to connect to an IRC server run by the web hosting company ColoGuys over a 7 hour period on June 6. Looks like somebody's IRC bot got k-lined, and the fact that it only ran for a period of several hours on one day would indicate that it was caught and stopped, and that the system owner probably knew about it, but additional checks within the OOS and scan logs didn't find any further data on either the source or destination.

A little bit more information would have been very beneficial for this report, and would have allowed for a much more in-depth analysis.

There were a very large number of requests, mostly outgoing, that caused either the "spp_http_decode: IIS Unicode attack detected" or "spp_http_decode: CGI Null Byte attack detected" alerts to be triggered. A search found that it is common to find a large number of false positives with these alerts, due to the way many websites format their HTML (ex, using /../../image.gif) and that many people shut them off, or at least configure their filters to ignore outbound web traffic.

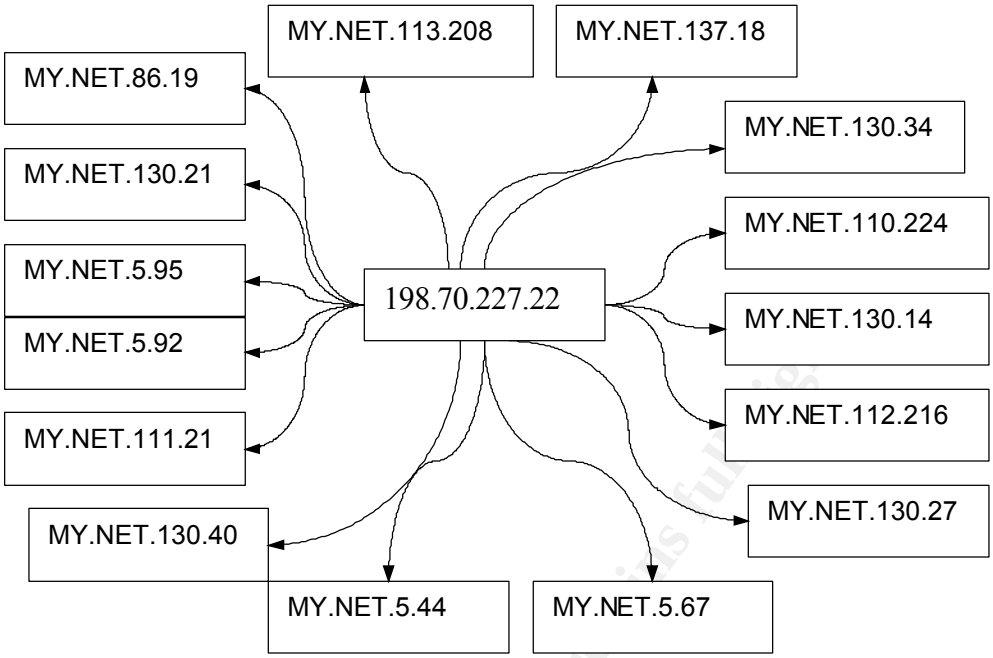
Detect #1 – WebDAV buffer overflow attack on IIS Web Services

An interesting attack occurred at 12:45 PM on June 7. The attack caused the "Exploit x86 NOOP" rule to be triggered. A fragment of the attacks are presented here:

```
06/07-12:45:07.232864  [**] EXPLOIT x86 NOOP [**] 198.70.227.22:3412 -> MY.NET.5.95:80
06/07-12:45:07.293947  [**] EXPLOIT x86 NOOP [**] 198.70.227.22:3413 -> MY.NET.86.19:80
06/07-12:45:07.309670  [**] EXPLOIT x86 NOOP [**] 198.70.227.22:3414 -> MY.NET.111.21:80
06/07-12:45:07.400201  [**] EXPLOIT x86 NOOP [**] 198.70.227.22:3412 -> MY.NET.5.95:80
06/07-12:45:07.408054  [**] EXPLOIT x86 NOOP [**] 198.70.227.22:3412 -> MY.NET.5.95:80
06/07-12:45:07.415950  [**] EXPLOIT x86 NOOP [**] 198.70.227.22:3412 -> MY.NET.5.95:80
06/07-12:45:07.433520  [**] EXPLOIT x86 NOOP [**] 198.70.227.22:3413 -> MY.NET.86.19:80
06/07-12:45:07.446986  [**] EXPLOIT x86 NOOP [**] 198.70.227.22:3413 -> MY.NET.86.19:80
06/07-12:45:07.477191  [**] EXPLOIT x86 NOOP [**] 198.70.227.22:3414 -> MY.NET.111.21:80
06/07-12:45:07.484961  [**] EXPLOIT x86 NOOP [**] 198.70.227.22:3414 -> MY.NET.111.21:80
06/07-12:45:07.492836  [**] EXPLOIT x86 NOOP [**] 198.70.227.22:3414 -> MY.NET.111.21:80
06/07-12:45:07.639526  [**] EXPLOIT x86 NOOP [**] 198.70.227.22:3413 -> MY.NET.86.19:80
06/07-12:45:07.647515  [**] EXPLOIT x86 NOOP [**] 198.70.227.22:3413 -> MY.NET.86.19:80
06/07-12:45:07.655236  [**] EXPLOIT x86 NOOP [**] 198.70.227.22:3413 -> MY.NET.86.19:80
```

This type of alert is common in attacks which are attempting to exploit buffer overflows on vulnerable systems. The entire attack lasted under a minute, and hit 15 systems. Due to the number of packets sent, and the fact that there was no delay in packets that would indicate a retransmission, all of these targets were almost certainly live and listening on TCP 80. The number of alerts per system was anywhere from 15 to 29, with the majority of systems getting about 20 alerts, indicating that the same attack was attempted on these systems. The fact that we don't see any retransmissions makes it very likely that this is a coordinated attack of known resources, and jumps in port numbers on the source system make it likely that systems on the university network aren't the only ones under attack. These systems weren't listed in any other warnings, and they the scan logs show that they weren't scanned more than normal over the course of several days, so they are probably web servers running on student systems. According to samspade.org, the attacking system 198.70.227.22 is on Sprint's DSL network. The attacker had not caused any system scan alerts during that day or the previous, making it likely that the data was collected several days previous, or that it was gathered in some manner other than an active scan.

While this alert is triggered fairly often by clients downloading executable code from web servers, that is not the case here, since the alert is triggered by packets being sent to the web servers, rather than coming from them.



Link Diagram depicting attack from 198.70.227.22

Reason this attack was selected

I selected this attack because it is a clear example of an external system compromising several internal systems within a short period, utilizing a vulnerability which was fairly new at the time.

Detect Generated By

This detect was generated by the university's Snort IDS. The only place I could find this specific rule on Snort.org was in the list of legacy rules, making me think the university is probably running Snort v1.6 or v1.7:
http://cvs.snort.org/viewcvs.cgi/sourceforge_legacy/snortold/exploit.rules?rev=1.6

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 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|"; flags: A+;
reference:arachnids,181;)
```

This alert would be triggered by any established TCP traffic coming from an external source containing at least 24 NOOPs. It's not exclusive to web traffic or webDAV, and can in some cases be triggered as a false positive when a user is downloading executable code, since a string of NOOPs is relatively common in compiled code. The ambiguity of this rule may be why it no longer one of the rules provided with Snort.

Attack Mechanism

This looks very much like an attack on the webDAV service available on MS Windows NT/2000/XP systems. None of the packets were caught in the OOS logs, so payload

analysis was not possible. However, Microsoft KB article 816930 recommends limiting the maximum request buffer to 16kb to fix the buffer overflow problem, and 20 packets would be more than enough to provide an overflow condition, plus a small amount of arbitrary code. Additionally, there are several other similar alerts in the correlating documents which could be analyzed in depth which point to webDAV. This vulnerability was discovered in late March 2003, and would have still been a fairly new attack at this point. It does not appear to be a worm, since it seems to be attacking only hosts with live web servers. Rather, it appears to be an attack on specific systems. The high number of systems hit in such a short period of time indicates that the exploit was run as a script.

The webDAV buffer overflow vulnerability was caused by a vulnerable system file, ntdll.dll, which contained an unchecked buffer which if exploited correctly could allow an attacker to run arbitrary code on a system with full privileges. WebDAV could be utilized to exploit this vulnerability because it called ntdll.dll to process incoming requests.

Likelihood of spoof

It is very unlikely that this address has been spoofed. Since this is a TCP attack, the TCP three-way handshake would have already occurred before these alerts could be triggered. It also appears that the client is interested in a response from the web server. Additionally, there are no similar alerts at this time which would have benefited from this type of misdirection.

Evidence of active targeting

The main web servers were not attacked in this attempt, but rather a number of systems which are probably student machines. Additionally, there is no evidence of a scan, and it does not appear that this was a random attack. This definitely has the look of active targeting.

Correlations

Tim Kroeger listed a similar attack triggered by a different NOOP rule in his GCIA practical, and included packet payload which showed the attack in greater detail than is possible here.
Another similar attack was posted to the Snort discussion board, where it was suggested that the attack was perpetrated by the Welchia worm, which exploits WebDAV:
http://archives.neohapsis.com/archives/snort/2004-03/0347.html

More information on the vulnerability can be found in the Microsoft security bulletin for this vulnerability:
http://www.microsoft.com/technet/security/bulletin/MS03-007.mspx

A much more detailed analysis is available from the KLC Consulting Company's website:
http://www.klcconsulting.net/articles/webdav/webdav_vuln.htm

The original CERT advisory:
http://www.cert.org/advisories/CA-2003-09.html

Technical Recommendation

Many of these users may be unaware that they are even running a web server. Users should be located and informed that their systems are vulnerable. Additionally, a weekly email bulletin should go out to all students listing current patches and linking to easy installs of all critical patches.

Severity

Criticality: 3
These systems do not appear to be the main web servers on the university network. They are most likely student systems with IIS on them. However, due to the fact that several systems are targeted in this attack, a slightly higher rating is assigned to these systems collectively.

Lethality: 5
This attack can allow attackers to run arbitrary code on the exploited system, allowing more advanced attacks against internal targets which may have been otherwise shielded.

System Countermeasures: 2
These are not production systems, thus are probably not patched regularly. Additionally, while there was exploit code for this vulnerability available shortly after it became public, no major worms seem to have exploited this vulnerability for several months after the fact, making it much less likely that these systems would have been patched.

Network Countermeasures: 2
It is unlikely that web services are being filtered on a University network. Indeed, it appears that all communication attempts were successful for this attack. However, the there was a rule on the IDS which caught the attack, so the rating is slightly higher than it would be otherwise.

Total Severity Rating: (3 + 5) – (2 + 2) = 4

Detect #2 SMB scans and subsequent connection to C$ share.

There were several detects wherein a large number of systems were scanned for open SMB access and triggered the "SMB Name Wildcard" rule, and several were

subsequently accessed on their C$ share, triggering the "SMB C access" rule:

```
06/06-07:33:44.408322  [**] SMB Name Wildcard [**] 62.210.220.109:28216 -> MY.NET.190.101:137
06/06-07:33:44.787388  [**] SMB Name Wildcard [**] 62.210.220.109:28216 -> MY.NET.190.103:137
06/06-07:33:44.873064  [**] SMB Name Wildcard [**] 62.210.220.109:28216 -> MY.NET.190.104:137
06/06-07:33:44.959604  [**] SMB Name Wildcard [**] 62.210.220.109:28216 -> MY.NET.190.105:137
06/06-07:33:45.280885  [**] SMB Name Wildcard [**] 62.210.220.109:28216 -> MY.NET.190.106:137
06/06-07:33:45.304355  [**] SMB C access [**] 62.210.220.109:28607 -> MY.NET.190.93:139
06/06-07:33:45.502498  [**] SMB C access [**] 62.210.220.109:28610 -> MY.NET.190.94:139
06/06-07:33:45.541476  [**] SMB Name Wildcard [**] 62.210.220.109:28216 -> MY.NET.190.107:137
06/06-07:33:47.125575  [**] SMB Name Wildcard [**] 62.210.220.109:28216 -> MY.NET.190.109:137
06/06-07:33:47.266571  [**] SMB Name Wildcard [**] 62.210.220.109:28216 -> MY.NET.190.110:137
06/06-07:33:47.566914  [**] SMB Name Wildcard [**] 62.210.220.109:28216 -> MY.NET.190.111:137
06/06-07:33:47.715770  [**] SMB Name Wildcard [**] 62.210.220.109:28216 -> MY.NET.190.112:137
06/06-07:33:47.793994  [**] SMB Name Wildcard [**] 62.210.220.109:28216 -> MY.NET.190.113:137
06/06-07:33:47.853853  [**] SMB C access [**] 62.210.220.109:28616 -> MY.NET.190.102:139
06/06-07:33:47.864387  [**] SMB C access [**] 62.210.220.109:28615 -> MY.NET.190.100:139
06/06-07:33:47.901064  [**] SMB Name Wildcard [**] 62.210.220.109:28216 -> MY.NET.190.114:137
06/06-07:33:48.212196  [**] SMB Name Wildcard [**] 62.210.220.109:28216 -> MY.NET.190.116:137
```

SMB Name Wildcard attempts were the most common type of alert over the course of the entire 3 days. The rule looks for service enumeration requests to the NetBIOS Name Service on port 137. A response to such a request would return information such as workstation name, domain name, currently logged on user, and file/print services.

I only found one access to the C$ share per client, which I found odd. According to a message in the Neohapsis archives from Dennis Ducamp (http://archives.neohapsis.com/archives/snort/2000-05/0267.html), this rule is only triggered when the system being attacked is a Win9x system. So I opened up Ethereal and attempted to connect to the C$ share on an old Win98 system on my network. When I examined the packets, my system had to do a negotiate protocol request first, before attempting to connect to any specific share, and then the string which would trigger this rule only appeared to be defined in the "SMB Tree Connect" packet, which would explain why the detect only shows up once per connection attempt. Additionally, since connection attempts to Windows 2000 and XP systems do not trigger this rule, it is possible that more systems were affected.

Reason this attack was selected

I selected this detect because it shows systems being probed for specific information, and vulnerable systems being exploited immediately.

Detect Generated by

The detect was generated by the university's Snort IDS.
The rule triggering the "SMB Name Wildcard" detect was:

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

While Windows systems will make a NetBIOS connection attempt using UDP 137 on both client and host, a scanning tool will not necessarily do so. This rule recognizes that, and will trigger on any UDP request to port 137. This rule is triggered when an attempt is made to discover NetBIOS resources based on IP address, similar to performing NBTSTAT –A from a Windows command line. This rule is listed in the arachNIDS IDS database as "IDS177/netbios-name-query".

The second rule, triggering the "SMB C$ Access" alert is listed here:

```
alert tcp any any -> $HOME_NET 139 (msg:"SMB C$ access";
content:"\C$|00 41 3a 00|";)
```

Any attempt from an external system to connect to the C$ chare with a connection to TCP 139 would be caught by this rule. However, Windows 2000/XP systems with NetBIOS disabled will use TCP 445 instead of 139 for direct SMB connections, so the rule is somewhat dated.

Attack Mechanism

The attackers are looking for open shares to the root of the primary drive on older Windows systems, most likely for storage purposes, although open access of this type would also allow access to password files, and access to modify the Windows registry and system configuration files, as well.

This is almost certainly a script. The source port is 28216 for all SMB enumeration requests, and a normal request from a Windows system would come from UDP 137. However, SAMBA's NMBLOOKUP doesn't use UDP 137 to do NetBIOS lookups, so it is possible that the script is utilizing this tool during to do the lookups. Since the NetBIOS name query commonly reveals the currently logged on user, it would be trivial to pipe that information into another script attempting to use the user name gathered in combination with a null or list of simple passwords.

Likelihood of spoof

Based on packet captures done on my internal network, it appears that for the C$ Access rule set to be triggered, the system must first go through an SMB protocol negotiation, so it is not likely that these are spoofed. While these could be caused by a tool indiscriminately sending these packets to systems, it is not likely that any purpose would be served by that action.
It is unlikely that the SMB Wildcard alerts are decoy scans, since the first attack is used specifically to find out information on Windows systems, and if they were decoys, the UDP requests should be coming from spoofed hosts and be more spread out than what is seen here. What we see are a large number of NetBIOS name enumeration requests, followed immediately by a c$ connection attempt by the same address.

Evidence of Active Targeting

It doesn't appear that the initial scan is actively targeting any systems, but the subsequent connection attempt definitely seems to making successful connections to systems with open C$ shares, showing the information gathered being immediately put to use.

Correlations

This is a very common attack on the internet. The NetBIOS scans are listed in arachNIDS database here: http://www.whitehats.com/info/IDS177
The C$ access attempt is listed here: http://www.whitehats.com/info/IDS339

Derek Buelna listed a similar attack in his GCIA practical, as did Al Maslowski-Yerges.

Technical Recommendation

Using Windows file sharing on the internet is a very bad idea, and students should not be doing so. Incoming requests to these ports should be filtered at the firewall.

Severity

Criticality: 2
The systems look like student desktop machines, so the criticality is not very high. However, a large number of systems are being targeted, and several are even successfully exploited.

Lethality: 3
Accessing the C$ is probably not intended to bring the systems down. More likely the attacker's intent is to use the systems as file storage. However, most old Win9x systems would have PWL files on the hard drive with weak encryption, making it fairly easy for an attacker to steal user passwords for use on other systems.

System Countermeasures: 2
Most Windows systems do not have their root directory open to the entire world. Systems would have to be configured explicitly to allow such access, whether intentionally or in error. However, the weakest link here is probably user passwords, since the NetBIOS name query would have allowed the attacker to gather user names from many of these systems, and quite often the user password is either blank or a very simple password.

Network Countermeasures: 1
Although university networks are typically very open, there is no reason any network needs to allow SMB traffic outside its network.

Total Severity Rating: (2 + 3) – (2 + 1) = 2

Detect #3 – DDoS against remote system

Over 1800 alerts from 212 source addresses containing the destination address of 67.80.77.94 port 6112 triggered the "TCP SRC and DST outside network" rule, spanning from about 6:30 PM on June 6th till a little after 7 AM on June 8th:

```
06/06-18:53:01.540286  [**] TCP SRC and DST outside network [**] 204.60.43.157:1074 ->
67.80.77.94:6112
06/06-18:53:01.923534  [**] TCP SRC and DST outside network [**] 24.203.111.240:1265 ->
67.80.77.94:6112
06/06-18:53:03.028166  [**] TCP SRC and DST outside network [**] 129.44.174.29:1359 ->
67.80.77.94:6112
06/06-18:53:15.263481  [**] TCP SRC and DST outside network [**] 129.44.174.29:1359 ->
67.80.77.94:6112
06/06-18:53:17.537532  [**] TCP SRC and DST outside network [**] 66.168.100.82:3018 ->
67.80.77.94:6112
06/06-18:53:23.862447  [**] TCP SRC and DST outside network [**] 24.214.132.132:33580 ->
67.80.77.94:6112
06/06-18:53:24.582613  [**] TCP SRC and DST outside network [**] 65.92.7.135:3252 ->
67.80.77.94:6112
06/06-18:53:27.785611  [**] TCP SRC and DST outside network [**] 68.32.167.90:1601 ->
67.80.77.94:6112
06/06-18:53:33.484384  [**] TCP SRC and DST outside network [**] 65.92.7.135:3252 ->
67.80.77.94:6112
06/06-18:53:37.951886  [**] TCP SRC and DST outside network [**] 213.39.228.59:1436 ->
67.80.77.94:6112
06/06-18:53:43.267529  [**] TCP SRC and DST outside network [**] 68.56.240.5:65156 ->
67.80.77.94:6112
06/06-18:53:49.257070  [**] TCP SRC and DST outside network [**] 68.56.240.5:65156 ->
67.80.77.94:6112
06/06-18:53:54.876560  [**] TCP SRC and DST outside network [**] 24.214.132.132:32776 ->
67.80.77.94:6112
```

The destination address resolves to a dynamically-assigned IP address on Optimum Online's cable broadband service. The port being targeted is commonly used for the Common Desktop Environment (CDE) desktop subprocess control service (dtspc). There have been several advisories for attacks against dtspc over the years, but this doesn't look like an attack intended to exploit a vulnerability on the system. Blizzard's BattleNet servers also require access to TCP 6112 on client systems, which could be why this port is open on what looks to be a home system. The signature looks like a standard SYN attack against a known-open port on this box. This level of traffic would not be enough to SYN-flood any newer systems, indicating that a system on the university's network is probably a zombie in a DDoS network. SYN-flooding is an old attack, and there is lots of information available on it, such as CERT's original advisory CA-1996-21.
Although this is an internal system attacking a remote system, it is clear that at least one internal system has been compromised, and may be acting as more than just a DDoS zombie. It is important to catch and clean up systems like these internally.

Reason this attack was selected

I chose this attack because it shows an example of an internal system attacking an external system, and demonstrates why egress filtering is important.

Detect Generated By

This detect was generated by the university's Snort IDS.
The rule triggering this alert is not found at Snort.org. Additionally, the only mention found of this alert through google were references to GCIA practical papers, meaning that it was probably a custom rule created by the staff. However, considering that all packets flagged with this alert have external source and destination packets, and given the name of the rule, it's fairly clear what it is supposed to find.
It is not likely that this is a mis-configured device, since in that case the source address would most likely use a private address such as 192.168.x.x or 169.254.x.x. In this case, the destination address remains the same, and the source address changes, and all are valid public addresses. If it were simply a mis-configured device, there should also be more retransmission attempts than are seen here. While most of the source IPs are used multiple times, and in some cases the source port is the same, there isn't any clear spaced retransmission of the packets, as would be expected of valid behavior from a mis-configured system.

Attack Mechanism

There is an agent on at least one system on the university network, causing it to send bogus packets to another host in an attempt to cause a denial of service. The agent was probably installed either when the system was compromised by an attack which allowed the attacker to run arbitrary code, or it was installed by the user, either by being tricked into opening an executable file, or by installing an application which included a Trojan. Infected applications are very common on P2P networks, and often license "cracks" are executables which surreptitiously install a Trojan when it is run to retrieve the license key for a given product.

Likelihood of spoof

These are definitely spoofs, since it would not be possible for these packets to be generated normally by an internal system. It is very unlikely that this could be a mis-configured system, since all packets are going to the same destination but have different source IPs.

Evidence of active targeting

An internal system has been compromised, and the target system is definitely being actively targeted, although it is unclear if this is part of a larger attack, and there was no other traffic found in any of the logs to 67.80.77.94 during this three day period.

Correlations

This is a common type of attack. Steve Gibson describes a similar attack from a few years ago in detail on his website:
http://www.grc.com/dos/drdos.htm

Technical Recommendation

Spoofed packets coming from the internal network should not be tolerated, and should be filtered at the firewall.

There was not enough information in the logs to determine where the spoofed packets originated, however if the packets could be caught in real or near-real time, it should be possible to trace the original source. Additionally, many DDoS networks use IRC to manage their agents, so closer analysis of IRC-related alerts may be beneficial.

Severity

Since the attack is against an external system, I will base the severity on the compromised system internally.

Criticality: 2
It is unclear which system has been compromised, but it is most likely a student system, since the compromise of a production system would usually be caught rather quickly.

Lethality: 4
The system has been compromised. It is uncertain as to what other functions the attacker may utilize on this system. Some zombies utilize very basic clients, while some use clients with functionality more akin to Netbus or BackOrifice.

System countermeasures: 1
The system has been compromised. Clearly either it wasn't being patched, or user installed an app with some malware included. Any system countermeasures are negligible.

Network countermeasures: 2
It doesn't appear that the network countermeasures were very good in warding off the initial compromise, and it doesn't appear to be blocking packets with source IP's outside of its subnet range, either. This would a good thing to do, if the university wanted to be a good neighbor.

Total Severity Rating
(2 + 4) – (1 + 2) = 3

Network Statistics

Alert Logs

Top Talkers:

Source IP	Total
216.39.48.2	12226
204.71.20.68	10186
195.101.253.232	8800
209.52.45.162	1882
150.214.16.43	1759
66.77.73.236	1748
68.155.6.153	1262
80.181.67.217	1195
218.28.149.18	1170
80.143.107.143	1162

The top talker in the alert logs, 216.39.48.2 resolves to an AltaVista netblock, and the alerts are exclusively "CS Webserver External Access" and "MY.NET.30 Access" alerts to web servers. It looks like harmless indexing traffic from a search engine. The second top talker, 204.71.20.68, resolves to www.valuengine.com. According to the WHOIS database, the domain name has been registered to this IP since 2002. The traffic all occurs within the span of a few minutes between 4:52 and 4:55 AM on June 7. It is unclear what this traffic really is, although it could be a decoy scan from a spoofed address. It is unlikely to be an actual scan, unless this system had been compromised. A search for information on valuengine.com scanning systems or having been compromised did not provide any relevant results.

Most Common Destination Ports:

Destination Port	Total
137	395026
80	41847
111	10187
65535	2859
6112	1831

None of these is surprising. Port 137 is a very commonly scanned port, since older Windows systems listen to NetBIOS name requests on this port, and much information can be gleaned from this service. Web servers are also a common target, which explains why port 80 is on the list. As explained in the determination section, many of these were also false positives. The alerts for Port 111 were a basic scan from 204.71.20.68, as mentioned above. Although 65535 can be associated with Code Red and its ilk, most of the traffic here also appeared to be valid requests to common services from internal systems. Traffic to port 6112 was detailed in Detect #3.

OOS Logs

Top Talkers:

Source IP	Total
66.117.30.14	5304
193.219.55.20	793
213.186.35.9	271
216.95.201.25	267
216.95.201.24	247

The two IPs at the bottom resolve to SMTP servers at dbhits.com, which is listed as a spamming domain on several spam lists. The traffic appears to be going to valid mail servers, and no fingerprinting or open relay probing appears to be taking place. Most of the rest of the entries appeared to be normal traffic with the ECN bits set.

Most Common Destination Ports:

DestPort	Total
25	5585
1182	5305
4662	1905
80	1241
6346	352

Packets to Port 25 were triggered in every case by the ECN flags, but an analysis of the logs shows that almost all of these packets were traffic from UUNET SMTP servers on the 216.95.201.x subnet. TCP 1182 is commonly used for HTTP proxying, but in this case, all traffic came from 66.117.30.14, with the destination of either MY.NET.224.134 or MY.NET.233.78. However, it doesn't appear that these systems ever responded, so it is doubtful that they are listening on TCP 1182. Port 4662 and 6346 are used for P2P apps Edonkey and Gnutella, respectively, and this traffic did not appear to be malicious.

Scan Logs

Top Talkers:

SrcIP	Total
MY.NET.1.3	330156
MY.NET.1.4	179472
MY.NET.87.80	107763
MY.NET.153.223	94898
MY.NET.219.42	52139
203.218.207.9	50596
62.97.164.221	50514
211.22.185.132	47642
202.178.162.87	46124
198.199.227.40	43273

In this case, the first five are internal systems, several of which are university servers, the

top two being the DNS servers. As mentioned in the network setup section, these scan alerts were false positives based on valid traffic.

Most Common Destination Ports:

DestPort	Total
445	1065214
53	565738
80	247397
6257	170965
139	137855

Four of the top 5 are expected, but UDP 6257 was not. A search found that UDP 6257 is one of the common ports associated with WinMX, a P2P application. Almost all alerts came from 11 internal systems, and while school policy may not allow P2P apps, this is probably not malicious traffic.

Suspicious IPs

System#1

The IP address 211.189.231.169 scanned ports 12345 and 27374 on over 400 university systems over the course of an afternoon. The OOSlogs were the only ones with enough information in them to guess at OS type, but this system was only found in the scan logs. The following registration was taken from samspade.org:

Server Used: [whois.krnic.net]

211.189.231.169 = [ ] (www.nic.or.kr) Whois
query: 211.189.231.169
ENGLISH
KRNIC is not a ISP but a National Internet Registry similar to APNIC.
The followings are information of the organization that is using the IPv4 address.
IPv4 Address : 211.189.231.0-211.189.231.255
Network Name : QRIXNET-INFRA
Connect ISP Name : QRIXNET
Connect Date : 20030304
Registration Date : 20031020
[Organization Information]
Organization ID : ORG84950
Org Name : QrixNetworks
State : SEOUL
Address : 47-47 Suyu 5(o)-dong Gangbuk-gu
Zip Code : 142-075
[Admin Contact Information]

Name : Kang kwang suk
Org Name : QrixNetworks
State : SEOUL
Address : 47-47 Suyu 5(o)-dong Gangbuk-gu
Zip Code : 142-075
Phone : 82-2-999-5975
Fax : 82-2-996-5975
E-Mail : mgr@qrix.com
[Technical Contact Information]
Name : Lee keun kyu
Org Name : QrixNetworks
State : SEOUL
Address : 47-47 Suyu 5(o)-dong Gangbuk-gu
Zip Code : 142-075
Phone : 82-2-999-5975
Fax : 82-2-996-5975
E-Mail : ip@qrix.com

System #2
On June 7, a system with IP 61.230.44.32 sent a number of mal-formed packets to 14 different internal systems. These were packets with no payload sent to specific well-known ports or port 0, most of which had invalid flag combinations. The packets were most likely intended for OS fingerprinting against these systems.

Server Used: [whois.twnic.net]

61.230.44.32 = [61-230-44-32.dynamic.hinet.net] Chunghwa Telecom Data communication Business Group
No.21 Hsin-Yi Rd. sec. 1
Taipei
TW
Netname: HINET-NET
Netblock: 61.228.0.0/14
Administrator contact:
Chung Yung Kang (CYK-TW) cykang@ms1.hinet.net
886-2-2322-3442
Technical contact:
Chung Yung Kang (CYK-TW) cykang@ms1.hinet.net

System #3
On June 6, a system with IP 210.83.159.228 did a basic SMB Name Wildcard request to MY.NET.189.23. On Jun 7 and June 8, the system attempted several attacks against a number of ports, including TCP 3269, commonly used for the MS Global Catalog server. There is not enough information to determine whether the system being attacked is a GC or not.

Server Used: [whois.apnic.net]

210.83.159.228 = [ ]
inetnum: 210.83.159.0 - 210.83.159.255
netname: people-hospital
country: cn
descr: moganshan road hangzhou city zhejiang province
admin-c: TC254-AP
tech-c: TC254-AP
status: ASSIGNED NON-PORTABLE
changed: moujh@china-netcom.com 20021023
mnt-by: MAINT-CN-ZM28
source: APNIC
person: TECH GROUP CNC
address: 9/F Building A Corporate Square No. 35 Financial Street
address: Xicheng District Beijing 100032 P.R.China
country: CN
phone: 86-10-88093588
fax-no: 86-10-88091442
e-mail: tech-group@china-netcom.com
nic-hdl: TC254-AP
mnt-by: MAINT-CN-ZM28
changed: zhaomq@china-netcom.com 20010917
source: APNIC

Analysis Methodology

All analysis was performed on a Dell 600MHZ laptop with 256 MB RAM, running Windows XP Pro.
Snort logs were cleaned up and converted to comma-delimited format using basic PERL scripts. I used a basic PERL script template from Bill Philips' GCIA practical, and modified it for my own purposes. Examples are available in the appendix.
ActiveState ActivePERL v5.8 for Windows was the PERL interpreter used.
Since SQL server was unavailable to me at the time, I used MS Access 2003, and analyzed the data using SQL queries. Examples of queries used are available in the appendix.
UltraEdit 32 was used for some simple analysis, and to view the raw logs. Its Regular Expression support was a tremendous help on this project.

For regex analysis, I initially used Qgrep, a grep-like tool available in the Windows Server 2003 Resource Kit. However, I found that UltraEdit work just as well, and made it much simpler to do sub-matches. I also used UltraEdit to view the raw logs, because Notepad tended to choke on the larger files, and didn't recognize carriage returns in the logs.

I decided to place all the data in a database for easier mining, since some of the logs contained millions of entries. I had planned on correlating different log types using standard join queries, but found that, with the limited system I was using, even simple queries took several minutes to process. Any query involving multiple tables or sub-queries caused a dramatic increase in processing time, up to 15 minutes in some cases. Due to the large number of queries performed on this project, it was more economical in most instances to perform basic queries and do additional correlation manually.

I also concatenated all logs of the same type before importing them as tables in the Access database. This made analysis simpler. I found that I had to include headers on the CSV files I created from the logs when I imported them into a table, otherwise some fields didn't get created.

Appendix

PERL Scripts

I used a template from Bill Phillips' GCIA practical. The example here was used to comma-delimit the OOS logs. Other scripts were similar.

OOScsv.pl

```
#!/usr/bin/perl
#
#   Start mainline code
while (<>) {
#
#  Check for blank line, if so process next line
#
   if ( $_ eq "" )  { next };
#
        if ( $_ =~ m/(^\d+\/\d+)\-([\d\:]+)\.\d+\s+([\d\w\.]+)\:(\d+)\s+\-
\>\s+([\d\w\.]+)\:(\d+)/)
        {
        $date  =        $1;
        $time  =        $2;
        $saddr =        $3;
        $sport  =       $4;
        $daddr =        $5;
        $dport =        $6;
        }
        if ( $_ =~
m/^TCP\sTTL\:(\d+)\sTOS\:([\d\w]+)\sID\:(\d+)\sIpLen\:(\d+)\sDgmLen\:(\d+)\s([\s\w]{2
```

```
})/)
        {
                $TTL    =       $1;
                $TOS    =       $2;
                $ID     =       $3;
                $IpLen  =       $4;
                $DgmLen         =       $5;
                $DF     =       $6;
                #print "$1, $2, $3, $4, $5, $6\n"
        }

        if ($_ =~
m/(^[\d\w\*]+)\s+Seq\:\s+([\d\w]+)\s+Ack\:\s([\d\w]+)\s+Win\:\s+([\d\w]+)\s+TcpLen\:\s
([\d\w]+)/)
        {
                $flags  =       $1;
                $Seq    =       $2;
                $Ack    =       $3;
                $Win    =       $4;
                $TcpLen         =       $5;
                #print "$1, $2, $3, $4, $5\n"
                print "$date/03 $time, $saddr, $sport, $daddr, $dport, $TTL, $TOS, $ID,
$IpLen, $DgmLen, $DF, $flags, $Seq, $Ack, $Win, $TcpLen\n"
        }

        #print "$date/03 $time, $saddr, $sport, $daddr, $dport, $TTL, $TOS\n"

} #end while loop
```

SQL Queries

Queries used were relatively basic. Most just matched on specific strings or "like" statements. A few examples are attached.

Distinct Alerts

The following query displays a total count of each specific alert found in the alert log table:

```
SELECT [AlertType], Count([AlertType]) AS total
FROM AlertLog
GROUP BY [AlertType];
```

Traffic matching by subnet

The following SQL query looked for any traffic in the scan log coming from or going to the 131.118.254.x subnet:

```
SELECT *
FROM ScanLog
WHERE ((([SrcIP]) Like '131.118.254' & "*") AND (([DestIP]) Like '131.118.254' & "*"));
```

Match by subnet and alert

This query looks at any traffic going to the MY.NET.90.x subnet with any type of SMB alert.

```
SELECT *
FROM AlertLog
WHERE ((([DestIP]) Like 'MY.NET.190.' & "*") AND (([AlertType]) Like 'SMB' & "*"));
```