Global Information Assurance Certification Paper

SANS Security Parliament Hill 2000
GIAC Intrusion Detection Practical - Bill Scherr

Contents:

Introduction:

The core material of this paper comes from real-world networks that the author is (was) responsible for securing. The data was taken from various versions of TCPDUMP, the GNU packet sniffer command line user interface. A more detailed explaination of the process is part of this assignment, and will be described below.

The traces are adjusted to protect the guilty (and innocent). My.Net.Here.yyy represents addresses of the monitored network. Them.Net.aaa.bbb represents the addresses in the incoming packets.

An example for your benefit:
06:19:50.280000 My.Net.Here.138.25 > Them.Net.134.132.4509: P 140:183(43) ack 92 win 8701 (DF)

Above: an ack reply in an ongoing mail transfer. NOT AN ATTACK!

Any pronouns are used for gramatical flow. No religion, employment, gender, or other motivating factor should be implied from the use of one or another personal pronoun. That being said...

Network Detects Analyzed:

top

The following detects are formatted, per SANS guidance and standards, as follows:

1. Source of Trace (generic description of network)
2. Tool or technique detect was generated by
3. Spoof Probability
4. Description of attack
5. Attack Mechanism
6. Correlations
7. Evidence of Active Targeting

8. Severity (according to given formula)
9. Defensive Recommendation
10. Multiple choice test question

Network Detect #1

1. NetBIOS session attack.(Posted to SANS 09/11/00) Nailed by Ken Armstrong

top

*The data (explaination of fields follows the last packet)

Aug 26 00:27:06 my.net.here.129 ASCEND: wan1 tcp my.net.here.128;139 <- them.net.181.58;1058 48 syn !pass (totcp-1)
Aug 26 00:27:08 my.net.here.129 ASCEND: wan1 tcp my.net.here.128;139 <- them.net.181.58;1058 48 syn !pass (tcp-1)
Aug 26 00:27:13 my.net.here.129 ASCEND: wan1 tcp my.net.here.129;139 <- them.net.181.58;1059 48 syn !pass (totcp-1)
Aug 26 00:27:14 my.net.here.129 ASCEND: wan1 tcp my.net.here.128;139 <- them.net.181.58;1058 48 syn !pass (tcp-1)
Aug 26 00:27:15 my.net.here.129 ASCEND: wan1 tcp my.net.here.129;139 <- them.net.181.58;1059 48 syn !pass (tcp-1)
Aug 26 00:27:21 my.net.here.129 ASCEND: wan1 tcp my.net.here.129;139 <- them.net.181.58;1059 48 syn !pass (tcp-1)
Aug 26 00:27:26 my.net.here.129 ASCEND: wan1 tcp my.net.here.128;139 <- them.net.181.58;1058 48 syn !pass (tcp-1)
Aug 26 00:27:27 my.net.here.129 ASCEND: wan1 tcp my.net.here.131;139 <- them.net.181.58;1061 48 syn !pass (totcp-1)
Aug 26 00:27:29 my.net.here.129 ASCEND: wan1 tcp my.net.here.131;139 <- them.net.181.58;1061 48 syn !pass (tcp-1)
Aug 26 00:27:33 my.net.here.129 ASCEND: wan1 tcp my.net.here.129;139 <- them.net.181.58;1059 48 syn !pass (tcp-1)
Aug 26 00:27:34 my.net.here.129 ASCEND: wan1 tcp my.net.here.132;139 <- them.net.181.58;1062 48 syn !pass (totcp-1)
Aug 26 00:27:35 my.net.here.129 ASCEND: wan1 tcp my.net.here.131;139 <- them.net.181.58;1061 48 syn !pass (tcp-1)
Aug 26 00:27:36 my.net.here.129 ASCEND: wan1 tcp my.net.here.132;139 <- them.net.181.58;1062 48 syn !pass (tcp-1)
Aug 26 00:27:41 my.net.here.129 ASCEND: wan1 tcp my.net.here.133;139 <- them.net.181.58;1063 48 syn !pass (totcp-1)
Aug 26 00:27:42 my.net.here.129 ASCEND: wan1 tcp my.net.here.132;139 <- them.net.181.58;1062 48 syn !pass (tcp-1)
Aug 26 00:27:43 my.net.here.129 ASCEND: wan1 tcp my.net.here.133;139 <- them.net.181.58;1063 48 syn !pass (tcp-1)
Aug 26 00:27:47 my.net.here.129 ASCEND: wan1 tcp my.net.here.131;139 <- them.net.181.58;1061 48 syn !pass (tcp-1)
Aug 26 00:27:48 my.net.here.129 ASCEND: wan1 tcp my.net.here.134;139 <- them.net.181.58;1064 48 syn !pass (totcp-1)

Aug 26 00:27:49 my.net.here.129 ASCEND: wan1 tcp my.net.here.133;139 <- them.net.181.58;1063 48 syn !pass (tcp-1)
Aug 26 00:27:50 my.net.here.129 ASCEND: wan1 tcp my.net.here.134;139 <- them.net.181.58;1064 48 syn !pass (tcp-1)
Aug 26 00:27:54 my.net.here.129 ASCEND: wan1 tcp my.net.here.132;139 <- them.net.181.58;1062 48 syn !pass (tcp-1)
Aug 26 00:27:55 my.net.here.129 ASCEND: wan1 tcp my.net.here.135;139 <- them.net.181.58;1065 48 syn !pass (totcp-1)
Aug 26 00:27:56 my.net.here.129 ASCEND: wan1 tcp my.net.here.134;139 <- them.net.181.58;1064 48 syn !pass (tcp-1)
Aug 26 00:27:57 my.net.here.129 ASCEND: wan1 tcp my.net.here.135;139 <- them.net.181.58;1065 48 syn !pass (tcp-1)
Aug 26 00:28:01 my.net.here.129 ASCEND: wan1 tcp my.net.here.133;139 <- them.net.181.58;1063 48 syn !pass (tcp-1)
Aug 26 00:28:03 my.net.here.129 ASCEND: wan1 tcp my.net.here.135;139 <- them.net.181.58;1065 48 syn !pass (tcp-1)
Aug 26 00:28:08 my.net.here.129 ASCEND: wan1 tcp my.net.here.134;139 <- them.net.181.58;1064 48 syn !pass (tcp-1)
Aug 26 00:28:15 my.net.here.129 ASCEND: wan1 tcp my.net.here.135;139 <- them.net.181.58;1065 48 syn !pass (tcp-1)

Explaination of Fields: Ascend Pipeline 130(v.35) Firewall Logs

From Left to Right.

Aug 26 00:28:15	- Date Time stamp in seconds
my.net.here.129 ASCEND:	- Router generating log
wan1	- Reporting Interface
tcp	- Packet Protocol
my.net.here.135;139	- Internal Address**;**Port Number (note the semi-colon)
"<-"	- Direction of Packet (incoming, in this case)
them.net.181.58;1065	- External Address**;**Port Number (note the semi-colon, again)
48	- size of packet
syn	- SYN Flag was set (all flags listed by three letter representation)
!pass (tcp-1)	- Firewall Activation Rule (No bearing on GUI manipulation tool - packet not passed)

1.1 Source of the Detect:
This was captured on the border of a production network that was designed and is maintained by the author.

1.2 Tool or technique detect was generated by:
The detecting mechanism was system log on an internal machine. A separate file is generated for this router using a syslog daemon.

1.3 Possibility of spoofing:
Not Likely! TCP attacks and reconnaissance require a return packet. A sniffer placed strategically on the spoofed network would allow this address to be spoofed for that purpose. DoS attacks do not necessarily require a return packet, allowing spoofing. This is very loud scan which may indicate a machine initiated

scan (Chode-911 virus). There is so much that can be done to port 139 that it makes simple syn packets like these VERY dangerous. The fact that these packets are on the external interface is interesting in its own right.

1.4 Description of Attack:
A syn packet to port 139 (netbios session) repeated four times to every host. The attack lasted just over a minute. This scan also originated from the same class-b network, which is obfuscated by sanitation. Normally these packets occur after a netbios-ns (port 137) name resolve exchange with a master browser. No such exchange was attempted. Much functionality is available with a connection to port 139 (i.e. DOS Prompt file manipulation).

1.5 Attack Mechanism:

- I did put my quarter on the Chode-911 batch worm. This worm will search a subnet for open windows shares and copy itself to those shares. See the analysis from Sophos. Worms generally require no user action to activate on a particular host.
  This particular worm requires a reboot of the Windows computer (evaluate the likelihood of that yourselves). The short timespan, repetitive host occurance (4x each) and low sending port sequence increment support this analysis. This is the antithesis of a stealth scan.
- A DoS CVE candidate is listed as CAN-2000-0580. This DoS increases CPU utilization of a Windows 2000 "server" significantly by sending a continuous stream of binary zeros to various UDP and TCP ports. **As of this writing** this has not been confirmed to work on TCP port 139. The attack also consists of relatively small packets. **This reported attack does not match this CVE candidate.**
- Misconfigured and unprotected windows 9[5,8] computers will grant a connection to requests regardless of authorization. This contributed to the success of the Chode-911 worm mentioned above. Port 139 is the data transfer port in windows networking. An excellent summary of the risks involved is the IDFAQ v1.35.These packets highlight the fact that internet security needs the involvement of all system owners.

The above list is not all inclusive. The detail offered by the Ascend firewall leaves a lot to be desired. The 48 byte syn packet size is odd. It probably means that some tcp option was employed. Netbios-ssn is the heart of windows networking. If a connection had been established, the hardrive could have been erased, or worse. The Ascend router does not forward any netbios packets. Logistics at the time of deployment determined that a safe network could not otherwise be achieved.

1.6 Correlations:
This could be the work of a trojan of the "Chode" style. No further correlations were readily available, however. It is not apparent from the sanitation that this packet originated from the same class b address space. Many network centered autospreading worms will limit scans to the local class b or class c address space. It retrieves the local IP info as any other program would. It would not do to go hitting every address on the internet, and a class b scanner can be written smaller. This would tend to support the analysis of a machine initiated scan.
A packet similar to this was analyzed by Sean Dickens for the 6 April GIAC listing. That analysis included UDP port 137 packets, and no packets to that port were detected. Referencing Sean's Packet 63 shows that TCP options were used. However, that trace does not show which options were used. This is not a match for correlation purposes.
In the weeks since this scan was found, a deeper look was undertaken. The first 48 byte port 139 scan was logged on 23 August. A total of eight addresses were logged as sources. It repeated ten times with two addresses scanning twice. For the sake of brevity, the traces are omitted. Suffice it to say that the patterns are identical to the above. This tends to buttress the analysis that this is a virus in a poorly configured neighborhood.

UPDATE: The scans have continued. Upstream providers are aware of this. Several Networks in this area have seen this as well. Ken Armstrong took the initiative to capture this beast. It turns out to be the QAZ trojan. "This is the critter that replaces the notepad.exe file and then tries to reinfect other systems.Ken" It is a machine initiated scan.

1.7 Evidence of Active Targeting:
Not Apparent. This hit all addresses that could be sensed. It hit them each four times on a port that should not usually available to the internet (re: SANS Windows NT Security, Step-by-Step sec 6.3.1) . No information was received from any host inside the firewall. It is not likely that anything was noticed by the scanner. Ergo, it is not likely that any host was identified before or by this scan.

1.8 Severity:
(Target Criticality + Attack Lethality) - (System Countermeasures + Network Countermeasures).

(4 + 5) - (4 + 5) = 0

1. **Target Criticality:** One of the targets was an Exchange SMTP server. This server handles all e-mail delivery to and from the network. (4)
2. **Attack Lethality:** This scan was looking for open windows shares. If C: is open on an NT system and the guest account is enabled, the SAM file can be had! The Chode Virus erases the contents of the drive, then calls 911 if a modem is attached. (5)
3. **System Countermeasures:** The exchange server does not participate in the rest of the network! It is patched to 6a with the C2 hotfix. All of the registry settings from the C2 configuration, except RPC, are set. A current virus checker is running on the NT server. The exchange software is not the latest version. No other windows machines exist in real address space. (4)
4. **Network Countermeasures:** Shadow is running on the inside of the border firewall. The packets did not penetrate the firewall. External modems exist, but are detached when not in use. Dial-up networking is limited to software vendors internal networks. (5)

Don't let the Zero fool you! Netbios packets have no business on the internet. Netbios session packets can indicate a severly compromised system. The chode virus, if allowed to complete it's business, would get you the attention of the local constabulary. In the interests of internet society, the owning admin, or the upstream provider should be made aware.

1.9 Defensive Recommendation:
The defenses are set to deflect this attack because of the inherent exposures of the Windows operating system. NetBIOS is not allowed on the open internet! Stay the course.

1.10 Multiple Choice Question:
Port 139 is associated with:
a) Windows Netbios share services
b) Windows Netbios session services ***
c) Its windows, so no particular cause for concern
d) Solaris ingreslock services.

Network Detect #2

2. SunRPC Scan (aka PortMapper, ToolTalk, etc.)

top

/home/shadow/LOG/site/Sep05

20:21:52.030152 Them.Net.109.189.4944 > My.Net.Here.162.111: S 2174170510:2174170510(0) win 32120 <'mss 1460,sackOK,timestamp 465272020,nop,wscale 0> (DF) (ttl 38, id 57310)
20:21:52.037577 Them.Net.109.189.4945 > My.Net.Here.163.111: S 2183830373:2183830373(0) win 32120 <'mss 1460,sackOK,timestamp 465272020,nop,wscale 0> (DF) (ttl 38, id 57311)
20:21:52.136325 Them.Net.109.189.4952 > My.Net.Here.170.111: S 2176551491:2176551491(0) win 32120 (DF) (ttl 38, id 57318)
20:21:52.136679 My.Net.Here.170 > Them.Net.109.189: icmp: My.Net.Here.170 tcp port 111 unreachable [tos 0xc0] (ttl 255, id 44729)
20:21:52.317855 Them.Net.109.189.4972 > My.Net.Here.190.111: S 2185022683:2185022683(0) win 32120 (DF) (ttl 38, id 57338)
20:21:52.320443 My.Net.Here.190.111 > Them.Net.109.189.4972: R 0:0(0) ack 2185022684 win 0 (ttl 255, id 153)

Explaination of Fields: TCPDump packet capture output

/home/shadow/LOG/site/Sep05	- Date stamp and path (Added by Shadow!)
20:21:52.030152	- Time stamp (microseconds; 10^{-6})
Them.Net.109.189.4944 >	- Source Address.Port Number (note the period separating address.port; address on left is source)
My.Net.Here.162.111:	- Destination Address.Port Number (note the period, again)
S	- 13th byte, lower nibble **flags** (Syn, in this case)
2174170510:2174170510(0)	- IP sequence number synchronised (Zero Data bytes) / note the matching numbers, the difference of which is Zero. Relative numbers may be displayed after initial handshake (command line option).
win 32120	- Suggested TCP Window size, subject to negotiation
<'mss 1460,sackOK,timestamp 465272020,nop,wscale 0>	- Various TCP options. Greater/Less than signs are HTML tag delimiters; hense the quote mark
(DF)	- Don't Fragment IP Flag set
(ttl 38, id 57310)	- Time To Live (ip[8:1]), IP Identification (ip[4:2])
ack (RST packet in each day.)	- 13th byte, upper nibble **flags** (ack, in this case, else urg)
icmp:	- ICMP protocol (port unreach... in this case)

*There are many more fields in tcpdump, and an very powerful filtering language.

2.1 Source of the Detect:
A network the author participates in on a part time basis. This is an experimental network with Linux, Solaris, NetWare, Cisco, and Windows machines.

2.2 Tool or technique detect was generated by:
A shadow intrusion detection tool with "(tcp[13] &2 = 2)" in the tcp.filter file. Additional information was gathered by searching the dumps over three days.

2.3 Possibility of spoofing:
Not Likely, for the same reasons and with the same conditions as Item 1.3.

2.4 Description of attack:
This is our old friend, Mr. Sun RPCScan. Consisting of a simple syn scan reconnaissance to each open

address, it almost rules out spoofing. The responses to these packets will be analyzed for signs of active and available RPC (Remote Procedure Call) services. Several root compromises are known for this port.
The time break between the second and third packets is attributed to non-existing hosts. There is a 0.007 second space between packet arrivals, and the id numbers increment appropriately for a sequential scan. No host means no entry in the router's ARP table; ergo no packets.

2.5 Attack Mechanism:
The following table shows the extent of vulnerabilities associated with the Sun RPC Service. Little information is available to match packet contents against individual vulnerabilities. The standard TCPDump output also gives no clues to identifying the individual attack. The table is verbatim from cve.mitre.org.

Name	Description
CVE-1999-0008	Buffer overflow in NIS+, in Sun's rpc.nisd program
CVE-1999-0212	Solaris rpc.mountd generates error messages that allow a remote attacker to determine what files are on the server.
CVE-1999-0320	SunOS rpc.cmsd allows attackers to obtain root access by overwriting arbitrary files.
CVE-1999-0493	rpc.statd allows remote attackers to forward RPC calls to the local operating system via the SM_MON and SM_NOTIFY commands, which in turn could be used to remotely exploit other bugs such as in automountd.
CVE-1999-0687	The ToolTalk ttsession daemon uses weak RPC authentication, which allows a remote attacker to execute commands.
CVE-1999-0696	Buffer overflow in CDE Calendar Manager Service Daemon (rpc.cmsd)
CVE-1999-0974	Buffer overflow in Solaris snoop allows remote attackers to gain root privileges via GETQUOTA requests to the rpc.rquotad service.

Two responses are sent back to the scanning host, an ICMP port unreachable from 170, and a Reset from 190. The other two packets are dropped by the hosts. Since the service is not served to the internet, nor present on this segment, no SYN ACK is sent. Without the completion of the handshake, no determination of the target is possible other than SunRPC.

2.6 Correlations:
One address or another has scanned for port 111 on a network watched by the author at least weekly. Source ports included both ephemeral and priviledged. Mr. Northcutt does an excellent explaination in the IDFAQ. A DENY entry was made by the ipchains firewall on the primary gateway. Also, see the CVE extract above.

2.7 Evidence of Active Targeting:
None whatsoever. The service does not run on this network segment. All addresses were probed sequentially. This was a recon.

2.8 Severity:
(Target Criticality + Attack Lethality) - (System Countermeasures + Network Countermeasures).

(2 + 5) - (5 + 5) = -4

1. **Target Criticality:** The service is not running on the network. (1)
2. **Attack Lethality:** Several Root compromises exist on port 111. (5)
3. **System Countermeasures:** Systems are running 2.2.14 or better linux kernels. Services that aren't running are completely removed from machines. All systems have tcpwrappers configured. (5)
4. **Network Countermeasures:** Latest kernel means better firewalls. Masquerading (NAT) in place. Shadow runs out of band with no IP address bound to external I/F. (5)

Here is a negative severity! Much has to do with the configuration of the network.

2.9 Defensive Recomendations:
Continue to watch logs and industry sources for new vulnerabilities. Stay the course!

2.10 Multiple choice test question:
#) Based on the following trace:
20:21:52.030152 Them.Net.109.189.4944 > My.Net.Here.162.111: S 2174170510:2174170510(0) win 32120 "<"mss 1460,sackOK,timestamp 465272020,nop,wscale 0">" (DF) (ttl 38, id 57310)
20:21:52.037577 Them.Net.109.189.4945 > My.Net.Here.163.111: S 2183830373:2183830373(0) win 32120 "<"mss 1460,sackOK,timestamp 465272020,nop,wscale 0">" (DF) (ttl 38, id 57311)
20:21:52.136325 Them.Net.109.189.4952 > My.Net.Here.170.111: S 2176551491:2176551491(0) win 32120 "<"mss 1460,sackOK,timestamp 465272020,nop,wscale 0">" (DF) (ttl 38, id 57318)
20:21:52.136679 My.Net.Here.170 > Them.Net.109.189: icmp: My.Net.Here.170 tcp port 111 unreachable [tos 0xc0] (ttl 255, id 44729)
20:21:52.317855 Them.Net.109.189.4972 > My.Net.Here.190.111: S 2185022683:2185022683(0) win 32120 "<"mss 1460,sackOK,timestamp 465272020,nop,wscale 0">" (DF) (ttl 38, id 57338)
20:21:52.320443 My.Net.Here.190.111 > Them.Net.109.189.4972: R 0:0(0) ack 2185022684 win 0 (ttl 255, id 153)

The Arrival timestamps above suggest that this scan is sequential across the subnet. What confirms it?
a) The IP ID numbers are spaced evenly with the addresses. ***
b) The TCP option timestamp is constant
c) Because Stephen says so
d) The TCP sequence numbers are increasing.

Network Detect #3

3. Hack-A-Tack Trojan Scan

top

Another Ascend Catch...
Aug 28 23:46:08 My.Net.Here.129 ASCEND: wan3 udp My.Net.Here.128;31789 <- Them.Net.70.16;31790 29 !pass (toudp-1)
Aug 28 23:46:08 My.Net.Here.129 ASCEND: wan3 udp My.Net.Here.129;31789 <- Them.Net.70.16;31790 29 !pass (toudp-1)

Aug 28 23:46:09 My.Net.Here.129 ASCEND: wan3 udp My.Net.Here.131;31789 <- Them.Net.70.16;31790 29 !pass (toudp-1)
Aug 28 23:46:09 My.Net.Here.129 ASCEND: wan3 udp My.Net.Here.132;31789 <- Them.Net.70.16;31790 29 !pass (toudp-1)
Aug 28 23:46:09 My.Net.Here.129 ASCEND: wan3 udp My.Net.Here.133;31789 <- Them.Net.70.16;31790 29 !pass (toudp-1)
Aug 28 23:46:09 My.Net.Here.129 ASCEND: wan3 udp My.Net.Here.134;31789 <- Them.Net.70.16;31790 46 !pass (toudp-1)
Aug 28 23:46:09 My.Net.Here.129 ASCEND: wan3 udp My.Net.Here.135;31789 <- Them.Net.70.16;31790 29 !pass (toudp-1)

Explanation of Fields: Ascend Pipeline 130(v.35) Firewall Logs
See Item 1 for individual fields.

3.1 Source of Trace:
The same network as #1

3.2 Tool or technique detect was generated by:
Ascend Firewall pointed at Syslog facility. (Same as #1)

3.3 Spoof Probability:
Not likely. The sender needs to see the responses. UDP packets do not get responses except from the target service. From my knowledge of Them.Net, it is on the Them.Net.70 network, at least.

3.4 Description of attack:
This is a trojan. A proper return is all the warning you will get. It is likely that a major portion of the Them.Net network was scanned, or more. From Simovits Consulting comes the listing of Hack-A-Tack at UDP 31789. The responses to this recon will be the Hack-A-Tack server. This trojan is designed for Windows 9[5,8] computers. An interesting anomoly is the 46 byte packet sent to My.Net.Here.134. As said above, the logger does not offer the opportunity to investigate these packets further.

3.5 Attack Mechanism:
This is a classic script kiddie tool. It even opens the cupholder! The server can be distributed in any way the cracker can get a user to run the install program. The client Uploads your IP to a FTP-Server. When a server is online, it downloads your IP from the FTP-Server and contacts your client. You can see the computer name of the server and its IP in the window below the transmit ip button. The client has a high-performance IP-Scanner integrated. Search an engine for a better description. *The Hack-A-Tack site caused my browser to slow down noticably until reboot.*

3.6 Correlations:
This attack is so well documented, this analysis serves as a correlation. The link in 3.5 is to SANS own offerings. This detect also highlights the need to implement better security across the entire internet. If the logs are not being watched, they're flying blind.

3.7 Evidence of Active Targeting:
Each host once = NO! The attacker is using the built-in subnet scanner of the Hack-A-Tack client. He's looking to capitalize on someone elses work (luck).

3.8 Severity:
(Target Criticality + Attack Lethality) - (System Countermeasures + Network Countermeasures).

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

1. **Target Criticality:** The target operating system does run on the network. There are some systems in use by users with additional priviledges. (3)
2. **Attack Lethality:** The target operating system has few protective facilities. The attack gives GUI access to functions that are not available to the average user. Scan has little effect on uninfected systems. (3)
3. **System Countermeasures:** Systems have automatically updated virus checkers with notification of users and admins, as well as a central log facility. Operating systems have no native log facilities. (4)
4. **Network Countermeasures:** Masquerading (NAT) in place. Shadow runs on network. Users are notified when virus checker updates are sent. Responses to problems are high. (5)

Negative Severity is indicated due to support given by management and users. Concientious effort is applied throughout the organization. Computers will never be smart; just dumb, really really fast. People provide the security.

3.9 Defensive Recommendation:
Maintain education plan and personal contact with users. Increase deployment of Shadow sensors.

3.10 Multiple choice test question:
The most important part of the security infrastructure is:
a) The Intrusion Detection system
b) The System Logs
c) The Attackers
d) The educated / motivated user. ***

Network Detect #4

4. "Whackjob" Trojan Scan

top

Shadow / TCPDump output...
/"home/shadow/LOG/site/Sep08
09:48:32.528084 Them.Net.243.254.38404 > My.Net.Here.162.12631: S 1432665828:1432665828(0) win 16384 <'mss 1460,nop,nop,sackOK> (DF) (ttl 110, id 22038)
09:48:32.554936 Them.Net.243.254.39721 > My.Net.Here.163.12631: S 1432752576:1432752576(0) win 16384 <'mss 1460,nop,nop,sackOK> (DF) (ttl 110, id 22042)
09:48:32.801612 Them.Net.243.254.25100 > My.Net.Here.170.12631: S 1433081841:1433081841(0) win 16384 <'mss 1460,nop,nop,sackOK> (DF) (ttl 110, id 22134)
09:48:33.370305 Them.Net.243.254.22178 > My.Net.Here.190.12631: S 1434227055:1434227055(0) win 16384 <'mss 1460,nop,nop,sackOK> (DF) (ttl 110, id 22180)
09:48:33.374481 My.Net.Here.190.12631 > Them.Net.243.254.22178: R 0:0(0) ack 1434227056 win 0 (ttl 255, id 161)
09:48:33.948649 Them.Net.243.254.22178 > My.Net.Here.190.12631: S 1434227055:1434227055(0) win 16384 <'mss 1460,nop,nop,sackOK> (DF) (ttl 111, id 22218)
09:48:33.950550 My.Net.Here.190.12631 > Them.Net.243.254.22178: R 0:0(0) ack 1 win 0 (ttl 255, id 162)
09:48:34.548301 Them.Net.243.254.60270 > My.Net.Here.190.12631: S 1434227055:1434227055(0) win 16384 <'mss 1460,nop,nop,sackOK> (DF) (ttl 111, id 22261)
09:48:34.550438 My.Net.Here.190.12631 > Them.Net.243.254.60270: R 0:0(0) ack 1434227056 win 0 (ttl 255, id 163)

09:48:35.449357 Them.Net.243.254.38404 > My.Net.Here.162.12631: S 1432665828:1432665828(0) win 16384 <'mss 1460,nop,nop,sackOK> (DF) (ttl 111, id 22332)
09:48:35.548297 Them.Net.243.254.39721 > My.Net.Here.163.12631: S 1432752576:1432752576(0) win 16384 <'mss 1460,nop,nop,sackOK> (DF) (ttl 111, id 22339)
09:48:35.766155 Them.Net.243.254.25100 > My.Net.Here.170.12631: S 1433081841:1433081841(0) win 16384 <'mss 1460,nop,nop,sackOK> (DF) (ttl 111, id 22361)

" Explaination of Fields: TCPDump packet capture output
See Item 2 for individual fields.

4.1 Source of Trace:
Same Network as number 2.

4.2 Tool or technique detect was generated by
Same Tools as #2. Three days were searched for completenesses sake.

4.3 Spoof Probability
Even lower than the three above! Notice the change in pattern as the scan gets to host My.Net.Here.190. Reset packets are sent and Them.Net.243.254 retries two more times. The scan is then repeated within the second, skipping host 190. This indicates a response to the reset packets.

4.4 Description of attack
From the Readme file of Whackjob:
"The Best Way to Get NetBus on a PC and Keep it there!" And, "Test it with antivirus and detector programs and let me know of any that clean it and I'll revise the program to beat it..." Whew... This is actually a kiddie's script to break into social engineering and more. The readme actually describes how to get an unsuspecting user to run the install program. The "game" is actually kind of cute. The underlying program is NetBus! NetBus and BackOrifice are competing "remote control" programs.

4.5 Attack Mechanism
Simovits Consulting and Sys-Security list WhackJob at TCP port 12631. Lance Spitzner's snort script looks for it at 23456. Lance's snort script looks for installed, communicating Netbus client / server pairs. Netbus can be installed to use any given port, especially because it runs on windows boxes which know not the meaning of priviledged ports. Whitehats has a great discription. The GUI client gives malicious or joyriding adventurers GUI tools to functions that most users are not aware of.

4.6 Correlations
SANS has the latest hit of this port, but it was in a group of other ports to the same address. Guy Bruneau shows 8 scans in Feb 1999. There are a lot of sources and download sites for this Trojan Horse.

4.7 Evidence of Active Targeting
Probable subnet focus. The scan comes back around to hit us again. Is this directed at any response recieved from our network?
The sending host was probably involved in other activities, surmised by the increment of the IP IDs. My.Net.Here does not have all addresses populated, so the border router does not have ARP entries to send to.
We have been given a meter to measure the consistency of packet sending rate by George Bakos in his GCIA practical in section 2.7. Barring a one to one relation in IP Address increments, IP IDs or sending ports, more intrinsic methods are necessary. It is a derivative of rate and it's constancy will give us an idea of whether the stimulus of returns has an effect on the scan. Note well that this is Addresses over time.

Acceleration as to addresses indicated a targeted scan. A slowing in rate will indicate additional processing on the scanning computer's part (wait state included). The return stimulus in this case is a telltale router (returns ICMP Host unreach's), the stated resets, and dead air (no response). To quote:

Arrival timestamp B - Arrival timestamp A = Tb-Ta , and Destination IP B - Destination IP A = Db-Da
(Tb-Ta)/(Db-Da) = x

Given constant network conditions, the quotient (x) will remain constant if scanning across the entire subnet. This points to a subnet scan beyond this network. Let's see the results for our case:

Timestamp	Target IP address	Tb-Ta	Db-Da	x
09:48:32.528084	My.Net.Here.162	-	-	-
09:48:32.554936	My.Net.Here.163	0.027	1	0.027
09:48:32.801612	My.Net.Here.170	0.156898	7	0.0224
09:48:33.370305	My.Net.Here.190	0.385264	20	0.0193
09:48:34.550438	Last Reset Sent Out	-	-	-
09:48:35.449357	My.Net.Here.162	0.898919	-28(1)	0.898
09:48:35.548297	My.Net.Here.163	0.098940	1	0.099
09:48:35.766155	My.Net.Here.170	0.217858	7	0.0311

There is a 0.9 second pause between the last response from us and the start of the second scan. The first scan establishes a rate of around 47 addresses per second (one per .021 seconds). This remains fairly constant and indicates a random scan.
On the return trip, the first and second addresses occur at a rate of 10 addresses per second (one per .099 seconds). This would tend to indicate that the scan was a) waiting for a response (this host returned dead air), or b) had found another host where the scan was successful. It is doubtful a scan would continue to eat bandwidth after a hit, but...
The next seven addresses appear to come at a rate of 30 addresses per second; until you consider that we dont see the arrival of the 164 packet. If we assume that the scanner also waited 0.099 seconds for .163 to respond, the value of x for 164 - 170 becomes 0.019819,
(0.217858 - 0.098940) / 6 addresses. (Drop the one that waited)
This is well in line with the rate established in scan #1. It indicates that 162 and 163 were targeted on the second scan, with the rest of the scan as smoke! Host 170 may also have been paused for, but we lack the information to verify. Host 170's OS does not support Netbus.

4.8 Severity (according to given formula)
(Target Criticality + Attack Lethality) - (System Countermeasures + Network Countermeasures).

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

1. **Target Criticality:** The target operating system does run on the network. There are some systems in use by users with additional priviledges. (3)
2. **Attack Lethality:** The attack gives GUI access to functions that are not available to the average user. Author has vowed to defeat trojan checkers. Scan has little effect on uninfected systems. (3)
3. **System Countermeasures:** Static systems have automatically updated virus checkers with

notification of users and admins, as well as a central log facility. (4)

4. **Network Countermeasures:** Masquerading (NAT) in place. Shadow runs on network. Users are notified when virus checker updates are sent. Responses to problems are high. (5)

4.9 Defensive Recommendation

This scan is immediate in its results. Continue to watch shadow alerts. Continue to maintain the currency of the virus checker.

4.10 Multiple choice test question

Based on the following: (it fit on one line at 1024x 768)

```
09:48:32.528084 Them.Net.243.254.38404 > My.Net.Here.162.12631: S 1432665828:1432665828(0) win
16384 <'mss 1460,nop,nop,sackOK> (DF) (ttl 110, id 22038)
09:48:32.554936 Them.Net.243.254.39721 > My.Net.Here.163.12631: S 1432752576:1432752576(0) win
16384 <'mss 1460,nop,nop,sackOK> (DF) (ttl 110, id 22042)
09:48:32.801612 Them.Net.243.254.25100 > My.Net.Here.170.12631: S 1433081841:1433081841(0) win
16384 <'mss 1460,nop,nop,sackOK> (DF) (ttl 110, id 22134)
09:48:33.370305 Them.Net.243.254.22178 > My.Net.Here.190.12631: S 1434227055:1434227055(0) win
16384 <'mss 1460,nop,nop,sackOK> (DF) (ttl 110, id 22180)
09:48:33.374481 My.Net.Here.190.12631 > Them.Net.243.254.22178: R 0:0(0) ack 1434227056 win 0 (ttl
255, id 161)
09:48:33.948649 Them.Net.243.254.22178 > My.Net.Here.190.12631: S 1434227055:1434227055(0) win
16384 <'mss 1460,nop,nop,sackOK> (DF) (ttl 111, id 22218)
09:48:33.950550 My.Net.Here.190.12631 > Them.Net.243.254.22178: R 0:0(0) ack 1 win 0 (ttl 255, id 162)
09:48:34.548301 Them.Net.243.254.60270 > My.Net.Here.190.12631: S 1434227055:1434227055(0) win
16384 <'mss 1460,nop,nop,sackOK> (DF) (ttl 111, id 22261)
09:48:34.550438 My.Net.Here.190.12631 > Them.Net.243.254.60270: R 0:0(0) ack 1434227056 win 0 (ttl
255, id 163)
09:48:35.449357 Them.Net.243.254.38404 > My.Net.Here.162.12631: S 1432665828:1432665828(0) win
16384 <'mss 1460,nop,nop,sackOK> (DF) (ttl 111, id 22332)
09:48:35.548297 Them.Net.243.254.39721 > My.Net.Here.163.12631: S 1432752576:1432752576(0) win
16384 <'mss 1460,nop,nop,sackOK> (DF) (ttl 111, id 22339)
09:48:35.766155 Them.Net.243.254.25100 > My.Net.Here.170.12631: S 1433081841:1433081841(0) win
16384 <'mss 1460,nop,nop,sackOK> (DF) (ttl 111, id 22361)
```

All hosts on the subnet are listed. We can infer that all addresses in the subnet received a packet. Why!
a) The TCP sequence numbers are rising in a predictable pattern.
b) The IP ID numbers decrease sequentially.
c) The arrival timestamps and addresses increase proportionally. ***
d) the destination IP Addresses are increasing in value.

Attack Evaluation:

Netbus "Remote Administration Tool"

top

If you watch packet logs, or IDS systems for any length of time you will eventually find a scan for TCP port 12345. This is the default port of netbus, a "remote administration" tool. The author claims that this is a viable administrative facility in a class with fdisk, format and PCAnywhere. This might have flown until

v1.7, in which a redirect feature was added. This gives unethical users a measure of anonymity. Read on...

The NetBus program was released in March of 1998. The original interface was in Swedish, but within a month it was released in English. It has gone through minor and major version releases. It serves to illustrate that the problem is not the program, but the person at the keyboard.

Back Orifice 2000 had been released in the previous months to the chagrin of the media and masses. BO2K was the end of computer privacy, according to the mainstream accounts. This eventually died off as folks learned that it was relatively easy to defeat; it just took all of us. It was in the wake of this revelation that world could be saved that Netbus was released. Even though it was more powerful, almost no one noticed.

The latest version is Netbus Pro v2.1. It costs $15 (US). Over 266K downloads are listed on CNet. UltraAccess.Net bought the rights, and the program was rewritten to show an icon in the systray when active. The icon can be removed "only with local access"! Given the nature of Windows management, this only serves as another hurdle to hop with a better utility for a goal. The icon does serve as notice on the local desktop, blurring the lines of trojan / netadmin. Symantec and Panda have removed detection of Netbus Pro. Perhaps, as netbus advocates point out, Netbus is a victim of the order in which the features were added to the program.

Netbus has many point and click features. It's client interface is essentially a panel of buttons. It will work on any Win32 system. Version 1.7 will scan for servers and allow IP redirection. This redirection will allow the client to use one server to control another. The control will appear to be coming from the middle server.

Redirection is a minor convenience to an administrator, that must be contemplated before use to properly utilize resources (Why would he be doing his job with netbus?). It is just as easy to disconnect from one server and connect to the other. The main benefit of this feature is to mask the ultimate source of the initiative. This masking affords a "cracker" a measure of anonymity. The "cracker" could delete all three text logs. Netbus does not interact with native windows logging facilities, other than the confines of the present user. The middle server would need another computer logging packets to prove that he was not the initiator of the mischief. This would complicate investigations involving the local network, as well as internet traces. The average user would be hard pressed to defend himself against Netbus mischief.

Netbus server and client both save logs, temporary, and configuration information in plain text files. The password of the server is saved in [servername].ini, in plain text. This feature offers the mischievous "cracker" point and click control should he be allowed, or otherwise gain, access to this file. This utility also makes available a MRU list of hosts "administered". Some would say that proper configuration would protect the system. With the lack of logging and encryption, the producers are abrogating the responsibility to participate in defense in depth.

The market will ultimately decide, as it always does. We hope that eventually, a system will be chosen that has stronger protections than are afforded by Netbus. As it is a windows program, the history behind it will do well to help security professionals discuss the attributes of the program with management to help make this decision.

The client in this case was a windows 2000 box. The server was a windows 95 box. They were on the same ethernet segment. Netbus runs on TCP. After the obligatory three way handshake, 13 bytes are sent to initiate the connection:

```
12:34:49.437658 192.168.5.40.12345 > 192.168.5.47.1234: P [tcp sum ok] 1:14(13) ack 1
win 8760 (DF) (ttl 128, id 19975, len 53)
0x00004500 0035 4e07 4000 8006 2114 c0a8 0528 E..5N.@...!....(
```

```
0x0010c0a8 052f 3039 04d2 0057 9398 839b 6cf9 .../09...W....l.
0x00205018 2238 84ad 0000 4e65 7442 7573 2031 P."8....NetBus.1
0x00302e36 3020 0d                            .60..
12:34:49.544014 192.168.5.47.1234 > 192.168.5.40.12345: . [tcp sum ok] 1:1(0) ack 14 win
17507 (DF) (ttl 128, id 3121, len 40)
0x00004500 0028 0c31 4000 8006 62f7 c0a8 052f E..(.1@...b..../
0x0010c0a8 0528 04d2 3039 839b 6cf9 0057 93a5 ...(..09..l..W..
0x00205010 4463 262d 0000 4e65 7442 7573      P.Dc&-..NetBus
```

After the connection is established on port 12345, the server (victim; user...) sends the version to the client (cracker). The ack reads 40 bytes in the header, but carries six extra bytes with the word NetBus as a reply.

Next we see the attacker hit the File Manager Button, and then the Get Files button:

```
12:34:58.119340 192.168.5.47.1234 > 192.168.5.40.12345: P [tcp sum ok] 1:10(9) ack 14
win 17507 (DF) (ttl 128, id 3122, len 49)
0x00004500 0031 0c32 4000 8006 62ed c0a8 052f E..1.2@...b..../
0x0010c0a8 0528 04d2 3039 839b 6cf9 0057 93a5 ...(..09..l..W..
0x00205018 4463 888b 0000 4765 7444 6973 6b73 P.Dc....GetDisks
0x00300d                                      .
12:34:58.245427 192.168.5.40.12345 > 192.168.5.47.1234: . [tcp sum ok] 14:14(0) ack 10
win 8751 (DF) (ttl 128, id 20231, len 40)
0x00004500 0028 4f07 4000 8006 2021 c0a8 0528 E..(O.@....!...(
0x0010c0a8 052f 3039 04d2 0057 93a5 839b 6d02 .../09...W....m.
0x00205010 222f 4858 0000 0000 0000 6973      P."/HX......is
12:35:03.078601 192.168.5.40.12345 > 192.168.5.47.1234: P [tcp sum ok] 14:31(17) ack 10
win 8751 (DF) (ttl 128, id 20487, len 57)
0x00004500 0039 5007 4000 8006 1f10 c0a8 0528 E..9P.@........(
0x0010c0a8 052f 3039 04d2 0057 93a5 839b 6d02 .../09...W....m.
0x00205018 222f 9adb 0000 4469 736b 7344 6f6e P."/....DisksDon
0x0030653b 3332 3737 3637 0d                  e;327767.
12:35:03.079985 192.168.5.47.1235 > 192.168.5.40.12346: S [tcp sum ok]
2211460162:2211460162(0) win 16384 (DF) (ttl 128, id 3123, len 48)
0x00004500 0030 0c33 4000 8006 62ed c0a8 052f E..0.3@...b..../
0x0010c0a8 0528 04d3 303a 83d0 3442 0000 0000 ...(..0:..4B....
0x00207002 4000 ca57 0000 0204 05b4 0101 0402 p.@..W..........
12:35:03.080512 192.168.5.40.12346 > 192.168.5.47.1235: S [tcp sum ok]
5753062:5753062(0) ack 2211460163 win 8760
0x00004500 0030 5107 4000 8006 1e19 c0a8 0528 E..0Q.@........(
0x0010c0a8 052f 303a 04d3 0057 c8e6 83d0 3443 .../0:...W....4C
0x00207012 2238 1ed1 0000 0204 05b4 0101 0402 p."8............
12:35:03.080681 192.168.5.47.1235 > 192.168.5.40.12346: . [tcp sum ok] 1:1(0) ack 1 win
17520 (DF) (ttl 128, id 3124, len 40)
0x00004500 0028 0c34 4000 8006 62f4 c0a8 052f E..(.4@...b..../
0x0010c0a8 0528 04d3 303a 83d0 3443 0057 c8e7 ...(..0:..4C.W..
0x00205010 4470 295d 0000 0204 05b4 0101      P.Dp)]........
12:35:03.088378 192.168.5.40.12346 > 192.168.5.47.1235: . 1:1461(1460) ack 1 win 8760
(DF) (ttl 128, id 20999, len 1500)
0x00004500 05dc 5207 4000 8006 176d c0a8 0528 E...R.@....m...(
0x0010c0a8 052f 303a 04d3 0057 c8e7 83d0 3443 .../0:...W....4C
0x00205010 2238 db52 0000 633a 0d0a 0942 4f4f P."8.R..c:...BOO
0x0030544c 4f47 2e54 5854 2028 3232 6b29 0d0a TLOG.TXT.(22k)..
0x00400942 4f4f 544c 4f47 2e50 5256 2028 3232 .BOOTLOG.PRV.(22
```

Obviously, the last packet above is truncated; for brevity's sake. The server then transmits the entire local directory tree, including all disks (there are two in the win95 machine). We see the "GetDisks" command

and the ack. When the client is ready it sends a "DisksDone". That old windows box took 4.8+ seconds to get the whole listing. The Client then starts a new connection to transfer the file list. We now have two open connections.
The server starts the new connection within 0.0015 seconds, making this occurance automatic. We see the FINs below:

```
12:35:03.690939 192.168.5.40.12346 > 192.168.5.47.1235: P [tcp sum ok] 327681:327768(87)
ack 1 win 8760 (DF) (ttl 128, id 16904, len 127)
12:35:03.865264 192.168.5.47.1235 > 192.168.5.40.12346: . [tcp sum ok] 1:1(0) ack 327768
win 17433 (DF) (ttl 128, id 3246, len 40)
12:35:03.865746 192.168.5.40.12346 > 192.168.5.47.1235: F [tcp sum ok] 327768:327768(0)
ack 1 win 8760 (DF) (ttl 128, id 17160, len 40)
12:35:03.865916 192.168.5.47.1235 > 192.168.5.40.12346: . [tcp sum ok] 1:1(0) ack 327769
win 17433 (DF) (ttl 128, id 3247, len 40)
12:35:06.247689 192.168.5.47.1235 > 192.168.5.40.12346: F [tcp sum ok] 1:1(0) ack 327769
win 17433 (DF) (ttl 128, id 3248, len 40)
12:35:06.248199 192.168.5.40.12346 > 192.168.5.47.1235: . [tcp sum ok] 327769:327769(0)
ack 2 win 8760 (DF) (ttl 128, id 17416, len 40)
```

So, now we are back to one connection. The attacker spends a little time checking the list for his target. The client and server processes wait silently while he makes his choice. Then he selects the target and fires away... Notice the port on the syn packet (1236)

```
12:35:17.634508 192.168.5.47.1234 > 192.168.5.40.12345: P [tcp sum ok] 10:38(28) ack 31
win 17490 (DF) (ttl 128, id 3249, len 68)
0x00004500 0044 0cb1 4000 8006 625b c0a8 052f E..D..@...b[.../
0x0010c0a8 0528 04d2 3039 839b 6d02 0057 93b6 ...(..09..m..W..
0x00205018 4452 a639 0000 446f 776e 6c6f 6164 P.DR.9..Download
0x00304669 6c65 3b63 3a5c 424f 4f54 4c4f 472e File;c:\BOOTLOG.
0x00405458 540d                               TXT.
12:35:17.637018 192.168.5.40.12345 > 192.168.5.47.1234: P [tcp sum ok] 31:53(22) ack 38
win 8723 (DF) (ttl 128, id 17672, len 62)
0x00004500 003e 4508 4000 8006 2a0a c0a8 0528 E..>E.@...*....(
0x0010c0a8 052f 3039 04d2 0057 93b6 839b 6d1e .../09...W....m.
0x00205018 2213 c1c2 0000 446f 776e 6c6f 6164 P."".....Download
0x00305265 6164 793b 313b 3232 3835 340d      Ready;1;22854.
12:35:17.638348 192.168.5.47.1236 > 192.168.5.40.12346: S [tcp sum ok]
2215131425:2215131425(0) win 16384 (DF) (ttl 128, id 3250, len 48)
```

Boom, There it is! This was also tested on a Windows NT default installation. It happily pulled a 4K file named sam._ from c:\winnt\repair. Why wouldn't it? An interesting aspect there is that it would not attach unless someone was logged in. That's some remote administration tool.

```
12:35:32.649827 192.168.5.47.1234 > 192.168.5.40.12345: P [tcp sum ok] 38:64(26) ack 53
win 17468 (DF) (ttl 128, id 3264, len 66)
0x00004500 0042 0cc0 4000 8006 624e c0a8 052f E..B..@...bN.../
0x0010c0a8 0528 04d2 3039 839b 6d1e 0057 93cc ...(..09..m..W..
0x00205018 443c 05a1 0000 446f 776e 6c6f 6164 P.D<....Download
0x00304669 6c65 3b64 3a5c 5745 4433 412e 4a50 File;d:\WED3A.JP
0x0040470d                                    G.
12:35:32.653402 192.168.5.40.12345 > 192.168.5.47.1234: P [tcp sum ok] 53:75(22) ack 64
win 8697 (DF) (ttl 128, id 23048, len 62)
0x00004500 003e 5a08 4000 8006 150a c0a8 0528 E..>Z.@........(
0x0010c0a8 052f 3039 04d2 0057 93cc 839b 6d38 .../09...W....m8
0x00205018 21f9 bfa8 0000 446f 776e 6c6f 6164 P.!.....Download
0x00305265 6164 793b 313b 3833 3338 350d      Ready;1;83385.
```

```
12:35:32.657094 192.168.5.47.1237 > 192.168.5.40.12346: S [tcp sum ok]
2218925549:2218925549(0) win 16384 (DF) (ttl 128, id 3265, len 48)
0x00004500 0030 0cc1 4000 8006 625f c0a8 052f E..0..@...b_.../
0x0010c0a8 0528 04d5 303a 8442 1ded 0000 0000 ...(..0:.B......
0x00207002 4000 e038 0000 0204 05b4 0101 0402 p.@..8..........
12:35:32.657560 192.168.5.40.12346 > 192.168.5.47.1237: S [tcp sum ok]
5782645:5782645(0) ack 2218925550 win 8760
0x00004500 0030 5b08 4000 8006 1418 c0a8 0528 E..0[.@........(
0x0010c0a8 052f 303a 04d5 0058 3c75 8442 1dee ".../0:...X p."8."..........
0x00207012 2238 c122 0000 0204 05b4 0101 0402
12:35:32.657736 192.168.5.47.1237 > 192.168.5.40.12346: . [tcp sum ok] 1:1(0) ack 1 win
17520 (DF) (ttl 128, id 3266, len 40)
0x00004500 0028 0cc2 4000 8006 6266 c0a8 052f E..(..@...bf.../
0x0010c0a8 0528 04d5 303a 8442 1dee 0058 3c76 ...(..0:.B...X(%60)v
0x00205010 4470 cbae 0000 0204 05b4 0101      P.Dp..........
```

Now our attacker is looking at a jpeg file from the remote machine. Probably in preparation for posting on the victims desktop. He could just as well had uploaded one of his own for posting.
Notice the Three Way Handshake. The incremented port indicates the new connection. The extra bytes are probably a tcpdump problem. A version is being used that includes a text representation of the Hex Dump. This version is not fully tested. (Hmm... Shadow *will* do content?) In support of that, tcpdump Hex Dumps show 0x0204 05b4 in many places with '. 0x05b4 = 1460 decimal.

What do you know...

```
12:35:58.784697 192.168.5.47.1234 > 192.168.5.40.12345: P [tcp sum ok] 64:87(23) ack 75
win 17446 (DF) (ttl 128, id 3301, len 63)
0x00004500 003f 0ce5 4000 8006 622c c0a8 052f E..?..@...b,.../
0x0010c0a8 0528 04d2 3039 839b 6d38 0057 93e2 ...(..09..m8.W..
0x00205018 4426 4d96 0000 5368 6f77 496d 6167 P.D&M...ShowImag
0x0030653b 643a 5c77 6564 3361 2e6a 7067 0d   e;d:\wed3a.jpg.
12:35:58.967109 192.168.5.40.12345 > 192.168.5.47.1234: . [tcp sum ok] 75:75(0) ack 87
win 8674 (DF) (ttl 128, id 39944, len 40)
0x00004500 0028 9c08 4000 8006 d31f c0a8 0528 E..(..@........(
0x0010c0a8 052f 3039 04d2 0057 93e2 839b 6d4f .../09...W....mO
0x00205010 21e2 481b 0000 5368 6f77 496d      P.!.H...ShowIm
```

And

```
12:38:29.300254 192.168.5.47.1234 > 192.168.5.40.12345: P [tcp sum ok] 177:185(8) ack
106 win 17415 (DF) (ttl 128, id 3308, len 48)
0x00004500 0030 0cec 4000 8006 6234 c0a8 052f E..0..@...b4.../
0x0010c0a8 0528 04d2 3039 839b 6da9 0057 9401 ...(..09..m..W..
0x00205018 4407 8c4a 0000 4765 7449 6e66 6f0d P.D..J..GetInfo.
12:38:29.302148 192.168.5.40.12345 > 192.168.5.47.1234: P [tcp sum ok] 106:214(108) ack
185 win 8576 (DF) (ttl 128, id 41480, len 148)
0x00004500 0094 a208 4000 8006 ccb3 c0a8 0528 E.....@........(
0x0010c0a8 052f 3039 04d2 0057 9401 839b 6db1 .../09...W....m.
0x00205018 2180 77d4 0000 496e 666f 3b50 726f P.!.w...Info;Pro
0x00306772 616d 2050 6174 683a 2043 3a5c 5041 gram.Path:.C:\PA
0x00405443 482e 4558 457c 5265 7374 6172 7420 TCH.EXE|Restart.
0x00507065 7273 6973 7465 6e74 3a20 5965 737c persistent:.Yes|
0x00604c6f 6769 6e20 4944 3a20 4253 4348 4e5a Login.ID:.BSCHNZ
0x00704c7c 436c 6965 6e74 7320 636f 6e6e 6563 L|Clients.connec
```

```
0x00807465 6420 746f 2074 6869 7320 686f 7374 ted.to.this.host
0x00903a20 310d                               :.1.
12:38:29.470845 192.168.5.47.1234 > 192.168.5.40.12345: . [tcp sum ok] 185:185(0) ack
214 win 17307 (DF) (ttl 128, id 3309, len 40)
0x00004500 0028 0ced 4000 8006 623b c0a8 052f E..(..@...b;.../
0x0010c0a8 0528 04d2 3039 839b 6db1 0057 946d ...(..09..m..W.m
0x00205010 439b 2575 0000 4765 7449 6e66      P.C.%u..GetInf
```

The two actions above take place on the original connection. The command to show image, and path to the image shown, are short. The information on the status of the server is only 108 bytes. These commands happens on the original. No further anomalous traffic was noticed.

Before I put this spell into action, I wanted to know how to remove it! I saw what happened to Mickey in Fantasia. By all accounts the best way was with the client itself. I had to remove Sophos Anti-Virus to gain access to v1.7, even though they do not mention Netbus on their website. When I reisntalled Sophos, it found nothing. I combed through the common auto-start places in Windows and found nothing. The program also copied itself and keydrop.dll to c:\windows. Nasty!

In conclusion Netbus is a risky program for anyone to be running on their system. It has a keylogger, file manager, and screen dumper, to name a few. If somehow, someone were to defeat the windows security features, the netbus password (if any) is saved in plain text in a readily recognizable file. The keylogger and file content format make it a bad choice. Address scanning and address/port redirection lock this down as a hacker tool to watch for. For now, keep 12345 on your radar screens.

Analyze This!

Sample Site Snort Data

top

This is in response to an RFP to perform a security review of the information technology investment of Beer Inc. A month's worth of Snort Data is available, as well as results of a previous month's data and analysis. Beer Inc. did not provide any other concrete data. The previous analyses may not be entirely correct. In any case, they pertains to last month's data.

There is no possibility of determining timezone. Having one snort sensor makes host time synchronization moot in this RFP. The timestamps will reflect a single timezone as well. Time syncronisation is important for correllation purposes. The data from Beer Inc. shows a network that is connected to the internet. Bonafide services are not a part of the given information. It appears that Beer Inc. has a Class B subnet that is quite fully populated.

This Response to the RFP should not be construed as a comprehensive analysis. More information is necessary to give such an opinion. There very well may be hosts not listed here that are under the control of someone you don't know, much less trust or approve of, that are not listed. This data also negects to take into account the people that use the computers. My colleagues and I are experienced in examining systems for security as well as reliability. There are many pitfalls that IDS's are not presently equipped to handle. That being said...

One or more computers are configured to use SNMP. The logged SNMP packets are all sent to the MY.NET.101.192 host. No SNMP traps are indicated, though this is probably a limitation of the ruleset used on the SNORT sensor. Several Commercial Off The Shelf (COTS) packages use SNMP, especially in a Windows network. This protocol is known to have weak authentication and several other vulnerabilities. It should be watched.

There is evidence that only one computer is communicating with MY.NET.101.192 using SNMP the protocol. The diverse Class C address space suggests this, but does not require it. DHCP may be in use on this network. No bootp or DHCP traffic is shown, but there may not have been such a rule in Snort. This could also be a program installed in a users windows profile database. At no time is more than one machine communicating with any other using this community name. This may or may not be troubling. Interviews with your people would settle this very quickly, and probably painlessly. We can verify the single use in the time sorted list of starting and ending SNMP packets from each address:

```
./SnortA3.txt:06/30-09:27:45.475626 [**] SNMP public access [**] MY.NET.97.109:1052 ->
MY.NET.101.192:161
./SnortA3.txt:06/30-09:33:19.522128 [**] SNMP public access [**] MY.NET.97.109:1239 ->
MY.NET.101.192:161

./SnortA8.txt:07/10-18:14:05.658622 [**] SNMP public access [**] MY.NET.97.159:1042 ->
MY.NET.101.192:161
./SnortA8.txt:07/10-18:19:17.718892 [**] SNMP public access [**] MY.NET.97.159:1063 ->
MY.NET.101.192:161

./SnortA7.txt:07/11-15:55:47.105484 [**] SNMP public access [**] MY.NET.97.122:1059 ->
MY.NET.101.192:161
./SnortA7.txt:07/11-16:01:26.018511 [**] SNMP public access [**] MY.NET.97.122:1128 ->
MY.NET.101.192:161

./SnortA9.txt:07/12-18:15:22.067534 [**] SNMP public access [**] MY.NET.97.87:1047 ->
MY.NET.101.192:161
./SnortA9.txt:07/12-19:04:44.318341 [**] SNMP public access [**] MY.NET.97.87:1186 ->
MY.NET.101.192:161

./SnortA10.txt:07/14-08:13:15.198541 [**] SNMP public access [**] MY.NET.97.237:1041 ->
MY.NET.101.192:161
./SnortA10.txt:07/14-10:54:59.758360 [**] SNMP public access [**] MY.NET.97.237:1530 ->
MY.NET.101.192:161

./SnortA10.txt:07/14-11:28:07.526936 [**] SNMP public access [**] MY.NET.97.80:1623 ->
MY.NET.101.192:161
./SnortA10.txt:07/14-13:24:56.765119 [**] SNMP public access [**] MY.NET.97.80:2025 ->
MY.NET.101.192:161

./SnortA10.txt:07/14-16:28:36.694062 [**] SNMP public access [**] MY.NET.97.239:1054 ->
MY.NET.101.192:161
./SnortA10.txt:07/14-17:46:03.465637 [**] SNMP public access [**] MY.NET.97.239:1226 ->
MY.NET.101.192:161

./SnortA12.txt:07/19-19:11:06.553815 [**] SNMP public access [**] MY.NET.97.219:1097 ->
MY.NET.101.192:161
./SnortA12.txt:07/19-19:41:35.404971 [**] SNMP public access [**] MY.NET.97.219:1236 ->
MY.NET.101.192:161

./SnortA13.txt:07/26-09:33:38.673441 [**] SNMP public access [**] MY.NET.97.186:1048 ->
MY.NET.101.192:161
./SnortA13.txt:07/26-12:03:09.806664 [**] SNMP public access [**] MY.NET.97.186:1644 ->
MY.NET.101.192:161

./SnortA13.txt:07/26-18:55:27.075096 [**] SNMP public access [**] MY.NET.97.198:1042 ->
MY.NET.101.192:161
./SnortA13.txt:07/26-18:56:36.078798 [**] SNMP public access [**] MY.NET.97.198:1055 ->
MY.NET.101.192:161
```

```
./SnortA14.txt:07/27-10:52:31.005194 [**] SNMP public access [**] MY.NET.98.150:1042 ->
MY.NET.101.192:161
./SnortA14.txt:07/27-10:55:08.174404 [**] SNMP public access [**] MY.NET.98.150:1070 ->
MY.NET.101.192:161

./SnortA14.txt:07/27-11:05:18.132611 [**] SNMP public access [**] MY.NET.98.152:1120 ->
MY.NET.101.192:161
./SnortA14.txt:07/27-11:53:40.053535 [**] SNMP public access [**] MY.NET.98.152:1400 ->
MY.NET.101.192:161

./SnortA14.txt:07/27-18:29:38.240554 [**] SNMP public access [**] MY.NET.98.118:1055 ->
MY.NET.101.192:161
./SnortA14.txt:07/27-20:13:44.848545 [**] SNMP public access [**] MY.NET.98.118:1367 ->
MY.NET.101.192:161

./SnortA14.txt:07/27-22:16:21.823966 [**] SNMP public access [**] MY.NET.97.208:1056 ->
MY.NET.101.192:161
./SnortA14.txt:07/27-22:16:45.690201 [**] SNMP public access [**] MY.NET.97.208:1074 ->
MY.NET.101.192:161

./SnortA17.txt:07/28-07:59:52.711253 [**] SNMP public access [**] MY.NET.98.178:1047 ->
MY.NET.101.192:161
./SnortA17.txt:07/28-08:03:15.087525 [**] SNMP public access [**] MY.NET.98.178:1069 ->
MY.NET.101.192:161

./SnortA17.txt:07/28-18:14:30.150085 [**] SNMP public access [**] MY.NET.98.202:1042 ->
MY.NET.101.192:161
./SnortA17.txt:07/28-18:17:48.025096 [**] SNMP public access [**] MY.NET.98.202:1063 ->
MY.NET.101.192:161

./SnortA17.txt:07/28-18:27:05.758630 [**] SNMP public access [**] MY.NET.98.127:1041 ->
MY.NET.101.192:161
./SnortA17.txt:07/28-18:32:24.955421 [**] SNMP public access [**] MY.NET.98.127:1071 ->
MY.NET.101.192:161

./SnortA16.txt:07/30-18:44:36.293952 [**] SNMP public access [**] MY.NET.97.219:1050 ->
MY.NET.101.192:161
./SnortA16.txt:07/30-18:47:07.537665 [**] SNMP public access [**] MY.NET.97.219:1117 ->
MY.NET.101.192:161

./SnortAle.txt:08/01-08:49:56.352489 [**] SNMP public access [**] MY.NET.98.148:1042 ->
MY.NET.101.192:161
./SnortAle.txt:08/01-09:08:56.362101 [**] SNMP public access [**] MY.NET.98.148:1103 ->
MY.NET.101.192:161

./SnortAle.txt:08/01-09:16:25.042943 [**] SNMP public access [**] MY.NET.97.176:1126 ->
MY.NET.101.192:161
./SnortAle.txt:08/01-09:59:17.955356 [**] SNMP public access [**] MY.NET.97.176:1238 ->
MY.NET.101.192:161

./SnortAle.txt:08/01-10:01:47.045751 [**] SNMP public access [**] MY.NET.98.187:1249 ->
MY.NET.101.192:161
./SnortAle.txt:08/01-10:01:47.224503 [**] SNMP public access [**] MY.NET.98.187:1249 ->
MY.NET.101.192:161

./SnortAle.txt:08/01-12:18:32.492268 [**] SNMP public access [**] MY.NET.97.186:1601 ->
MY.NET.101.192:161

./SnortAle.txt:08/01-13:37:59.716824 [**] SNMP public access [**] MY.NET.97.191:1835 ->
MY.NET.101.192:161
```

```
./SnortA18.txt:08/03-11:30:14.088089 [**] SNMP public access [**] MY.NET.97.242:1795 ->
MY.NET.101.192:161
./SnortA18.txt:08/03-11:30:15.770648 [**] SNMP public access [**] MY.NET.97.242:1796 ->
MY.NET.101.192:161

./SnortA19.txt:08/04-09:02:49.288711 [**] SNMP public access [**] MY.NET.97.207:1193 ->
MY.NET.101.192:161
./SnortA19.txt:08/04-09:02:50.774592 [**] SNMP public access [**] MY.NET.97.207:1195 ->
MY.NET.101.192:161

./SnortA19.txt:08/04-11:17:07.154986 [**] SNMP public access [**] MY.NET.97.214:1473 ->
MY.NET.101.192:161
./SnortA19.txt:08/04-12:45:44.243481 [**] SNMP public access [**] MY.NET.97.214:1822 ->
MY.NET.101.192:161

./SnortA19.txt:08/04-14:07:09.107525 [**] SNMP public access [**] MY.NET.97.238:2040 ->
MY.NET.101.192:161
./SnortA19.txt:08/04-14:38:59.371936 [**] SNMP public access [**] MY.NET.97.238:2125 ->
MY.NET.101.192:161

./SnortA19.txt:08/04-15:33:53.600373 [**] SNMP public access [**] MY.NET.97.225:2269 ->
MY.NET.101.192:161

./SnortA19.txt:08/04-17:41:43.833624 [**] SNMP public access [**] MY.NET.97.202:2603 ->
MY.NET.101.192:161

./SnortA19.txt:08/04-18:31:27.316215 [**] SNMP public access [**] MY.NET.98.194:2741 ->
MY.NET.101.192:161
./SnortA19.txt:08/04-18:31:33.304654 [**] SNMP public access [**] MY.NET.98.194:2741 ->
MY.NET.101.192:161
```

Below we see return traffic from MY.NET.101.192 to two addresses. This confirms that SNMP is an active service. The community name on MY.NET.101.192 is easily guessable (Public) and should be changed.

```
./SnortA17.txt:07/28-18:28:08.117468 [**] SNMP public access [**] MY.NET.98.127:1055 ->
MY.NET.101.192:161
./SnortA17.txt:07/28-18:28:08.327311 [**] SNMP public access [**] MY.NET.98.127:1056 ->
MY.NET.101.192:161
./SnortA17.txt:07/28-18:28:08.710307 [**] SNMP public access [**] MY.NET.98.127:1058 ->
MY.NET.101.192:161

./SnortS16.txt:Jul 28 18:28:08 MY.NET.101.192:161 -> MY.NET.98.127:1055 UDP
./SnortS16.txt:Jul 28 18:28:08 MY.NET.101.192:161 -> MY.NET.98.127:1056 UDP
./SnortS16.txt:Jul 28 18:28:08 MY.NET.101.192:161 -> MY.NET.98.127:1057 UDP

./SnortA16.txt:07/30-18:44:57.932330 [**] SNMP public access [**] MY.NET.97.219:1087 ->
MY.NET.101.192:161
./SnortA16.txt:07/30-18:44:58.141155 [**] SNMP public access [**] MY.NET.97.219:1088 ->
MY.NET.101.192:161

./SnortS12.txt:Jul 30 18:44:57 MY.NET.101.192:161 -> MY.NET.97.219:1087 UDP
./SnortS12.txt:Jul 30 18:44:58 MY.NET.101.192:161 -> MY.NET.97.219:1088 UDP
```

The extract below shows external addresses attempting to make contact with MY.NET.15.127 and MY.NET.70.66 on port 137. MS-Exchange will try to resolve the netbios name of the mail servers it connects with. It may be that MY.NET.15.127 is a mail server. At least one of the incoming source addresses resolves to a mail exchanger.

```
./SnortA17.txt:07/28-18:27:04.242680 [**] SMB Name Wildcard [**] MY.NET.101.160:137 ->
```

```
MY.NET.101.192:137
./SnortA18.txt:08/03-11:38:50.220540 [**] SMB Name Wildcard [**] 206.15.45.126:137 ->
MY.NET.70.66:137
./SnortA19.txt:08/04-12:39:44.098377 [**] SMB Name Wildcard [**] MY.NET.101.160:137 ->
MY.NET.101.192:137
./SnortA19.txt:08/04-12:41:06.156276 [**] SMB Name Wildcard [**] MY.NET.101.160:137 ->
MY.NET.101.192:137
./SnortA19.txt:08/04-16:23:34.611008 [**] SMB Name Wildcard [**] 132.201.232.167:137 ->
MY.NET.15.127:137
./SnortA19.txt:08/04-16:23:34.611091 [**] SMB Name Wildcard [**] 208.16.237.10:137 ->
MY.NET.15.127:137
./SnortA19.txt:08/04-16:23:42.770412 [**] SMB Name Wildcard [**] 208.16.237.10:137 ->
MY.NET.15.127:137
./SnortA19.txt:08/04-16:23:42.770527 [**] SMB Name Wildcard [**] 132.201.232.167:137 ->
MY.NET.15.127:137
./SnortA19.txt:08/04-16:23:44.293564 [**] SMB Name Wildcard [**] 132.201.232.167:137 ->
MY.NET.15.127:137
./SnortA19.txt:08/04-16:23:44.293784 [**] SMB Name Wildcard [**] 208.16.237.10:137 ->
MY.NET.15.127:137
./SnortA19.txt:08/04-16:23:50.888340 [**] SMB Name Wildcard [**] 132.201.232.167:137 ->
MY.NET.15.127:137
./SnortA19.txt:08/04-16:23:50.888445 [**] SMB Name Wildcard [**] 208.16.237.10:137 ->
MY.NET.15.127:137
./SnortA19.txt:08/04-18:31:28.807995 [**] SMB Name Wildcard [**] MY.NET.101.160:137 ->
MY.NET.101.192:137
./SnortA19.txt:08/04-18:31:30.295042 [**] SMB Name Wildcard [**] MY.NET.101.160:137 ->
MY.NET.101.192:137
```

There are 240 Alerts concerning Server Message Block Name Wildcards this month. The majority of these are from MY.NET.101.160 to MY.NET.101.192. MY.NET.101.160 is nearly the sole source of internal SMB Name Wildcards indicated this month. Last month, the hosts at MY.NET.70.234 and MY.NET.101.55 traded five SMB Wildcard calls. This month, MY.NET.70.66 seems to be trying to get something going with MY.NET.101.55, MY.NET.101.116, and MY.NET.101.117.

The SMB wildcard is generated by using NBTSTAT -A at the command line of a Windows host. Samba will also do this. This is the most likely initiative behind all SMB Wildcards. It should be noted that this does not happen in the course of "normal" user activity. MY.NET.101.160 should be checked for purpose and services. Removing NBTSTAT.EXE is the fix, but each workstation must be visited. The information can still be gathered from Network Neighborhood by logged on users.

MY.NET.101.89 appears to be a Network Address Translation server. The number of name responses to ports in and around the 50000+ range point to a linux masquerading computer. MY.NET.1.3 appears to be the nameserver for the subnet behind this NAT server. MY.NET.101.89 only shows up for failed DNS lookups, triggering a scan rule. Several hosts are using outside nameservers. These include MY.NET.98.152 and MY.NET.97.233. These nameservers include a german corporation and a .mil host. This is odd. MY.NET.1.4, MY.NET.1.5, MY.NET.1.8 and MY.NET.1.9 are probably nameservers. The following is a common pattern across the last two months:

```
./SnortA2.txt:06/29-04:50:11.984954 [**] spp_portscan: PORTSCAN DETECTED from
210.189.72.176 (STEALTH) [**]
./SnortA2.txt:06/29-04:40:23.638239 [**] SYN-FIN scan! [**] 210.189.72.176:0 ->
MY.NET.1.3:53
./SnortA2.txt:06/29-04:40:23.654151 [**] SYN-FIN scan! [**] 210.189.72.176:0 ->
MY.NET.1.4:53
./SnortA2.txt:06/29-04:40:23.676196 [**] SYN-FIN scan! [**] 210.189.72.176:0 ->
MY.NET.1.5:53
./SnortA2.txt:06/29-04:50:13.444377 [**] spp_portscan: portscan status from
```

```
210.189.72.176: 3 connections across 3 hosts: TCP(3), UDP(0) ST
./SnortA2.txt:06/29-04:50:14.734424 [**] spp_portscan: End of portscan from
210.189.72.176 (TOTAL HOSTS:3 TCP:3 UDP:0) [**]
```

This month the traces included tries on SMTP and FTP. As well, 210.222.31.100 appears looking for trinoo, "rockwell-csp" (Yeah, right!!!), and port 9704 (whatever that is). The common thread here is the three address pattern above. This is a targeted attack. Your firewall should be blocking these.

Last Month, Lenny Zeltzer built a set of tools to summarize Snort output. The Hosts he mentions in his counts are worth visiting again this month.

Lenny's Hosts

Alert Destinations

MY.NET.253.105 is probably still being used as a WinGate proxy. It also experienced attempts (successful?) at BIND, SOCKS, SunSeven, FTP (INVALIDACK ***FRPA*), and port scans from 216.131.17.59, 216.93.53.130, and 207.206.126.223. The "Attempts" warning should not be construed as 'unsuccessful attempts'. This host may be listed at http://www.cyberarmy.com. The proxy and firewall should be reconfigured to save your bandwidth by stopping this.

MY.NET.217.2 only showed up in 9 alerts and 14 scans this month. The alerts were all Socks tries. The scans included FTP, SubSeven and other Leet ports. It looks like Mr. ICQ has been reigned in.

MY.NET.253.41 appears to have smtp running on it. The following trace extract includes port 113 (identd) as well as several return hits on SMTP.

```
./SnortA20.txt:08/05-01:39:42.736908 [**] Watchlist 000222 NET-NCFC [**]
159.226.66.130:113 -> MY.NET.253.41:52416
./SnortA20.txt:08/05-01:39:44.130151 [**] Watchlist 000222 NET-NCFC [**]
159.226.66.130:1740 -> MY.NET.253.41:25
./SnortA20.txt:08/05-01:41:06.581911 [**] Watchlist 000222 NET-NCFC [**]
159.226.66.130:1740 -> MY.NET.253.41:25
./SnortA20.txt:08/05-01:41:06.581955 [**] Watchlist 000222 NET-NCFC [**]
159.226.66.130:1740 -> MY.NET.253.41:25
./SnortA20.txt:08/05-02:37:35.390428 [**] Watchlist 000222 NET-NCFC [**]
159.226.66.130:1784 -> MY.NET.253.41:25
./SnortA20.txt:08/05-02:37:41.450699 [**] Watchlist 000222 NET-NCFC [**]
159.226.66.130:113 -> MY.NET.253.41:52853
./SnortA20.txt:08/05-02:37:53.435836 [**] Watchlist 000222 NET-NCFC [**]
159.226.66.130:1784 -> MY.NET.253.41:25
./SnortA20.txt:08/05-02:39:15.247856 [**] Watchlist 000222 NET-NCFC [**]
159.226.66.130:1784 -> MY.NET.253.41:25
./SnortA20.txt:08/05-02:39:15.899066 [**] Watchlist 000222 NET-NCFC [**]
159.226.66.130:1784 -> MY.NET.253.41:25
./SnortA20.txt:08/05-02:39:15.920483 [**] Watchlist 000222 NET-NCFC [**]
159.226.66.130:1784 -> MY.NET.253.41:25
```

The trigger here appears to be the source address. Are we really doing business with the Chinese NOC?

IP block lookup for 159.226.64.164

whois -h whois.arin.net 159.226.64.164

```
The Computer Network Center Chinese Academy of Sciences (NET-NCFC)
```

```
[snipped]
```

Seriously, there is nothing wrong with the Chinese participating in e-mail. Targeting an entire block has its downfalls as well! Beware the boy's call of wolf. Sometimes he's right. MY.NET.100.230 also appears to be a Chinese SMTP favorite. Both of the "Chinese" Mail Servers experienced the same "false positives" relating to trinoo and portscanning. These two hosts are reacting as designed and appropriate.

Alert Source Hosts

The top two source hosts from last month do not show up (much) this month. 202.38.128.188 is not listed in this months logs. The operator of MY.NET.253.12 seems to have been effectively disciplined. That would be neat trick indeed to keep that host off the port scans. Speaking of port scans, Snort did not recognize any internal hosts running NMAP against anything this month. I guess the word got out.

The next two source hosts 142.150.225.137 (#3), and 204.60.176.2 (#5) do not show up this month. This is to be expected. Sources of SYN-FIN scans are not generally participating in that activity voluntarily. 159.226.45.3 (#4) is still happily accepting mail from several of our servers.

128.231.171.123 is still using MY.NET.253.105 as a Web Proxy. Someone really ought to reconfigure MY.NET.253.105. Anonymity has an effect similar to power, sooner or later you get corrupted.

Scan Destination Hosts

MY.NET.101.89 was tops in this category last month with 4115 hits. This month it only managed 843. I stand by my call that this is a masquerading box, with users visiting web sites with many offsite links. There may be a performance issue here.

MY.NET.70.234 is down in the noise floor, being scanned at ports 53, 21, and 110. There are five (of 16 total) of the following packets, suggesting that someone ought to kill port 137 at the firewall.

```
./SnortS3.txt:Jul 9 21:17:07 62.158.45.121:137 -> MY.NET.70.234:1057 UDP
```

MY.NET.97.73 is in the same boat as MY.NET.70.234. Last month's errant nameserver is not recieving any more queries. MY.NET.14.1 is totally gone from the logs! Neat trick.

MY.NET.101.192 has triggered an event for responding to SNMP traffic. There are approximately 1421 alerts on this account, and 29 Scan records. The scans are a false postitive from and errantly connected DHCP workstation (see above). Reconfigure that community! It is also the recipient of the SMB Wild Cards.

To save a Heading, not one of the top Scan Sources show up this month. Lenny's tools are too customized to adapt to this analysis system in the given time.

Final Caveat

./SnortA20.txt:08/05-18:30:03.777730 [**] IDS247 - MISC - Large UDP Packet [**] 211.40.176.214:29536 -> MY.NET.98.179:6970
./SnortA20.txt:08/05-18:30:03.835886 [**] IDS247 - MISC - Large UDP Packet [**] 211.40.176.214:29536 -> MY.NET.98.179:6970
...
./SnortA20.txt:08/05-18:59:56.542056 [**] IDS247 - MISC - Large UDP Packet [**] 211.40.176.214:29536

-> MY.NET.98.179:6970
./SnortA20.txt:08/05-18:59:58.027556 [**] IDS247 - MISC - Large UDP Packet [**] 211.40.176.214:29536 -> MY.NET.98.179:6970

There does seem to be a copy of GateCrasher appears to be running on host MY.NET.98.179 communicating with 211.40.176.214. GateCrasher is a netbus prototype. The trace (above) spans almost 30 minutes. It does have a keylogger. It is also more obscure, gaining some longevity in its rarety. A search for "GateCrasher" revealed Zero (0) hits at Sophos, NAI, and Symantec. The fix is easy though. Just remove it from CurrentVersion\Run in the windows registry, AND REBOOT. 216.46.175.35 was scanning the MY.NET.1xx.x subnet for GateCrasher as well.

Several hosts scanned the network for Nameservers, FTP Servers, SubSeven Servers, et al. These scans did not appear to have any responding traffic, reducing their severity. They have been given the notice an ankle-biter deserves. (Given the unknown ruleset!)

We look forward to serving you,

Bill Scherr IV

The Process

top
Consider this: Someone throws a stack of papers on your desk three feet high, each page full of lines and lines of hieroglyphics, and you have to make sense of it. Been there? If you're reading this chances are you have. That is what Stephen does at the end of the Intrusion Detection Track at SANS. Why not? That is what we will face in the wild, we just wont know as much about it!

To eat an elephant, you must have a sharp knife and patience to spare. I know of no tools better suited for this than bash with less and grep. You can not cut through the leg of the elephant, you must strip the meat off the bones. You must know that the bones exist, less can help you find them.

I'm sorry, did I neglect to mention that you have a megaphone welded by the the big end to your face, the elephant is still alive, and your knife is a regular leatherman? OK, the elephant is not alive, I was just checking.

Grep uses regular expressions. The first question you ask when you see that in the man page is: OK, whose definition of regular. They can take some time to get used to, but anything that powerful has to be somewhat complicated. Grep will also span files and directories. You can apply a single search to all the files. This was key. The main benefit of grep is the ability to use a file as an expression. This enables you to keep track of where you have been. As well, when used with less, you have some idea of how far you have to go. These can be great for the analyst's morale, or not.

Bash is the Bourne Again SHell (as if I had to say that). It is the operating environment. It is a highly configurable command line interface. Mine does not show the current directory. I prefer the Konsole wrapper from KDE as wide as I can stretch it. I set at least a 1000 line buffer, just in case.

Less keeps the lines from spinning out of view. It also gives you a virtual window wide enough to show any output I have come across. You also have a intermediate search function. This is useful for returning to the same spot if you have to get out for one reason or another (Rats, the battery!). It is also useful for getting to the exact spot in a grep when your filter returns more than you expect.

One thing to be careful of in less is that it is a virtual window. Once you hit "q" it may disappear. Several other features of the unix environment can save you from this, like the .bash/history file. I found GPM (the mouse driver) invaluable in this and many other projects. Being able to highlight something with your mouse and click it into another window is something I still do with joy.

I did use arachnophilia to generate most of the tags and other glitz. This is a windows html editor. It is a webmaster's dream, from what I understand. I use it to edit configuration files for users. It does have several table generation and tag inserting tools as well. It also does not add anything that it doesn't show by default.

On the unix side, I edited the report with Kedit. It has some text formatting (like hard line feeds on word wrap) and a search / replace function on a par with wordpad.

I made a cusory attempt at adapting Lenny Zeltser's perl scripts. Alas, the perl is not that strong with me (yet). Events in my life put learning the "brain" package out of reach. I commend Lenny on an excellent submission, and congratulate him on achieving honors status.

Beyond that, I'll wipe my mouth and look for some help disposing of all these elephant bones...

The Severity Formula is:
(Target Criticality + Attack Lethality) - (System Countermeasures + Network Countermeasures).

1. Target Criticality is an estimate of the power of the host and its function in the networked system. Is the targeted host vulnerable to this attack? What opportunities does the host offer to the rest of the network? etc...
2. Attack Lethality is an estimate of the intended effect on the target host. Does the attack offer root access? Are user accounts compromised by this attack? Does this attack render the target useless? etc...
3. System Countermeasures is an estimate of how well the target is protected by good administration. Is this machine running the latest OS version? Does this machine have its own firewall? Are all un-necessary services removed? Is tcp-wrappers running on this computer? etc...
4. Network Countermeasures is an estimate of how well best practices are applied to the entire network. Is a system in place to validate the firewall? Do Users have unrestricted access to modems? Are there firewalls on any modems? **Did the attack penetrate the defenses?** etc...

More information is available in the SANS Intrusion Detection course.