Global Information Assurance Certification Paper

*** Northcutt, you gotta read detect 8, full 10 point bonus for a new pattern. Otherwise, Eric shows a full grasp of analysis, research on the source address, check out detect 2 to see how this can pay off. 93 *

GIAC Practical Analysis
For IDIC Certification

Eric Gallagher

Detect 1

From the GIAC site:

```
Mar 24 01:54:58 cc1014244-a kernel:
securityalert: tcp if=ef0 from
24.3.57.38:11111 to 24.3.21.199 on unserved port 12345
Mar 24 03:14:13 cc1014244-a kernel:
securityalert: tcp if=ef0 from
171.214.113.228:2766 to 24.3.21.199 on unserved port 1243
Mar 24 04:45:01 cc1014244-a kernel:
securityalert: tcp if=ef0 from
208.61.109.243:3578 to 24.3.21.199 on unserved port 1243
Mar 24 04:45:06 cc1014244-a kernel:
securityalert: tcp if=ef0 from
208.61.109.243:3832 to 24.3.21.199 on unserved port 27347
Mar 24 05:40:42 cc1014244-a kernel:
securityalert: udp if=ef0 from
24.24.100.172:2147 to 24.3.21.199 on unserved port 137
Mar 24 14:56:08 cc1014244-a kernel:
securityalert: udp if=ef0 from
63.17.79.40:4294 to 24.3.21.199 on unserved port 137
Mar 24 17:20:44 cc1014244-a kernel:
securityalert: tcp if=ef0 from
62.6.100.45:1828 to 24.3.21.199 on unserved port 27374
Mar 24 20:50:47 cc1014244-a kernel:
securityalert: tcp if=ef0 from
194.27.62.179:4857 to 24.3.21.199 on unserved port 27374
```

Targeting: Here we have multiple hosts all centering on one target. This is actually an impressive trojan scan, since only one source host is used more than once. Someone either has a lot of machine access already or is capable of spoofing multiple host addresses quickly. The targeting in this case is extremely focused.

Intent: The intent seems to be to scan for multiple trojans and services including the popular NetBus, BackDoor/SubSeven, and SubSeven 2.1 ports, plus a look for the Netbios name service. Oddly, this does not seem to be an attempt to determine the type of system. This scan seems to assume the destination host is a MS-Windows PC. Perhaps an OS determination has already been made and this is the reason for the targeting described above.

Technique: This type of signature could come from any number of trojan scanners, especially since the attacker can customize them. A valid question here seems to be whether or not this is only one scan. If it were two interleaved scans, could an intrusion analyst separate them with this little data? Maybe not, which is something to keep in mind when attempting to analyze persistent scans for a definitive signature. Note here that there is some indication that the cracker may understand the difference between UDP and

TCP, since he/she uses UDP to check for the Netbios name service port instead of TCP, which leaves a larger trace footprint. That's more likely to be due to a sophisticated tool than a sophisticated attacker but this scan (if it is only one) seems fairly sophisticated already, given the number of source hosts used. That's the biggest signature of this technique.

History: A lookup of these multiple source hosts reveals that 24.3.57.38 is registered as cc940888-a.owml1.md.home.com; 171.214.113.228 is ABD671E4.ipt.aol.com; 208.61.109.243 is adsl-61-109-243.sdf.bellsouth.net; 24.24.100.172 is m8hDs1n172.midsouth.rr.com; 63.17.79.40 is 1Cust40.tnt33.dfw5.da.uu.net; 62.6.100.45 is host62-6-100-45.btinternet.com. It's only mildly surprising that all these addresses seem to come from ISPs and Telcos. What's much more surprising is that all of them (except one, 194.27.62.179, which is not registered) are in the DNS. Perhaps this reveals something about the attacker's technique, such as a tendency to find registered hosts to crack or spoof. If so, the instance of 194.27.62.179 may reveal something more: a machine address for which the attacker did not use his/her normal techniques, implying a possibly more traceable connection.

Severity: Low. The ports seem to be unserved, so the likelihood of a successful crack seems remote. Since the target is a home.com machine, cc1014244-a.hwrd1.md.home.com, it's also unlikely that any critical data is at risk, although the users on that machine may feel differently.

Action Recommendation: Look at the packet contents to see the number of hops for each machine. If they are all the same number, for instance, it would appear almost certain that these are spoofed addresses. On the other hand, if the hop counts seem real, this may be a genuine distributed scan. Also, it's important to watch the target machine closely, probably with a packet sniffer devoted to its address for a while. This has to be the best approach, given the range of addresses of the apparent distributed scan.

Detect 2

From the GIAC Site:

```
**********************************************************
Snort Alert Report at Sun Mar 19 00:09:40 2000
**********************************************************
[**] SYN-FIN scan! [**]
03/18-02:25:46.587048 194.112.42.193:53 -> MY.NET.1.1:53
[**] SYN-FIN scan! [**]
03/18-02:25:46.608902 194.112.42.193:53 -> MY.NET.1.2:53
[**] SYN-FIN scan! [**]
03/18-02:25:46.630302 194.112.42.193:53 -> MY.NET.1.3:53
[**] SYN-FIN scan! [**]
03/18-02:25:46.651612 194.112.42.193:53 -> MY.NET.1.4:53
[**] SYN-FIN scan! [**]
03/18-02:25:46.672247 194.112.42.193:53 -> MY.NET.1.5:53
[**] SYN-FIN scan! [**]
03/18-02:25:46.692692 194.112.42.193:53 -> MY.NET.1.6:53
[**] SYN-FIN scan! [**]
03/18-02:25:46.710063 194.112.42.193:53 -> MY.NET.1.7:53
[**] SYN-FIN scan! [**]
03/18-02:25:46.734950 194.112.42.193:53 -> MY.NET.1.8:53
[**] SYN-FIN scan! [**]
03/18-02:25:46.753369 194.112.42.193:53 -> MY.NET.1.9:53
[**] SYN-FIN scan! [**]
03/18-02:25:46.770616 194.112.42.193:53 -> MY.NET.1.10:53
[**] SYN-FIN scan! [**]
03/18-02:25:46.790791 194.112.42.193:53 -> MY.NET.1.11:53
[**] SYN-FIN scan! [**]
```

03/18-02:25:46.808440 194.112.42.193:53 -> MY.NET.1.12:53
[**] SYN-FIN scan! [**]
03/18-02:25:46.828961 194.112.42.193:53 -> MY.NET.1.13:53
[**] SYN-FIN scan! [**]
03/18-02:25:46.851741 194.112.42.193:53 -> MY.NET.1.14:53
[**] SYN-FIN scan! [**]
03/18-02:25:46.873321 194.112.42.193:53 -> MY.NET.1.15:53
[**] SYN-FIN scan! [**]
03/18-02:25:46.930169 194.112.42.193:53 -> MY.NET.1.16:53
[**] SYN-FIN scan! [**]
03/18-02:25:46.951785 194.112.42.193:53 -> MY.NET.1.17:53
[**] SYN-FIN scan! [**]
03/18-02:25:46.957222 194.112.42.193:53 -> MY.NET.1.18:53
[**] SYN-FIN scan! [**]
03/18-02:25:46.965475 194.112.42.193:53 -> MY.NET.1.19:53
[**] SYN-FIN scan! [**]
03/18-02:25:46.976485 194.112.42.193:53 -> MY.NET.1.20:53
[**] SYN-FIN scan! [**]
03/18-02:25:46.990538 194.112.42.193:53 -> MY.NET.1.21:53
[**] SYN-FIN scan! [**]
03/18-02:25:47.017313 194.112.42.193:53 -> MY.NET.1.22:53
[**] SYN-FIN scan! [**]
03/18-02:25:47.030220 194.112.42.193:53 -> MY.NET.1.23:53
[**] SYN-FIN scan! [**]
03/18-02:25:47.050602 194.112.42.193:53 -> MY.NET.1.24:53
[**] SYN-FIN scan! [**]
03/18-02:25:47.070911 194.112.42.193:53 -> MY.NET.1.25:53
[**] SYN-FIN scan! [**]

Targeting: There is targeting here, obviously, but of a network rather than of a particular host.

Intent: This appears to be a machine-by-machine scan for DNS servers. The scanner may be hoping to find an old DNS server running without patches, in which case there are canned exploits available.

Techniques: The attacker may be trying to elude detection by intrusion detection devices that look for SYN-only connections and do not look for impossible flag combinations such as SYN-FIN. (By which I mean normally impossible settings; it is obviously possible to craft them.) Also, since Linux boxes will often respond to a SYN-FIN with a SYN-FIN-ACK, the attacker may be trying to locate this OS.

History: The source machine address is registered as dh007-00.web.dircon.net, another ISP. Beyond that, the time stamps indicate a quick, automated scan but not much more.

Severity: Low, since this is a rather crude and ultimately unsuccessful scan.

Action Recommendation: Call dircon.net and let them know they may have a compromised machine.

Detect 3

(Nslookup says ras5.iet.co.il == 194.90.83.122 == Intelligent Electronics, Israel.
dig -x says 194.90.83.130 == ras5.iet.co.il == NetVision, Israel)

Mar 22 04:38:13 dns1 telnetd[220834]: warning:
/etc/hosts.allow, line 17: host name/address mismatch:
194.90.83.130 != ras5.iet.co.il
Mar 22 04:38:13 dns1 telnetd[220834]: refused

connect from 194.90.83.130
Mar 22 04:38:16 dns3 in.telnetd[19860]: refused
connect from 194.90.83.130

Targeting: Why is someone trying to telnet into these DNS servers? This is very obviously targeted at a particular network or series of known DNS machines. Note the absence of attempts to non-DNS server addresses. This is obviously not a scan.

Intent: Someone seems to have had reason to believe he/she could log in to these DNS servers. It's hard to imagine a completely innocent reason for doing this, since most former employees who might have worked on these servers would know that a login from a remote host ought to raise alarms. There are a nearly infinite number of malicious reasons for attempting to login to these, including crashing the servers and denying network services to the former company. But the best reasons involve changing the DNS flat files to either degrade network service slowly or to create false DNS records for addresses to be used in a later attack.

Techniques: The /etc/hosts.allow reference implies that tcpwrappers are in place on the destination machine and that there is actually an entry the attacker expected to use! This could indicate a current employee working on the DNS servers. However, there is a mismatch between the hosts.allow address and the actual source of the login attempt. If the attacker installed tcpwrappers on the DNS server a while ago and left this back door for himself/herself, he/she may have been stopped only because of the differences between the name lookup and reverse lookup in the DNS system.

History: The problem of mismatched forward and reverse DNS lookups still exists! Since this was noticed and reported on March 22, perhaps I don't find this surprising (although perhaps I am cynical). In any case, this is what my DNS servers told me:

reverse lookup

Name: ras5.iet.co.il
Address: 194.90.83.130

name lookup

Name: ras5.iet.co.il
Address: 194.90.83.122

Severity: High. This is not merely a scan but a crack attempt, however unsophisticated it might appear. Many systems depend on DNS entries for validating remote hosts. Most web service will grind to a halt without available DNS servers. Also, if this login attempt is not the result of a present or past employee leaving a back door in the tcpwrappers file, then there is a possibility that the machine is already compromised and that the tcpwrappers allow entry was created by a cracker with admin privileges.

Action Recommendation:
Look up the /etc/hosts.allow entries and correct them to remove the possibility of an illegal login. Look throughout the hosts for signs of system compromise, since it's a real possibility. If everything looks good, put Tripwire or another such system in place to watch for changes in the hosts.allow file as well as other files that might allow remote access.

Detect 4

From the GIAC Site:

Feb 27 02:36:00 cc1014244-a kernel: securityalert: tcp if=ef0 from
24.129.24.105:4109 to 24.3.21.199 on unserved port 8080

Feb 27 08:07:29 cc1014244-a kernel: securityalert: tcp if=ef0 from 212.49.226.251:1525 to 24.3.21.199 on unserved port 1080
Feb 27 12:09:07 cc1014244-a kernel: securityalert: tcp if=ef0 from 24.6.118.9:1178 to 24.3.21.199 on unserved port 27374
Feb 27 13:59:03 cc1014244-a kernel: securityalert: tcp if=ef0 from 12.77.12.140:4211 to 24.3.21.199 on unserved port 1080

Targeting: Several source hosts are targeting a home.com server.

Intent: The first host probes the HTTP-ALT port 8080, probably hoping to elicit a response from a proxy server. The second source host tries a SOCKS packet. The third host attempts to locate the SubSeven trojan. The last searches for SOCKS again. Each of these attempts seems to be the prelude to an attack, an attempt to determine if 24.3.21.199 has a particularly exploitable service.

Techniques: Each of these probes against the 24.3.21.199 system seem to be a singleton, which is somewhat unusual in the GIAC trace archives. Alternatively, these source addresses could be spoofed as part of a slow, port-scanning recon mission. (There is no evidence for this in the data above. In fact, the repeat of the SOCKS probe argues against it.) I would argue these were uncoordinated, seperate attacks unless presented with additional data leading to a contrary conclusion.

History: The source host 24.129.24.105 is surf03-24-105.naplesfl.net and 12.77.12.140 is 140.atlanta-31-32rs.ga.dial-access.att.net, so both of those machines reside at telcos, which is no surprise. However, 212.49.226.251 is dyn251-ras3.froglike.co.uk, which doesn't offer a service that I can find. That's no big deal but 24.6.118.9 is ci945915-a.grnvle1.sc.home.com. Since the target host, 24.3.21.199, is cc1014244-a.hwrd1.md.home.com, this means that one home.com site seems to be searching another for the SubSeven trojan. Perhaps this is a home.com security scanner showing up in the logs. If not, however, this is an unfortunate sign. The other probes are almost benign in comparison.

Severity: Low, but only for the target host. This may be unusual but I would have to give a higher severity rating (medium, a 3 on the 1-5 scale) to the other home.com machine doing the SubSeven probe. It might not appear to be a target here but that may only be because it has already been infected with SubSeven or compromised in some other way.

Action Recommendation: Watch for all of these other addresses in the logs, in case there are low, slow scans going on. Look at the TTL (time to live) of these packets and make sure there's no spoofing going on that would indicate a coordinated attack. Most of all, contact the administrator of the other home.com machine, 24.6.118.9, to make sure it the SubSeven port probe was a deliberate security measure. (Really. I hope this has been done in real life.)

Detect 5

Mar 27 14:59:06 pooky kernel: Packet log:
input DENY eth0 PROTO=6
38.31.117.17:2443 xxx.xxx.xxx.204:12345
L=48 S=0x00 I=2221 F=0x4000 T=116

Mar 27 14:59:06 morton kernel: Packet log:
input DENY eth0 PROTO=6
38.31.117.17:2440 xxx.xxx.xxx.201:12345
L=48 S=0x00 I=1453 F=0x4000 T=116 SYN

Mar 27 14:59:06 www kernel: Packet log:
input DENY eth0 PROTO=6
38.31.117.17:2444 xxx.xxx.xxx.205:12345
L=48 S=0x00 I=2477 F=0x4000 T=116 SYN

[**] Netbus/GabanBus [**]
03/27-14:59:06.560788 0:E0:D0:10:EF:7F ->
0:20:78:14:1F:7B type:0x800 len:0x3E
38.31.117.17:2443 ->
xxx.xxx.xxx.204:12345 TCP TTL:116 TOS:0x0 ID:2221 DF
S***** Seq: 0x46145DB Ack: 0x0 Win: 0x2000
TCP Options => MSS: 536 NOP NOP SackOK

Targeting: In this trace, the LAN is being targeted rather than any single machine.

Intent: This is a straightforward netbus scan. PROTO=6 indicates TCP and ports 12345 and 12346 for TCP belong to the netbus service.

Techniques: The point of the SYN packets is to initiate a TCP handshake and establish, through the returned ACK, which particular machine on the xxx.xxx.xxx network has netbus running. After this, the netbus attack can begin.

History: The source host is ip17.minneapolis16.mn.pub-ip.psi.net.

Severity: This seems to be a fairly determined scan, as much as can be inferred from this data. Given that netbus is a remote systems control program, this makes the severity at least a 2 on a scale of 1-5. However, this is only the information gathering stage of an attack.

Action Recommendation: Notify PSInet of the netbus scan and politely ask them to stop. If it is possible for the local (not PSInet) network administrator to deny netbus scans at the firewall, he/she should do so.

Detect 6

From local tcpdump:

15:09:34.402695 webanalyzer.somenet.gov.137 > MY.NET.1.8.137: udp 50
15:09:35.894114 webanalyzer.somenet.gov.137 > MY.NET.1.8.137: udp 50
15:09:37.393968 webanalyzer.somenet.gov.137 > MY.NET.1.8.137: udp 50
15:12:14.537011 webanalyzer.somenet.gov.137 > MY.NET.1.124.137: udp 50
15:12:16.036261 webanalyzer.somenet.gov.137 > MY.NET.1.124.137: udp 50
15:12:17.535980 webanalyzer.somenet.gov.137 > MY.NET.1.124.137: udp 50
16:06:19.881266 webanalyzer.somenet.gov.137 > MY.NET.2.7.137: udp 50
16:06:21.372891 webanalyzer.somenet.gov.137 > MY.NET.2.7.137: udp 50
16:06:22.872286 webanalyzer.somenet.gov.137 > MY.NET.2.7.137: udp 50
16:08:56.209947 webanalyzer.somenet.gov.137 > MY.NET.1.49.137: udp 50
16:08:57.701418 webanalyzer.somenet.gov.137 > MY.NET.1.49.137: udp 50
16:08:59.202073 webanalyzer.somenet.gov.137 > MY.NET.1.49.137: udp 50
16:18:31.553671 webanalyzer.somenet.gov.137 > MY.NET.1.49.137: udp 50
16:18:33.049327 webanalyzer.somenet.gov.137 > MY.NET.1.49.137: udp 50
16:18:34.550180 webanalyzer.somenet.gov.137 > MY.NET.1.49.137: udp 50

The MY.NET.2.7 address supposedly does not have a machine at it. Same for the MY.NET.1.49 address. The MY.NET.1.8 machine was turned off when it was supposed to receive these packets. None of the sites responds to a ping at the time of the investigation, so it doesn't look like anyone's sneaked a machine onto the network.

Targeting: This seems to be poor but definite targeting. It's likely that I'm only seeing these packets in my switched environment because the switch doesn't know what to do with them. The machines really are not online despite the way they've been targeted. That part is odd.

Intent: This is coming from a government site, so I'm inclined to believe there are good intentions behind this traffic or, at worst, oblivious non-intentions. The problems with this assumption are that a) the reason for targeting non-existent nodes is not apparent and b) government computers have been cracked and used for malicious activity in the past, so a government address is no guarantee of good behavior.

Techniques: It's apparently common to see NT-based websites respond to a port 80 Reset by sending some netbios packets a while later. But no one in the IDIC courses seemed to know why. However, these aren't even the result of such a response. This is deliberate polling of some sort. There may have been Win95/98/NT machines at MY.NET.1.49 and MY.NET.2.7 at some point in time, in which case the webanalyzer machine expects a netbios response because of it. The troubling part is that so many people bring in laptops with generous shares to the entire world. I know the MY.NET.1.8 machine had such open shares, once. (They were set by an NT administrator, too.)

History: The machine is a legitimate one but its owners don't seem to know why it's doing this. I also notice there's usually about two hours' time between one set of UDP packets and the next. So I think the webanalyzer site does this on a regular schedule.

Severity: Low but worth watching, which I continue to do. I'm in the process of getting to know my network traffic, or I wouldn't have been so suspicious in the beginning. But it seems the suspicion is well founded in some ways because the netbios polling continues across the campus of my company and no routers, firewalls, or administrators are acting to stop it.

Action Recommendation: Keep bugging the network administrators between divisions. Also, since there's been no luck with indirect contact with the webanalyzer site administrators, I should go to direct contact. At this point, I would not block the traffic because, well, I can't. I don't have firewall authority between divisions. That belongs to another division at my company.

Detect 7

From tcpdump:

23:41:00.300460 212.108.4.153.80 > MY.NET.2.5.63694: R 0:0(0) ack 3618313488 win 0 [tos 0x20]
03:16:41.998171 212.108.4.154.80 > MY.NET.1.55.28430: R 0:0(0) ack 3087512077 win 0 [tos 0x20]
03:35:06.871755 212.108.4.153.80 > MY.NET.1.67.8497: R 0:0(0) ack 971785307 win 0 [tos 0x20]
04:50:22.608371 212.108.4.154.80 > MY.NET.1.47.21427: R 0:0(0) ack 859913805 win 0 [tos 0x20]
05:19:01.152543 212.108.4.154.80 > MY.NET.2.57.8117: R 0:0(0) ack 1544118613 win 0 [tos 0x20]
05:50:25.648816 212.108.4.153.80 > MY.NET.2.18.60179: R 0:0(0) ack 3575887728 win 0 [tos 0x20]
09:48:12.439100 212.108.4.154.80 > MY.NET.1.37.36871: R 0:0(0) ack 2048786269 win 0 [tos 0x20] (ttl 49, id 469)
15:41:52.825619 212.108.4.153.80 > MY.NET.2.7.29912: R 0:0(0) ack 2045183752 win 0 [tos 0x20]
15:56:36.579087 212.108.4.152.80 > MY.NET.1.79.28047: R 0:0(0) ack 2433793293 win 0 [tos 0x20]
15:57:01.985515 212.108.4.176.80 > MY.NET.1.67.52111: R 0:0(0) ack 893334842 win 0 [tos 0x20]
16:04:47.246961 212.108.4.153.80 > MY.NET.2.15.65292: R 0:0(0) ack 1714241317 win 0 [tos 0x20]
18:09:50.759571 212.108.4.176.80 > MY.NET.1.12.64557: R 0:0(0) ack

4052446528 win 0 [tos 0x20]
18:43:24.396538 212.108.4.180.80 > MY.NET.2.7.3846: R 0:0(0) ack
2901898522 win 0 [tos 0x20]
19:43:49.925457 212.108.4.152.80 > MY.NET.1.126.15109: R 0:0(0) ack
2587073094 win 0 [tos 0x20]
06:18:05.089964 212.108.4.152.80 > MY.NET.1.79.36762: R 0:0(0) ack
3824865851 win 0 [tos 0x20]
11:15:20.125994 212.108.4.176.80 > MY.NET.2.57.15529: R 0:0(0) ack
219468150 win 0 [tos 0x20]
12:39:00.479330 212.108.4.180.80 > MY.NET.1.90.3129: R 0:0(0) ack
1044520726 win 0 [tos 0x20]
04:02:36.470463 212.108.4.180.80 > MY.NET.1.79.13902: R 0:0(0) ack
3730043421 win 0 [tos 0x20]
07:05:01.224520 212.108.4.176.80 > MY.NET.1.90.5870: R 0:0(0) ack
3100835108 win 0 [tos 0x20]
07:05:01.257073 212.108.4.153.80 > MY.NET.1.90.5870: R 0:0(0) ack
3100835108 win 0 [tos 0x20]
08:06:01.000345 212.108.4.152.80 > MY.NET.2.57.14045: R 0:0(0) ack
3144551732 win 0 [tos 0x20]
08:13:33.906535 212.108.4.178.80 > MY.NET.1.68.21353: R 0:0(0) ack
2548065327 win 0 [tos 0x20]

Targeting: It's hard to tell how many machines and networks were targeted but I would suspect quite a lot. The fact that it's coming in over port 80 means it was designed to get through firewalls configured to allow that port (and that means most of them).

Intent: This isn't a search for a particular trojan port or legitimate but exploitable service. This is an attempt to map our network. As with any mapping scan, the purpose is to reconnoiter for a later attack. This particular hacker/cracker shows a great deal of confidence and patience. He (or she) seems to feel that knowing the network nodes will be enough for the first stage, implying an ability to determine multiple OSes (with a customized nmap or something similar?) and launch appropriate attacks.

Technique: Interesting, to say the least. Reset scans are generally hard to detect. It's lucky for me that I looked at the packets manually rather than through an IDS, since IDSes can miss these. They are a stealthy sort of scan. Reset scans depend on ICMP to help with the network mapping, a fact I had to look up. In many circumstances, machines will make no reply to a Reset packet. Why would they? However, our router probably sends information back to the attacker when a particular address proves to be unreachable. This allows for an inverse map of the nodes.

History: The 212.108.4.X network is registered to NS2U.NET. However, NS2U does not have a website that I could find. Nor does it offer publically-available FTP. These services may be on non-standard ports or they may not exist at all. Both possibilities are suspicious anyway. The owners of this domain are probably hostile or have been seriously spoofed, not an easy thing to happen. Hostile is a relative term here, of course. They could be mapping MY COMPANY for academic or commercial reasons. Probably not, though. The upshot is that 212.108.4.153, 154, etc, etc, do not seem to be DNS registered, nor to they respond with webservers on those ports. In fact, NS2U.net does not seem to have a webserver at all. This confirms the hostility of the scan, as if that's needed.

Severity: This is only a scan but because it is fairly sophisticated, I would have to rate this as a medium risk. Certainly people who know to scan from different machines on a port that almost all firewalls let pass (the web port 80) so that they can map firewalled sites for attack are capable of launching serious attacks.

Action Recommendation: Crank up a real IDS and scan for incoming and outgoing network traffic with associations to these NS2U addresses. Of course, it would be nice to contact NS2U too, but I can't find any information on them. I don't know if it's practical for CIT to block this whole network at the router or not. Probably not. That's another reason to worry, of course.

Detect 8

From local tcpdump:

21:57:18.468912 MY.NET.2.4.4277 > MY.NET.1.0.1600: udp 46
21:59:18.478004 MY.NET.2.4.4278 > MY.NET.1.0.1600: udp 46
22:01:18.487730 MY.NET.2.4.4279 > MY.NET.1.0.1600: udp 46
22:03:18.498327 MY.NET.2.4.4280 > MY.NET.1.0.1600: udp 46
22:05:18.507726 MY.NET.2.4.4281 > MY.NET.1.0.1600: udp 46
22:07:18.517192 MY.NET.2.4.4282 > MY.NET.1.0.1600: udp 46
22:09:18.527483 MY.NET.2.4.4283 > MY.NET.1.0.1600: udp 46
22:11:18.536690 MY.NET.2.4.4284 > MY.NET.1.0.1600: udp 46
22:13:18.546665 MY.NET.2.4.4285 > MY.NET.1.0.1600: udp 46

This scan repeated itself any number of times on my networks. The source machine ran through its ports in order, always broadcasting (in the older, BSD-style way, at the network address) to destination port 1600. Since I am fairly new to this particular job and therefore unfamiliar with these networks, I didn't recognize the source machine address offhand, although I discovered it was a legitimate node on my network. It is an older medical imaging device manufactured by General Electric. Recently, it has been moved to a new room and turned back on after a long period of downtime.

Targeting: This much is obvious, since the source machine is sending to the network address of the entire MY.NET.1 segment. The nature of the response expected is a mystery but the scan is targeted to my networks, albeit in a primitive way.

Intent: A scan for port 1600 looks suspicious for a couple of reasons. First, there is a trojan called Shivka-Burka that uses this UDP port. I'm seeing only the broadcast traffic on our network (because we're switched) but this could be one side of a two-way communication with a Shivka-Burka trojan. Of course, this is such a loud, obvious scan that the Shivka-Burka scenario seems unlikely but it can't be ruled out. The second point of suspicion is the ISSD service that's registered as the legitimate service user of port 1600. If MY.NET.2.4 is a compromised machine, the cracker could have set up a scan for this service in order to exploit a weakness in it.

Techniques: A typical broadcast mapping attempt, albeit on an unusual port.

History: The network mapping restarted every time this particular medical console was turned back on. Unfortunately, none of the GE technicians had any idea why the machine was doing this. Some of the more experienced medical/technical workers felt that we were seeing a hostile signature. Certainly the machine had been around long enough to get cracked (nine years).

When I came to this department, four Solaris machines had been lost to successful cracks and more were under attack. However, no other old SunOS system had not been cracked during the months of network break-ins and it seemed likely that a network hacker/cracker smart enough to get into one would have reached some of the others, as well. Since the MY.NET.2.4 machine was based on SunOS, it seemed, due to our local, historical reasons, more safe than a Solaris or IRIX based station.

I took a look, via nmap, at the interesting ports left open on MY.NET.2.4. Among them were:

Port	State	Service
104/tcp	open	acr-nema
655/tcp	open	unknown
658/tcp	open	unknown
670/tcp	open	unknown
723/tcp	open	unknown

725/tcp open unknown
727/tcp open unknown
728/tcp open unknown
733/tcp open unknown
736/tcp open unknown
1024/tcp open unknown

So the nmap tool did not detect an open port 1600 on the MY.NET.2.4 machine itself. This was not what I expected and increased my concern, since I felt that this medical console would be configured like all others from GE and would therefore have a service open on port 1600 if it was a legitimate destination port. The port 104 is a medical standard for DICOM data transfer, so that was expected. The unknown ports seem to be specific to this older generation of GE equipment, so they were not a serious cause for alarm. At this point, however, I had to step up my severity estimate.

Severity: Moderate. To understand why, you have to know the full history. Although I did not see any crack attempt (no lethality) and I felt the crude nature of the scan revealed its innocence, I had to increase my judgement of the scan's severity the longer this activity remained unsolved. GE's apparent ignorance of the network software along with the nature of the data (medical and therefore confidential) necessitated further investigation.

Action Recommendation: The first action would be to talk with the manufacturer's representatives and determine whether or not this is expected behavior. A check for rootkits (although often futile) must be done, as should a check for all other signs of intrusion into the GE medical workstation. On the IDS end, it's possible to check to see if the packets were corrected formed or showed any signs of tampering. It's an even better idea to monitor port 1600 activity on all machines in the network and turn off the GE medical workstation if there is any sign of attack through this port.

Resolution: I failed to find any clear signs of a break-in on this machine and had determined that the UDP packets looked legitimate based on this pattern:

16:25:08.759896 ic1.3701 > MY.NET.1.0.1600: udp 46
4500 004a 60ee 0000 3c11 72e1 80e7 d504
80e7 d400 0e75 0640 0036 0000 0000 007e
0000 05e8 4e4d 5231 0000 4943 3000 0000
0000 0000 0000
16:27:08.769361 ic1.3702 > MY.NET.1.0.1600: udp 46
4500 004a 60ef 0000 3c11 72e0 80e7 d504
80e7 d400 0e76 0640 0036 0000 0000 007e
0000 05e8 4e4d 5231 0000 4943 3000 0000
0000 0000 0000

Not only were the packets correctly formed but, as far as UDP was concerned, they were all identical. Above, you can see that three octets increment or decrement regularly. These are all part of the IP datagram. The change from 60ee to 60ef represents the changing ID number of the datagram. The decrement from 72e1 to 72e0 represents the differences in IP header checksum. The change from 0e75 to 0e76 shows the IP destination address incrementing, which we can already see in our earlier trace. I admit I don't understand the IP header checksum business but the behavior seems consistent. All further packets display the same properties:

16:31:08.790067 ic1.3704 > MY.NET.1.0.1600: udp 46
4500 004a 60f1 0000 3c11 72de 80e7 d504
80e7 d400 0e78 0640 0036 0000 0000 007e
0000 05e8 4e4d 5231 0000 4943 3000 0000
0000 0000 0000
16:33:08.799578 ic1.3705 > MY.NET.1.0.1600: udp 46
4500 004a 60f2 0000 3c11 72dd 80e7 d504

80e7 d400 0e79 0640 0036 0000 0000 007e
0000 05e8 4e4d 5231 0000 4943 3000 0000
0000 0000 0000

In the end, my judgement was that this traffic consisted of legitimate broadcast packets. I could find no signs of a break-in on the medical workstation, although I found myself somewhat unfamiliar with what it means to have a normal machine environment in an old GE medical system.

When I mentioned that the GE medical workstation was running NIS services (which looked more strange than suspicious, to me), this helped one of the GE technicians. In fact, the very first GE technician I'd asked about the port scan came back to me after doing a bit of research and some plain-old remembering. He said that port 1600 was used by the old GE AdvantageNet for communication between medical imaging stations as part of the coordination of traffic flow between them. An old Genesis master console not only scans port 1600 for other GE AdvantageNet devices but administer them as an NIS server as well, although data transfers via the DICOM standard take place over port 104, not over any of the GE or NIS ports. Since the MY.NET.2.4 machine had been a master console when it was originally active, it ran the scans and the NIS services.

I still do not know the purpose of all the other GE-specific services but I do expect I will be able to unravel those mysteries.

Historical side note: DICOM image transfers between many old medical systems do not take place over TCP/IP, as would be normal for the new, IP-based transmission standard. DICOM was once its own transmission protocol, not merely a format encapsulated in a TCP header. I wonder if anyone aside from the original vendors has a way to detect traffic signatures from these old devices. I wouldn't be surprised if some hospitals out there have old, misconfigured networks clogged by legacy DICOM traffic they can't properly diagnose due to poor firewalling, filtering, and/or packet detection.

Detect 9

From local tcpdump:

05:22:07.758726 209.235.11.254.64189 > MY.NET.1.8.5556: S
1035542920:10355429
20(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
05:22:07.758886 209.235.11.254.64190 > MY.NET.1.9.5556: S
1035590962:10355909
62(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
05:22:07.759191 209.235.11.254.64193 > MY.NET.1.12.5556: S
1035754830:1035754
830(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
05:22:07.766648 209.235.11.254.64209 > MY.NET.1.28.5556: S
1036686751:1036686
751(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
05:22:07.770915 209.235.11.254.64216 > MY.NET.1.35.5556: S
1037308052:1037308
052(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
05:22:07.776136 209.235.11.254.64228 > MY.NET.1.47.5556: S
1038107792:1038107
792(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
05:22:07.776417 209.235.11.254.64230 > MY.NET.1.49.5556: S
1038256101:1038256
101(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
05:22:07.784767 209.235.11.254.64248 > MY.NET.1.67.5556: S
1039462252:1039462

252(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
05:22:07.784926 209.235.11.254.64249 > MY.NET.1.68.5556: S
1039548593:1039548
593(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
05:22:07.787372 209.235.11.254.64254 > MY.NET.1.73.5556: S
1039851016:1039851
016(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
05:22:07.789122 209.235.11.254.64257 > MY.NET.1.76.5556: S
1039968873:1039968
873(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
05:22:07.789629 209.235.11.254.64260 > MY.NET.1.79.5556: S
1040202613:1040202
613(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
05:22:07.797361 209.235.11.254.64271 > MY.NET.1.90.5556: S
1041054542:1041054
542(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
05:22:07.798137 209.235.11.254.64278 > MY.NET.1.97.5556: S
1041503479:1041503
479(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
05:22:07.814590 209.235.11.254.64305 > MY.NET.1.124.5556: S
1043146852:104314
6852(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
05:22:07.814743 209.235.11.254.64306 > MY.NET.1.125.5556: S
1043209950:104320
9950(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
05:22:07.814890 209.235.11.254.64307 > MY.NET.1.126.5556: S
1043263290:104326
3290(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
05:22:07.816258 209.235.11.254.64313 > MY.NET.1.132.5556: S
1043707584:104370
7584(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
05:22:07.817743 209.235.11.254.64315 > MY.NET.1.134.5556: S
1043817560:104381
7560(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
05:22:07.822359 209.235.11.254.64316 > MY.NET.1.135.5556: S
1043892751:104389
2751(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
05:22:07.824374 209.235.11.254.64318 > MY.NET.1.136.5556: S
1044014500:104401
4500(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
05:22:07.824534 209.235.11.254.64319 > MY.NET.1.137.5556: S
1044102225:104410
2225(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
05:22:07.845709 209.235.11.254.64322 > MY.NET.1.140.5556: S
1044292878:104429
2878(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
05:22:07.846498 209.235.11.254.64321 > MY.NET.1.139.5556: S
1044232180:104423
2180(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
05:22:07.846795 209.235.11.254.64325 > MY.NET.1.143.5556: S
1044525857:104452
5857(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
05:22:07.851628 209.235.11.254.64328 > MY.NET.1.146.5556: S
1044683881:104468
3881(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
05:22:07.853683 209.235.11.254.64330 > MY.NET.1.148.5556: S

1044803580:104480
3580(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
05:22:07.853824 209.235.11.254.64331 > MY.NET.1.149.5556: S
1044849029:104484
9029(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
05:22:07.853987 209.235.11.254.64332 > MY.NET.1.150.5556: S
1044883408:104488
3408(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
05:22:07.854728 209.235.11.254.64336 > MY.NET.1.154.5556: S
1045114886:104511
4886(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
05:22:07.856865 209.235.11.254.64342 > MY.NET.1.160.5556: S
1045471169:104547
1169(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
05:22:07.929856 209.235.11.254.64442 > MY.NET.2.2.5556: S
1051362121:10513621
21(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
05:22:07.932520 209.235.11.254.64446 > MY.NET.2.6.5556: S
1051621416:10516214
16(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
05:22:07.932682 209.235.11.254.64445 > MY.NET.2.5.5556: S
1051548547:10515485
47(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
05:22:07.932845 209.235.11.254.64447 > MY.NET.2.7.5556: S
1051666535:10516665
35(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
05:22:07.934459 209.235.11.254.64448 > MY.NET.2.8.5556: S
1051733406:10517334
06(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
05:22:07.939005 209.235.11.254.64455 > MY.NET.2.15.5556: S
1052140447:1052140
447(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
05:22:07.941896 209.235.11.254.64458 > MY.NET.2.18.5556: S
1052340339:1052340
339(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
05:22:07.942258 209.235.11.254.64460 > MY.NET.2.20.5556: S
1052448173:1052448
173(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
05:22:07.964650 209.235.11.254.64498 > MY.NET.2.57.5556: S
1054624179:1054624
179(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)

Targeting: It's only my network that was targeted, not an individual machine. Probably, other networks in my company were targeted, too, and they already know all about this. No IDS system could miss it. But I thought I'd better bring it up to be safe.

Intent: The attacker is searching for the BackOrifice trojan here and who's to say he or she didn't find one somewhere? We don't have a whole lot of 95/98/NT in the my division, so we're not a great target for this. But we do have a number of other serious vulnerabilities. I keep waiting for a linuxconf (port 98) scan but it hasn't happened yet. Point is, I don't administer a lot of the PCs on my network, so I wouldn't necessarily be in a position to tell if one got cracked.

Technique: There's really nothing unusual about this recon approach, although I'm surprised the person responsible was so blatant about coming from a single machine. Maybe they had a small window of opportunity on a recently-compromised machine.

History: An nslookup on the machine address reveals:

Name: www7.clever.net
Address: 209.235.11.254

which is not very reassuring, actually. Who is clever.net? A look at their website reveals them to be an ISP from Atlanta, GA. Assuming they're operating in good faith and one of their employees isn't doing this, they were cracked. This supports the urgent-scan theory.

Severity: Low-Medium. This is maybe a 2 on a scale of 1-5 because it's easy to counteract and we don't have any systems that are compromised by BackOrifice. I did my own nmap scan to check for port 5556.

Recommendation: Block this machine address for a while at the router. Since it's a single machine, this should be easy enough. Also, we (my company, not necessarily my division) should contact clever.net if we haven't already.

Detect 10

From local tcpdump:

04:05:48.586092 209.235.11.254.51762 > MY.NET.1.8.110: S
819968060:819968060(
0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
04:05:48.588138 209.235.11.254.51766 > MY.NET.1.12.110: S
820282367:820282367
(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
04:05:48.600209 209.235.11.254.51789 > MY.NET.1.35.110: S
821752372:821752372
(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
04:05:48.603326 209.235.11.254.51801 > MY.NET.1.47.110: S
822472977:822472977
(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
04:05:48.603644 209.235.11.254.51803 > MY.NET.1.49.110: S
822599101:822599101
(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
04:05:48.608843 209.235.11.254.51818 > MY.NET.1.64.110: S
823542826:823542826
(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
04:05:48.610636 209.235.11.254.51821 > MY.NET.1.67.110: S
823767682:823767682
(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
04:05:48.637846 209.235.11.254.51880 > MY.NET.1.125.110: S
827477486:82747748
6(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
04:05:48.638015 209.235.11.254.51881 > MY.NET.1.126.110: S
827567727:82756772
7(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
04:05:48.639049 209.235.11.254.51887 > MY.NET.1.132.110: S
827988339:82798833
9(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
04:05:48.640864 209.235.11.254.51889 > MY.NET.1.134.110: S
828078782:82807878
2(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
04:05:48.641005 209.235.11.254.51890 > MY.NET.1.135.110: S
828152614:82815261

4(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
04:05:48.648575 209.235.11.254.51892 > MY.NET.1.137.110: S
828274651:82827465
1(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
04:05:48.650050 209.235.11.254.51894 > MY.NET.1.139.110: S
828412016:82841201
6(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
04:05:48.650184 209.235.11.254.51895 > MY.NET.1.140.110: S
828450387:82845038
7(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
04:05:48.650532 209.235.11.254.51898 > MY.NET.1.143.110: S
828578876:82857887
6(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
04:05:48.651766 209.235.11.254.51901 > MY.NET.1.146.110: S
828793950:82879395
0(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
04:05:48.652072 209.235.11.254.51904 > MY.NET.1.149.110: S
828948924:82894892
4(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
04:05:48.652235 209.235.11.254.51905 > MY.NET.1.150.110: S
829015660:82901566
0(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
04:05:48.653200 209.235.11.254.51909 > MY.NET.1.154.110: S
829221839:82922183
9(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
4:05:48.720487 209.235.11.254.52017 > MY.NET.2.6.110: S
836261610:836261610(
0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
04:05:48.720581 209.235.11.254.52018 > MY.NET.2.7.110: S
836349354:836349354(
0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
04:05:48.720617 209.235.11.254.52019 > MY.NET.2.8.110: S
836437804:836437804(
0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
04:05:48.721814 209.235.11.254.52026 > MY.NET.2.15.110: S
836941866:836941866
(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
04:05:48.722302 209.235.11.254.52029 > MY.NET.2.18.110: S
837122687:837122687
(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
04:05:48.726306 209.235.11.254.52031 > MY.NET.2.20.110: S
837260366:837260366
(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)
04:05:48.738888 209.235.11.254.52068 > MY.NET.2.57.110: S
840011394:840011394
(0) win 8192 <mss 1460,nop,wscale 0,nop,nop,timestamp[|tcp]> (DF)

Targeting: Same as before, this is not a port scan of a particular machine but a network scan of a particular port, specifically the POP3 protocol port. Of course, I have no idea if the rest of my company has been or is being affected.

Intent: The attacker is looking for POP3 mail servers to try some of the canned exploits, most likely. Maybe he or she hopes to find an old version of POP3 without any patches. That's pretty likely, at my company, and even within my division. Until now, I didn't think we were running any POP3 servers. But it turns out we are, mostly because Linux runs the service as a default in many installations. So we may be seeing a direct attack soon.

Technique: Ouch! There is something very distrubing about this technique. Not only is the source machine the same as the other previous big, bold, (rude) scan but the order of our machines scanned is the same as well. Someone is operating different scans from the same systems table. This implies that they've got a systems table for my division, too. Ugh. As a side note, this pretty much eliminates any multiple-hacker scenario. If it's not one person, it's two people sharing the same data.

History: Again, this machine address resolves in the DNS to

Name: www7.clever.net
Address: 209.235.11.254

Combine this with the apparently nearly identical attack scripts used and we now have a steady problem. If this machine isn't blocked at the router, I would expect to see another attack this weekend. Maybe more than one. After all, we have machines vulnerable to POP3 attacks, unlike the BackOrifice trojan scan we saw before.

Severity: High-Medium. This is a serious scan for a vulnerability we may, in fact, have. This has the potential for becoming more serious quickly.

Recommendation: Same as before. Contact clever.net about these intrusions and see if we can get it solved on their end. I will start by calling them, since I see a phone number on their site but no contact email address. Also, I know our vulnerable machines. I will try to shut down the POP3 services on them or make sure they have the latest patches. In fact, as of this writing, I have done so to all the machines I administer directly.

Last Updated: July 23rd, 2017

Upcoming Training

SANS San Antonio 2017	San Antonio, TX	Aug 06, 2017 - Aug 11, 2017	Live Event
SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Adelaide 2017	Adelaide, Australia	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Virginia Beach 2017	Virginia Beach, VA	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS Network Security 2017	Las Vegas, NV	Sep 10, 2017 - Sep 17, 2017	Live Event
SANS vLive - SEC503: Intrusion Detection In-Depth	SEC503 - 201709,	Sep 11, 2017 - Oct 18, 2017	vLive
Baltimore Fall 2017 - SEC503: Intrusion Detection In-Depth	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	vLive
SANS London September 2017	London, United Kingdom	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Baltimore Fall 2017	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	Live Event
Community SANS Scottsdale SEC503	Scottsdale, AZ	Oct 02, 2017 - Oct 07, 2017	Community SANS
SANS October Singapore 2017	Singapore, Singapore	Oct 09, 2017 - Oct 28, 2017	Live Event
Community SANS Ottawa SEC503	Ottawa, ON	Oct 16, 2017 - Oct 21, 2017	Community SANS
SANS Berlin 2017	Berlin, Germany	Oct 23, 2017 - Oct 28, 2017	Live Event
San Diego Fall 2017 - SEC503: Intrusion Detection In-Depth	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	vLive
SANS San Diego 2017	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Seattle 2017	Seattle, WA	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Paris November 2017	Paris, France	Nov 13, 2017 - Nov 18, 2017	Live Event
Community SANS Pensacola SEC503	Pensacola, FL	Nov 27, 2017 - Dec 02, 2017	Community SANS
SIEM & Tactical Analytics Summit & Training	Scottsdale, AZ	Nov 28, 2017 - Dec 05, 2017	Live Event
SANS Cyber Defense Initiative 2017	Washington, DC	Dec 12, 2017 - Dec 19, 2017	Live Event
SANS Security East 2018	New Orleans, LA	Jan 08, 2018 - Jan 13, 2018	Live Event
SANS Las Vegas 2018	Las Vegas, NV	Jan 28, 2018 - Feb 02, 2018	Live Event
SANS Dallas 2018	Dallas, TX	Feb 19, 2018 - Feb 24, 2018	Live Event
SANS OnDemand	Online	Anytime	Self Paced
SANS SelfStudy	Books & MP3s Only	Anytime	Self Paced