Global Information Assurance Certification Paper

*** Northcutt, OK, this kid is a smart cookie, if he is running these in a lab, he is learning the behavior of the attacks, clearly went to a lot of effort., I vote pass. 70 *

SANS

GIAC Level Two Certification - Practical

Intrusion Detection Subject Area: 10 Detects with Analyses

Student: Robert L. Grill,
CISA, CISSP, MBA, CNA

April 10, 2000

Capture 1

Time Src IP Src Port > Dest IP Dest Port: Flag (Synchronization numbers and other TCP attributes ommitted for brevity)

11:57:26.146235 10.22.15.5.139 > 10.22.15.5.139: S
11:57:36.363528 10.22.15.5.139.> 10.22.15.5.139: S
11:57:46.300258 10.22.15.5.139 > 10.22.15.5.139: S

Active Targeting: Yes

History: No history, this was done in a test environment for hands on learning purposes, it worked against a Cisco switch causing a denial of service.

Intent: This is a sample example of a LAND ATTACK. This LAND ATTACK allows attackers to deny service to legitimate users and to administrators for the equipment at the destination IP address. Recovery may require physically visiting the affected hardware.

Source (Tool Used) A program called, latierra.c, which is capable of flooding and of scanning ports and address ranges, was used.

Background, history, additional information about the detect:
The signature of the packet is source port = destination port for a TCP packet with SYN flag set and the port open on target host. The source address must be spoofed (Obviously).

The first Cert advisory on the problem that I saw was on Friday, November 21, 1999, this was also the date that an advisory was posted on Cisco's web site.

Capture 2

Time Src Port Dest (resolved) Protocol Size

12:43:58.431 10.22.15.11.31655 > www.mynetwork.net: **icmp: echo request (frag 4321:380@0+)**
12:43:58.431 10.22.15.11.31655 > www.mynetwork.net: **(frag 4321:380@2656+)**
12:43:58.431 10.22.15.11.31655 > www.mynetwork.net: **(frag 4321:380@3040+)**
12:43:58.431 10.22.15.11.31655 > www.mynetwork.net: **(frag 4321:380@3416+)**
12:43:58.431 10.22.15.11.31655 > www.mynetwork.net: **(frag 4321:380@376+)**
12:43:58.431 10.22.15.11.31655 > www.mynetwork.net: **(frag 4321:380@3800+)**
12:43:58.431 10.22.15.11.31655 > www.mynetwork.net: **(frag 4321:380@4176+)**
12:43:58.431 10.22.15.11.31655 > www.mynetwork.net: **(frag 4321:380@760+)**

Active Targeting: Yes

Intent: Denial of Service - The ping-of-death causes a buffer to overflow on the target host by sending an echo request packet that is larger than the maximum IP packet size of 65535 bytes.

Source (Tool Used): My favorite packet assembly tool is LIBNET was used with some C code to send the actual packets.

Background, history, additional information about the detect:
Attacker sends a ping packet that is larger than the maximum IP packet size of 65535 bytes. The TCP/IP specification (the basis for many protocols used on the Internet) allows for a maximum packet size of up to 65536 octets (1 octet = 8 bits of data), containing a minimum of 20 octets of IP header information and 0 or more octets of optional information, with the rest of the packet being data. It is known that some systems will react in an unpredictable fashion when receiving oversized IP packets. Reports indicate a range of reactions including crashing, freezing, and rebooting. In particular, the reports received by the CERT Coordination Center indicate that Internet Control Message Protocol (ICMP) packets issued via the "ping" command have been used to trigger this behavior. ICMP is a subset of the TCP/IP suite of protocols that transmits error and control messages between systems. Two specific instances of the ICMP are the ICMP ECHO_REQUEST and ICMP ECHO_RESPONSE datagrams. These two instances can be used by a local host to determine whether a remotesystem is reachable via the network; this is commonly achieved using the "ping" command. Discussion in public forums has centered around the use of the "ping" command to construct oversized ICMP datagrams (which are encapsulated within an IP packet). Many ping implementations by default send ICMP datagrams consisting only of the 8 octets of ICMP header information but allow the user to specify a larger packet size if desired. This attack was first document in CERT CA 96.26

Capture 3

Time: Src Port: Dest Logical: Dest Port: Protocol : ICMP Message

```
00:43:58.094644 10.168.20.20 > 10.168.64.255: icmp: echo request
00:43:58.604889 10.168.20.20 > 10.168.64.0: icmp: echo request
00:50:02.297035 10.168.20.20 > 10.168.65.255: icmp: echo request
00:50:02.689911 10.168.20.20 > 10.168.65.0: icmp: echo request
00:54:56.911891 10.168.20.20 > 10.168.66.255: icmp: echo request
00:54:57.265833 10.168.20.20 > 10.168.66.0: icmp: echo request
00:59:52.822243 10.168.20.20 > 10.168.67.255: icmp: echo request
00:59:53.415182 10.168.20.20 > 10.168.67.0: icmp: echo request
```

Active Targeting: YES (subnet 168 was my target)

Intent: Port Mapping used for reconnaissance used for enumeration of a target network. – The scan above sends a broadcast PING packet to the entire 10.168 subnet .

Source (Tool Used) – Nmap is a common port mapping tool, probably the best one out there.

Background, history, additional information about the detect:

Ping (Packet InterNet Groper) is standard network diagnostic tool that is included in most modern operating systems and routers. Sending an echo request packet to see if a host is alive is the appropriate use of the tool, however this probing behavior can be indicative of a pre-attack probe, or larger network mapping effort. Many implementations of the ping program create unique echo-request packets, which is how we can tell what host and program to attack. The scan above is also known as a ping sweep.

The broadcasts were sent with some time in between to so the system would not be overloaded.

Capture 4

Time Src Logical Src Port Dest Port

Jan 29 23:44:08.536110 206.172.251.126,3319 -> 10.0.0.1,1243 PR tcp len 20 48 -S
Jan 29 23:44:08.571091 206.172.251.126,3320 -> 10.0.0.2,1243 PR tcp len 20 48 -S
Jan 29 23:44:08.602512 206.172.251.126,3321 -> 10.0.0.3,1243 PR tcp len 20 48 -S

Active Targeting: Yes (looking for a particular trojan)

History: This is a detect from the GIAC web site.

Intent: This scan is trolling for a tojan on port 1243. The Trojan is known as BackDoor-G, SubSeven or Apocalypse.

Source (Tool Used): N/A

Background, history, additional information about the detect:

This trojan is a "remote administration tool" that allows an attacker to take complete control over the victims server. Client desktop machinesin Window 9x/NT environments are most likely to suffer from this trojan infection. The Trojan is usually installed by disguise in an email attachment, or hidden in other software available for download.

A whois query indicated that the registrant is Bell Sygma 160 Elgin St. Flr. 12 Ottawa, Ontario K1G 3J4 CA, this could have been forged. A Traceroute indicated that the route to the source address remianed with in the USA.

Capture 5

Time	Src	Port	Dest	Port	Protocol	Message Length
14:39:43.920006	172.20.20.1	31790	> 192.168.1.3	31789:	udp	1
14:39:43.922788	172.20.20.1	31790	> 192.168.1.4	31789:	udp	1

Active Targeting: Yes (looking for a particular trojan)

History: This is a detect from the GIAC web site.

Intent: According to GIAC port 31789 is a Trojan called Hack'a Tack

Source (Tool Used): N/A

Background, history, additional information about the detect:

The server portion of Hack a Tack is named "expl32.exe" (236KB 5/16/99 2:49PM) and it will be found in the WINDOW$ directory.
TCP ports 31785, 31787 and UDP ports 31789, 31791 (by default) are used to establish the connection between the "client" and "server".
Once installed, it is rerun every time the computer is started by means of an entry under the
"HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" branch in the Window$ Registry. Hack 'a' Tack currently affects Windows 95/98 PC's.

The source address is from a private network and so is the destination.

Capture 6

Time Src Dest….Port Flag Sequence #'s (The remainder TCP stuff removed for brevity)

01:56:58.62 bob.bob.**0** > 192.168.93.0.**53**: **SF** 2216558592:2216558592
01:56:58.63 bob.bob.**0** > 192.168.93.1.**53**: **SF** 2216558592:2216558592
01:56:58.65 bob.bob.**0** > 192.168.93.2.**53**: **SF** 2216558592:2216558592
01:56:58.67 bob.bob.**0** > 192.168.93.3.**53**: **SF** 2216558592:2216558592
01:56:58.69 bob.bob.**0** > 192.168.93.4.**53**: **SF** 2216558592:2216558592
01:56:58.71 bob.bob.**0** > 192.168.93.5.**53**: **SF** 2216558592:2216558592
01:56:58.73 bob.bob.**0** > 192.168.93.6.**53**: **SF** 2216558592:2216558592

Active Targeting: Yes, this is a scan of a private network.

History: This is from my test network.

Intent: With a SF scan a potential network attacker performs reconnaissance with the intent of finding hosts on a network without being detected.

Source (Tool Used): This Syn Fin scan was sent out over the wire using a Pearl script that I am experimenting with. It was originally written by a friend of mine.

Background, history, additional information about the detect:

It is simple to set up a router to filter out packets with anomalous TCP flags. Hosts sent a SYN FIN flag get confused and respond with a Reset if the port is listening. The destination host and its port is the service you want to perform reconnaissance on. Once a weakness is identified than an attack can begin.

Capture 7

Time	Src Port	Dest Port	Details
08:08:16.155354	10.168.20.20 >	10.169.203.17.	**chargen**: udp
08:21:48.891451	10.168.20.20 >	10.169.14.50.	**chargen**: udp
08:25:12.968929	10.168.20.20 >	10.169.102.3.	**chargen**: udp
08:42:22.605428	10.168.20.20 >	10.169.18.28.	**chargen**: udp
08:47:21.450708	10.168.20.20 >	10.169.130.93.	**chargen**: udp
08:51:27.491458	10.168.20.20 >	10.169.153.78.	**chargen**: udp

Active Targeting: Yes – A denial of service attack against a certain network.

History: This was done in a test lab.

Intent: The intent is a denial of service. Client connects to the port, the server spits characters back. Chargen spits back characters in an endless stream until the connection is closed.

UDP Port number 19 is chargen.

Source (Tool Used): My favorite packet assembly tool is LIBNET was used with some C code to send the actual packets.
.

Background, history, additional information about the detect:

When a connection is established between two UDP services, each of which produces output, these two services can produce a very high number of packets that can lead to a denial of service on the machine(s) where the services are offered. Anyone with network connectivity can launch an attack; no account access is needed.

For example, by connecting a host's chargen service to the echo service on the same or another machine, all affected machines may be effectively taken out of service because of the excessively high number of packets produced. In addition, if two or more hosts are so connected, the intervening network may also become congested and deny service to all hosts whose traffic traverses that network.

Capture 8

Time Src Src Port Dest Port Protocol Length

01:47:07.597435 10.10.66.6.10431 > 192.168.1.140.31337: udp 19
01:47:07.655861 10.10.66.6.10431 > 192.168.1.141.31337: udp 19
01:47:08.418523 10.10.66.6.10431 > 192.168.1.177.31337: udp 19

Active Targeting: Yes – This was a search for Back Oriface.

History: This was done in a test lab.

Intent: The source computer is scanning the network for UDP port 31337, which is most commonly related to Back Orifice a "remote administration" tool.

Source (Tool Used): This scan was performed with Nmap.

Background, history, additional information about the detect:

This signature matches the known default port of the trojan Back Oriface. It is possible the trojan could be configured to use an alternate port. This scan does not necessarily represent a major hazard. Back Oraface is a well know Trojan commonly removed by virus detectors.

Capture 9 TCP Scan from a Router

Time	Src	Src Port	Dest Port	Details

```
Jan 12 04:22:04: Packet log: input DENY eth0 PROTO=6 24.7.160.80:4326 192.168.2.4:143
L=60 S=0x00 I=17311 F=0x4000 T=49
Jan 12 04:22:10: Packet log: input DENY eth0 PROTO=6 24.7.160.80:4326 192.168.2.4:143
L=60 S=0x00 I=17847 F=0x4000 T=50
Jan 12 15:32:08: Packet log: input DENY eth0 PROTO=6 24.7.160.80:4329 192.168.2.1:143
L=60 S=0x00 I=17314 F=0x4000 T=50 SYN (#64)
Jan 12 15:32:08: Packet log: input DENY eth0 PROTO=6 24.7.160.80:4329 192.168.2.1:143
L=60 S=0x00 I=17850 F=0x4000 T=50 SYN (#64)
```

Active Targeting: Yes – This detect is a target network for an Imap vulnerability.

History: This scan was taken from the Giac Web site.

Intent: Gain remote control over a victim's computer through a vulnerability in the Imap application on port 143.

Remote intruders can execute arbitrary commands under the privileges of the process running the vulnerable IMAP server. If the vulnerable IMAP server is running as root, remote intruders can gain root access. The standard method for invoking this attack is to run the output of the compiled binary through NetCat, which after successful completion of the attack will leave the attacker with a root shell.

Source (Tool Used): Net Cat - a simple utility which reads and writes data across network connections, using TCP or UDP protocol can be used to perform the buffer overload. Net Cat is available at the L0ft.

Background, history, additional information about the detect:

The Imap vulnerability is unique to Linux machines `taking advantage of a buffer overrun condition in the LOGIN command.`

IMAP stands for **I**nternet **M**essage **A**ccess **P**rotocol. It is a method of accessing electronic mail or bulletin board messages that are kept on a (possibly shared) mail server. In other words, it permits a "client" email program to access remote message stores as if they were local. For example, email stored on an IMAP server can be manipulated from a desktop computer at home, a workstation at the office, and a notebook computer while traveling, **without** the need to transfer messages or files back and forth between these computers.

The Registrant of the source IP is Home Network 425 Broadway Redwood City, CA 94063 US

Capture 10

Time Src Port Dest Dest Port Protocol Size Flag

```
Feb 18 07:50:47.675559 209.88.177.67,4484 -> 10.0.0.8,23 PR tcp len 20 60 -S
Feb 18 07:50:47.676150 209.88.177.67,4483 -> 10.0.0.3,23 PR tcp len 20 60 -S
Feb 18 07:50:50.649197 209.88.177.67,4483 -> 10.0.0.3,23 PR tcp len 20 60 -S
Feb 18 07:50:50.651535 209.88.177.67,4484 -> 10.0.0.8,23 PR tcp len 20 60 -S
```

Active Targeting: Yes – This scan is to a specific machine. Scans on a specific machine implies that the reconnaissance

History: This scan was taken from the Giac website.

Intent: The intent of this scan is reconnaissance. Port 23 is Telnet, the potential attacker is looking to log in remotely to the Telnet application.

Source (Tool Used): N/A

Background, history, additional information about the detect:

I noted that the above scan has a origination with a public address and a destination in a private network. My question was, how did the intruder get past the Network Address Translator (NAT) ? After some research into the workings of a NAT, I realized that a NAT works by address and port address translation of the source address of outgoing packets. This attacker was looking for responses from a private network as a result of stimulus to the proxy. (Cool) The reconnaissance packets are using a SYN flag. The destination host will either respond with a SYNACK to this stimulus if the port is listening or a port unreachable message if it is not listening.

The scan was coming from Moti Digestaniaddress: CSTS Technical Servicesaddress: 25St Geulaaddress: Haifa 33197 Israel.

Last Updated: July 27th, 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 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