# Global Information Assurance Certification Paper

*** Northcutt, I gave David specific permission to use this formula, the score will reflect the work required to create this lab and the value these traces have to the community. 90 ***

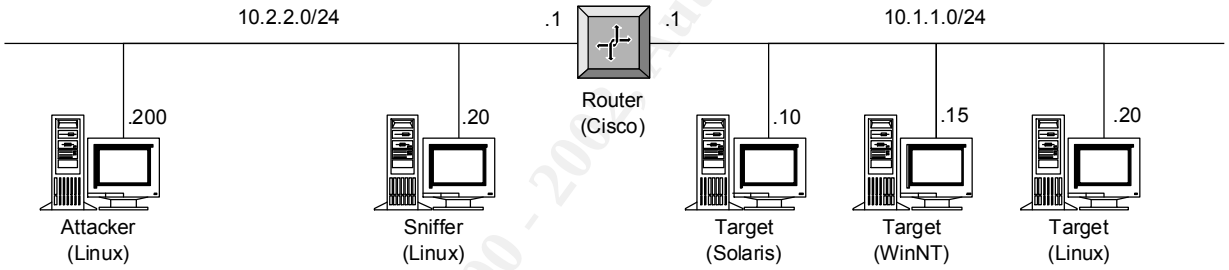


# GCIA Practical Exam
# SANS 2000 IDIC Curriculum

**David L. Wagner**
**April 4, 2000**

## BACKGROUND

The detects displayed on the following pages were captured using TCPDUMP on a test network configured per the following diagram:




The attacking system utilized NMAP and various freely available hacking tools to generate the attack packets.

## DETECT 1 – NETWORK MAPPING

```
17:56:33.086634 eth0 P 10.2.2.200 > 10.1.1.1: icmp: echo request
17:56:33.086696 eth0 P 10.2.2.200 > 10.1.1.2: icmp: echo request
17:56:33.086764 eth0 P 10.2.2.200 > 10.1.1.3: icmp: echo request
17:56:33.086833 eth0 P 10.2.2.200 > 10.1.1.4: icmp: echo request
17:56:33.086902 eth0 P 10.2.2.200 > 10.1.1.5: icmp: echo request
17:56:33.086972 eth0 P 10.2.2.200 > 10.1.1.6: icmp: echo request
17:56:33.087038 eth0 P 10.2.2.200 > 10.1.1.7: icmp: echo request
17:56:33.087107 eth0 P 10.2.2.200 > 10.1.1.8: icmp: echo request
17:56:33.087175 eth0 P 10.2.2.200 > 10.1.1.9: icmp: echo request
17:56:33.087244 eth0 P 10.2.2.200 > 10.1.1.10: icmp: echo request
17:56:33.087311 eth0 P 10.2.2.200 > 10.1.1.11: icmp: echo request
17:56:33.087380 eth0 P 10.2.2.200 > 10.1.1.12: icmp: echo request
17:56:33.087448 eth0 P 10.2.2.200 > 10.1.1.13: icmp: echo request
17:56:33.087516 eth0 P 10.2.2.200 > 10.1.1.14: icmp: echo request
17:56:33.087585 eth0 P 10.2.2.200 > 10.1.1.15: icmp: echo request
17:56:33.087654 eth0 P 10.2.2.200 > 10.1.1.16: icmp: echo request
17:56:33.087722 eth0 P 10.2.2.200 > 10.1.1.17: icmp: echo request
17:56:33.087790 eth0 P 10.2.2.200 > 10.1.1.18: icmp: echo request
17:56:33.087859 eth0 P 10.2.2.200 > 10.1.1.19: icmp: echo request
17:56:33.087927 eth0 P 10.2.2.200 > 10.1.1.20: icmp: echo request
17:56:33.092169 eth0 P 10.1.1.10 > 10.2.2.200: icmp: echo reply (DF)
17:56:33.092358 eth0 P 10.1.1.15 > 10.2.2.200: icmp: echo reply
17:56:33.092528 eth0 P 10.1.1.20 > 10.2.2.200: icmp: echo reply
17:56:33.094730 eth0 P 10.1.1.1 > 10.2.2.200: icmp: echo reply
17:56:39.086448 eth0 P 10.2.2.200 > 10.1.1.2: icmp: echo request
17:56:39.086516 eth0 P 10.2.2.200 > 10.1.1.3: icmp: echo request
17:56:39.086584 eth0 P 10.2.2.200 > 10.1.1.4: icmp: echo request
17:56:39.086652 eth0 P 10.2.2.200 > 10.1.1.5: icmp: echo request
17:56:39.086721 eth0 P 10.2.2.200 > 10.1.1.6: icmp: echo request
17:56:39.086789 eth0 P 10.2.2.200 > 10.1.1.7: icmp: echo request
17:56:39.086858 eth0 P 10.2.2.200 > 10.1.1.8: icmp: echo request
17:56:39.086926 eth0 P 10.2.2.200 > 10.1.1.9: icmp: echo request
17:56:39.086995 eth0 P 10.2.2.200 > 10.1.1.11: icmp: echo request
17:56:39.087063 eth0 P 10.2.2.200 > 10.1.1.12: icmp: echo request
17:56:39.087132 eth0 P 10.2.2.200 > 10.1.1.13: icmp: echo request
17:56:39.087200 eth0 P 10.2.2.200 > 10.1.1.14: icmp: echo request
17:56:39.087269 eth0 P 10.2.2.200 > 10.1.1.16: icmp: echo request
17:56:39.087336 eth0 P 10.2.2.200 > 10.1.1.17: icmp: echo request
17:56:39.087405 eth0 P 10.2.2.200 > 10.1.1.18: icmp: echo request
17:56:39.087473 eth0 P 10.2.2.200 > 10.1.1.19: icmp: echo request
```

This capture depicts what appears to be an attacker attempting to map active hosts on the target network. Notice that each IP address is pinged sequentially using ICMP echo requests in a relatively short amount of time. For this scan to work the source IP address must not be spoofed so that the attacker is able to receive echo reply messages. From this scan the attacker is able to learn that hosts 10.1.1.1, 10.1.1.10, 10.1.1.15 and 10.1.1.20 are currently active. A more efficient method of this scan would be for the hacker to ping the network and/or broadcast address(es) of the target network. Using the network and/or broadcast address(es) requires that edge router of the target network not block directed broadcasts. Disabling directed broadcasts on router interfaces helps prevent networks from being used as intermediaries in DOS attacks such as Smurf.

## DETECT 2 – TCP PORT SCAN (SYN)

```
18:05:35.189182 eth0 P 10.2.2.200.34104 > 10.1.1.1.netbios-ssn: S 1942359554:1942359554(0) win 1024
18:05:35.189270 eth0 P 10.2.2.200.34104 > 10.1.1.1.www: S 1942359554:1942359554(0) win 1024
18:05:35.191864 eth0 P 10.1.1.1.netbios-ssn > 10.2.2.200.34104: R 0:0(0) ack 1942359555 win 0
18:05:35.194159 eth0 P 10.1.1.1.www > 10.2.2.200.34104: R 0:0(0) ack 1942359555 win 0
18:05:35.198262 eth0 P 10.2.2.200.34104 > 10.1.1.10.netbios-ssn: S 1548192060:1548192060(0) win 1024
18:05:35.198335 eth0 P 10.2.2.200.34104 > 10.1.1.10.www: S 1548192060:1548192060(0) win 1024
18:05:35.199151 eth0 P 10.1.1.10.netbios-ssn > 10.2.2.200.34104: R 0:0(0) ack 1548192061 win 0 (DF)
18:05:35.199338 eth0 P 10.1.1.10.www > 10.2.2.200.34104: S 2294091210:2294091210(0) ack 1548192061 win 9112 <mss 536> (DF)
18:05:35.199564 eth0 P 10.2.2.200.34104 > 10.1.1.10.www: R 1548192061:1548192061(0) win 0
18:05:35.252161 eth0 P 10.2.2.200.34104 > 10.1.1.15.netbios-ssn: S 4272728633:4272728633(0) win 1024
18:05:35.252235 eth0 P 10.2.2.200.34104 > 10.1.1.15.www: S 4272728633:4272728633(0) win 1024
18:05:35.253046 eth0 P 10.1.1.15.netbios-ssn > 10.2.2.200.34104: S 61562:61562(0) ack 4272728634 win 8576 <mss 1460> (DF)
18:05:35.253232 eth0 P 10.1.1.15.www > 10.2.2.200.34104: R 0:0(0) ack 4272728634 win 0
18:05:35.253301 eth0 P 10.2.2.200.34104 > 10.1.1.15.netbios-ssn: R 4272728634:4272728634(0) win 0
18:05:35.262115 eth0 P 10.2.2.200.34104 > 10.1.1.20.netbios-ssn: S 3957223622:3957223622(0) win 1024
18:05:35.262185 eth0 P 10.2.2.200.34104 > 10.1.1.20.www: S 3957223622:3957223622(0) win 1024
18:05:35.262999 eth0 P 10.1.1.20.netbios-ssn > 10.2.2.200.34104: R 0:0(0) ack 3957223623 win 0
18:05:35.263182 eth0 P 10.1.1.20.www > 10.2.2.200.34104: R 0:0(0) ack 3957223623 win 0
```

This capture shows the attacker performing a TCP Port Scan using SYN packets. Notice that the attacker kills the connection of any ports that respond with a SYN-ACK using a RESET packet. Also, notice that the source port remains the same throughout the scan. This scan utilizes a method known as Half-Open Scanning due to the three-way handshake never being completed. This method relies on the fact that closed ports respond to a SYN request with a RESET-ACK and open ports respond with a SYN-ACK. From this capture it appears the attacker is attempting to locate specific hosts on the target network with open TCP ports 80 (WWW) and 139 (NetBIOS). The results of this scan provide the attacker with the ability to determine that 10.1.1.10 has port 80 open and 10.1.1.15 has port 139 open due to their SYN-ACK responses.

## DETECT 3 – TCP PORT SCAN (FIN)

```
18:25:49.137430 eth0 P 10.2.2.200.44335 > 10.1.1.10.smtp: F 0:0(0) win 2048
18:25:49.137505 eth0 P 10.2.2.200.44335 > 10.1.1.10.telnet: F 0:0(0) win 2048
18:25:49.137575 eth0 P 10.2.2.200.44335 > 10.1.1.10.gopher: F 0:0(0) win 2048
18:25:49.137642 eth0 P 10.2.2.200.44335 > 10.1.1.10.www: F 0:0(0) win 2048
18:25:49.137710 eth0 P 10.2.2.200.44335 > 10.1.1.10.finger: F 0:0(0) win 2048
18:25:49.137779 eth0 P 10.2.2.200.44335 > 10.1.1.10.sunrpc: F 0:0(0) win 2048
18:25:49.137848 eth0 P 10.2.2.200.44335 > 10.1.1.10.ftp: F 0:0(0) win 2048
18:25:49.137915 eth0 P 10.2.2.200.44335 > 10.1.1.10.time: F 0:0(0) win 2048
18:25:49.141826 eth0 P 10.1.1.10.gopher > 10.2.2.200.44335: R 0:0(0) ack 0 win 0 (DF)
18:25:49.451717 eth0 P 10.2.2.200.44336 > 10.1.1.10.smtp: F 0:0(0) win 2048
18:25:49.451789 eth0 P 10.2.2.200.44336 > 10.1.1.10.telnet: F 0:0(0) win 2048
18:25:49.451857 eth0 P 10.2.2.200.44336 > 10.1.1.10.www: F 0:0(0) win 2048
18:25:49.451925 eth0 P 10.2.2.200.44336 > 10.1.1.10.finger: F 0:0(0) win 2048
18:25:49.451994 eth0 P 10.2.2.200.44336 > 10.1.1.10.sunrpc: F 0:0(0) win 2048
18:25:49.452061 eth0 P 10.2.2.200.44336 > 10.1.1.10.ftp: F 0:0(0) win 2048
18:25:49.452130 eth0 P 10.2.2.200.44336 > 10.1.1.10.time: F 0:0(0) win 2048
18:25:49.771822 eth0 P 10.2.2.200.44335 > 10.1.1.10.time: F 0:0(0) win 2048
18:25:49.771890 eth0 P 10.2.2.200.44335 > 10.1.1.10.ftp: F 0:0(0) win 2048
18:25:49.771959 eth0 P 10.2.2.200.44335 > 10.1.1.10.sunrpc: F 0:0(0) win 2048
18:25:49.772027 eth0 P 10.2.2.200.44335 > 10.1.1.10.finger: F 0:0(0) win 2048
18:25:49.772096 eth0 P 10.2.2.200.44335 > 10.1.1.10.www: F 0:0(0) win 2048
18:25:49.772164 eth0 P 10.2.2.200.44335 > 10.1.1.10.telnet: F 0:0(0) win 2048
18:25:49.772232 eth0 P 10.2.2.200.44335 > 10.1.1.10.smtp: F 0:0(0) win 2048
18:25:50.091770 eth0 P 10.2.2.200.44336 > 10.1.1.10.time: F 0:0(0) win 2048
18:25:50.091838 eth0 P 10.2.2.200.44336 > 10.1.1.10.ftp: F 0:0(0) win 2048
18:25:50.091907 eth0 P 10.2.2.200.44336 > 10.1.1.10.sunrpc: F 0:0(0) win 2048
18:25:50.091975 eth0 P 10.2.2.200.44336 > 10.1.1.10.finger: F 0:0(0) win 2048
18:25:50.092043 eth0 P 10.2.2.200.44336 > 10.1.1.10.www: F 0:0(0) win 2048
18:25:50.092112 eth0 P 10.2.2.200.44336 > 10.1.1.10.telnet: F 0:0(0) win 2048
18:25:50.092180 eth0 P 10.2.2.200.44336 > 10.1.1.10.smtp: F 0:0(0) win 2048
```

This capture shows the attacker attempting another TCP Port Scan. Take note of the attacker repeatedly sending packets with only the FIN flag set from the same source port to multiple destination ports on the target. Using the FIN Scan, the attacker is able to detect closed ports on the target by listening for RESET-ACK response packets. According to RFC 793, closed ports should respond to FIN only requests with a RESET-ACK and open ports should discard the request. In this scan it appears the attacker is attempting to find specific open ports on an active target. Due to the target only responding with a RESET-ACK on port 70 (gopher), the attacker is able to determine that all of the scanned ports are open except port 70.

## DETECT 4 – UDP PORT SCAN

```
18:30:13.789365 eth0 P 10.2.2.200.43562 > 10.1.1.1.31337: udp 0
18:30:13.791706 eth0 P 10.2.2.1 > 10.2.2.200: icmp: 10.1.1.1 udp port 31337 unreachable
18:30:14.302018 eth0 P 10.2.2.200.43562 > 10.1.1.10.31337: udp 0
18:30:14.302722 eth0 P 10.1.1.10 > 10.2.2.200: icmp: 10.1.1.10 udp port 31337 unreachable (DF)
18:30:14.811930 eth0 P 10.2.2.200.43562 > 10.1.1.15.31337: udp 0
18:30:14.812594 eth0 P 10.1.1.15 > 10.2.2.200: icmp: 10.1.1.15 udp port 31337 unreachable (DF)
18:30:15.322018 eth0 P 10.2.2.200.43562 > 10.1.1.20.31337: udp 0
18:30:15.322755 eth0 P 10.1.1.20 > 10.2.2.200: icmp: 10.1.1.20 udp port 31337 unreachable [tos 0xc0]
```

This capture shows the attacker attempting a UDP Port Scan on selected hosts on a target network. Closed UDP ports respond with ICMP 'unreachable' messages while open ports do not respond. Unlike this capture, multiple UDP packets are usually sent to each host during a UDP scan due to UDP's unreliability. This helps ensure that UDP packets lost in transit don't cause false positives. Notice the scanning of UDP port 31337. The use of this destination port on multiple machines in a short duration of time suggests that the hacker is attempting to locate hosts with Back Orifice installed but this cannot be considered absolute as port 31337 is a valid high-port and could be used by any application. Due to the ICMP 'unreachable' responses displayed in the capture, none of the scanned hosts have this port open.

## DETECT 5 – TCP PORT SCAN (FRAGMENTED)

```
18:46:45.431666 eth0 P truncated-tcp 16 (frag 52561:16@0+)
18:46:45.431730 eth0 P 10.2.2.200 > 10.1.1.15: (frag 52561:4@16)
18:46:45.436125 eth0 P 10.1.1.15.ftp > 10.2.2.200.59200: S 61649:61649(0) ack 1512150924 win 8576 <mss 1460> (DF)
18:46:45.436376 eth0 P 10.2.2.200.59200 > 10.1.1.15.ftp: R 1512150924:1512150924(0) win 0
```

This capture contains packets that appear to be a Fragmented TCP SYN Scan. Note the size of the initial packet in this exchange is less than the 20 bytes required for the TCP header. By fragmenting the initial packet in the scan it makes it more difficult for packet filters and IDS systems to detect. Using this scan method the attacker was able to detect an open FTP port on the target machine.

## DETECT 6 – PING-OF-DEATH

```
19:01:26.540838 eth0 P 10.1.1.13 > 10.1.1.10: icmp: echo request (frag 4321:380@0+)
19:01:26.545463 eth0 P 10.1.1.13 > 10.1.1.10: (frag 4321:380@376+)
19:01:26.549912 eth0 P 10.1.1.13 > 10.1.1.10: (frag 4321:380@760+)
19:01:26.554150 eth0 P 10.1.1.13 > 10.1.1.10: (frag 4321:380@1136+)
19:01:26.558606 eth0 P 10.1.1.13 > 10.1.1.10: (frag 4321:380@1520+)
19:01:26.563061 eth0 P 10.1.1.13 > 10.1.1.10: (frag 4321:380@1896+)
19:01:26.567553 eth0 P 10.1.1.13 > 10.1.1.10: (frag 4321:380@2280+)
19:01:26.571587 eth0 P 10.1.1.13 > 10.1.1.10: (frag 4321:380@2656+)
. . .
19:01:26.711181 eth0 P 10.1.1.13 > 10.1.1.10: (frag 4321:380@62696+)
19:01:26.711533 eth0 P 10.1.1.13 > 10.1.1.10: (frag 4321:380@63080+)
19:01:26.711886 eth0 P 10.1.1.13 > 10.1.1.10: (frag 4321:380@63456+)
19:01:26.712239 eth0 P 10.1.1.13 > 10.1.1.10: (frag 4321:380@63840+)
19:01:26.712591 eth0 P 10.1.1.13 > 10.1.1.10: (frag 4321:380@64216+)
19:01:26.712943 eth0 P 10.1.1.13 > 10.1.1.10: (frag 4321:380@64600+)
19:01:26.713298 eth0 P 10.1.1.13 > 10.1.1.10: (frag 4321:380@64976+)
19:01:26.713648 eth0 P 10.1.1.13 > 10.1.1.10: (frag 4321:380@65360)
```

This capture shows what is commonly known as the Ping-Of-Death. Note the maximum size of the assembled fragments is greater than the max IP packet size of 65535. Also, note that since these packets were captured outside the 10.1.1.0 network the use of 10.1.1.13 for the source address indicates that the source was spoofed. In this scenario, the attacker hopes to create a buffer-overflow on the target by sending this oversized packet. If the buffer-overflow is successful it could create a denial-of-service for clients attempting to connect to the target. Since the creation of oversized ping packets has been disabled in many operating systems, Ping-Of-Death packets may be hand-crafted or generated by many canned hacking tools.

## DETECT 7 – TEARDROP ATTACK

```
19:27:09.916389 eth0 P 10.1.1.13.38253 > 10.1.1.10.62937: udp 28 (frag 242:36@0+)
19:27:09.920968 eth0 P 10.1.1.13 > 10.1.1.10: (frag 242:4@24)
19:27:09.944828 eth0 P 10.1.1.13.38253 > 10.1.1.10.62937: udp 28 (frag 242:36@0+)
19:27:09.948756 eth0 P 10.1.1.13 > 10.1.1.10: (frag 242:4@24)
19:27:09.964798 eth0 P 10.1.1.13.38253 > 10.1.1.10.62937: udp 28 (frag 242:36@0+)
19:27:09.969234 eth0 P 10.1.1.13 > 10.1.1.10: (frag 242:4@24)
19:27:09.985353 eth0 P 10.1.1.13.38253 > 10.1.1.10.62937: udp 28 (frag 242:36@0+)
19:27:09.989831 eth0 P 10.1.1.13 > 10.1.1.10: (frag 242:4@24)
19:27:10.014801 eth0 P 10.1.1.13.38253 > 10.1.1.10.62937: udp 28 (frag 242:36@0+)
19:27:10.019229 eth0 P 10.1.1.13 > 10.1.1.10: (frag 242:4@24)
```

This capture depicts what is commonly known as the Teardrop Attack. Note the use of fragmented packets. Also, note that the fragmented packets overlap when they are reassembled and that the second fragment's offset of 24 is less than the length of the first packet in the fragment. Since these packets were captured outside the 10.1.1.0 network the use of 10.1.1.13 for the source address indicates that the source was spoofed. This attack is generally used against older Linux machines that were unable to handle the resulting negative offset caused by the re-assembly of the fragments and would crash.

## DETECT 8 – TCP SYN FLOODING

```
19:38:41.425229 eth0 P 10.1.1.13.1978 > 10.1.1.10.www: S 674719801:674719801(0) win 65535
19:38:41.444102 eth0 P 10.1.1.13.1825 > 10.1.1.10.www: S 674719801:674719801(0) win 65535
19:38:41.463609 eth0 P 10.1.1.13.2194 > 10.1.1.10.www: S 674719801:674719801(0) win 65535
19:38:41.483616 eth0 P 10.1.1.13.1754 > 10.1.1.10.www: S 674719801:674719801(0) win 65535
19:38:41.503634 eth0 P 10.1.1.13.1407 > 10.1.1.10.www: S 674719801:674719801(0) win 65535
19:38:41.523615 eth0 P 10.1.1.13.1854 > 10.1.1.10.www: S 674719801:674719801(0) win 65535
19:38:41.543617 eth0 P 10.1.1.13.1598 > 10.1.1.10.www: S 674719801:674719801(0) win 65535
19:38:41.563617 eth0 P 10.1.1.13.2124 > 10.1.1.10.www: S 674719801:674719801(0) win 65535
19:38:41.583620 eth0 P 10.1.1.13.1021 > 10.1.1.10.www: S 674719801:674719801(0) win 65535
19:38:41.603624 eth0 P 10.1.1.13.1557 > 10.1.1.10.www: S 674719801:674719801(0) win 65535
19:38:41.623627 eth0 P 10.1.1.13.1769 > 10.1.1.10.www: S 674719801:674719801(0) win 65535
19:38:41.643629 eth0 P 10.1.1.13.1967 > 10.1.1.10.www: S 674719801:674719801(0) win 65535
19:38:41.663632 eth0 P 10.1.1.13.1859 > 10.1.1.10.www: S 674719801:674719801(0) win 65535
19:38:41.683633 eth0 P 10.1.1.13.2287 > 10.1.1.10.www: S 674719801:674719801(0) win 65535
19:38:41.703635 eth0 P 10.1.1.13.1750 > 10.1.1.10.www: S 674719801:674719801(0) win 65535
19:38:41.723640 eth0 P 10.1.1.13.2104 > 10.1.1.10.www: S 674719801:674719801(0) win 65535
19:38:41.743641 eth0 P 10.1.1.13.1907 > 10.1.1.10.www: S 674719801:674719801(0) win 65535
19:38:41.763647 eth0 P 10.1.1.13.1342 > 10.1.1.10.www: S 674719801:674719801(0) win 65535
19:38:41.783684 eth0 P 10.1.1.13.2202 > 10.1.1.10.www: S 674719801:674719801(0) win 65535
19:38:41.803647 eth0 P 10.1.1.13.2349 > 10.1.1.10.www: S 674719801:674719801(0) win 65535
19:38:41.823650 eth0 P 10.1.1.13.1609 > 10.1.1.10.www: S 674719801:674719801(0) win 65535
19:38:41.843652 eth0 P 10.1.1.13.2384 > 10.1.1.10.www: S 674719801:674719801(0) win 65535
19:38:41.863653 eth0 P 10.1.1.13.2306 > 10.1.1.10.www: S 674719801:674719801(0) win 65535
19:38:41.883656 eth0 P 10.1.1.13.1981 > 10.1.1.10.www: S 674719801:674719801(0) win 65535
19:38:41.903658 eth0 P 10.1.1.13.2203 > 10.1.1.10.www: S 674719801:674719801(0) win 65535
19:38:41.923662 eth0 P 10.1.1.13.2004 > 10.1.1.10.www: S 674719801:674719801(0) win 65535
```

This capture contains what appears to be a TCP SYN Flood attack. Note the quick burst of TCP SYN packets to the target host on port 80 (WWW). Also, note the use of duplicate sequence numbers indicating that the packets were crafted. Since these packets were captured outside the 10.1.1.0 network the use of 10.1.1.13 for the source address indicates that the source was spoofed. By rapidly sending these packets to the target machine the attacker hopes to fill the target's incomplete connection queue. This will result in denial of service for clients attempting to access the web server on 10.1.1.10 until the incomplete connections have timed out. This attack hinges on 10.1.1.13 not responding to the SYN-ACK requests that would be generated by 10.1.1.10 and completing the three-way handshake.

## DETECT 9 – LAND ATTACK

```
19:52:59.558670 eth0 P 10.1.1.15.netbios-ssn > 10.1.1.15.netbios-ssn: S 3868:3868(0) win 2048
```

This capture contains a TCP SYN packet with the same source/destination IP address and the same source/destination port address. This packet configuration is characteristic of the Land Attack. This attack is generally used against Windows hosts and the expected outcome is that the target will lock due to its inability to handle the identical source/destination addresses/ports.

## DETECT 10 – WINNUKE

```
19:57:09.233393 eth0 P 10.2.2.200.1058 > 10.1.1.15.netbios-ssn: S 2583630555:2583630555(0) win 32120 <mss 1460,sackOK,timestamp 344133 0,nop,wscale 0> (DF)
19:57:09.233622 eth0 P 10.1.1.15.netbios-ssn > 10.2.2.200.1058: S 330418:330418(0) ack 2583630556 win 8760 <mss 1460,nop,nop,sackOK> (DF)
19:57:09.234325 eth0 P 10.2.2.200.1058 > 10.1.1.15.netbios-ssn: . 1:1(0) ack 1 win 32120 (DF)
19:57:09.239763 eth0 P 10.2.2.200.1058 > 10.1.1.15.netbios-ssn: P 1:4(3) ack 1 win 32120 urg 3>>> NBT (DF)
19:57:09.239939 eth0 P 10.1.1.15.netbios-ssn > 10.2.2.200.1058: FP 1:6(5) ack 4 win 8758>>> NBT (DF)
19:57:09.240618 eth0 P 10.2.2.200.1058 > 10.1.1.15.netbios-ssn: . 4:4(0) ack 7 win 32120 (DF)
19:57:09.352708 eth0 P 10.2.2.200.1058 > 10.1.1.15.netbios-ssn: R 4:4(0) ack 7 win 32120 (DF)
```

This capture appears to be of an attacker attempting the WinNuke attack. Note the attacker successfully completing the three-way handshake with the target on port 139 (NetBIOS). Once the connection is complete the attacker sends a packet on port 139 with the urgent flag set and the urgent value set to 3. On some Windows systems this combination will cause a "Blue Screen of Death". Note that this system was able to deny the attack.

Last Updated: July 26th, 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 Virginia Beach 2017 | Virginia Beach, VA | Aug 21, 2017 - Sep 01, 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 |
| SANS Baltimore Fall 2017 | Baltimore, MD | Sep 25, 2017 - Sep 30, 2017 | Live Event |
| 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 |
| 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 |
| SANS Seattle 2017 | Seattle, WA | Oct 30, 2017 - Nov 04, 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 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 |