Thermoptic Camouflage

TOTAL IDS EVASION

Brian Caswell

H D Moore

Who are we

- **Brian Caswell**
  - Principal Research Engineer, Sourcefire
  - Metasploit Developer
  - The Shmoo Group
- **H D Moore**
  - Director of Security Research, BreakingPoint Systems
  - Metasploit Founder

Our Plan for Today

- Evasion at every layer
- Attacking the IDS
- Demonstration

Evasion Principles

- Know your target
  - Abuse target-specific behavior
- Know your network
  - Abuse TTL and routing issues
- Know your IDS
  - Abuse signature matching engines
  - Abuse hardware limitations

Evasion Layers

- Hardware
  - Layers 1-2
- Operating System
  - Layers 3-4
- Application
  - Layers 5-7

Driver Modeling - Evasion at Layer 2

Ethernet ambiguities

- Slightly oversized frames
- Broadcast destinations
- Multiple VLAN headers

Not useful or practical

- Requires local media access
- IPS likely to drop the frame

OS Modeling - Evasion at Layer 3/4

Ptacek & Newsham

- TTL tricks
- IP fragmentation
- TCP fragmentation
- TCP sequence issues

Other tricks

- Host vs network filtering
- Fake connection tear-downs

Fragmentation - Overview

IP fragmentation for newbies

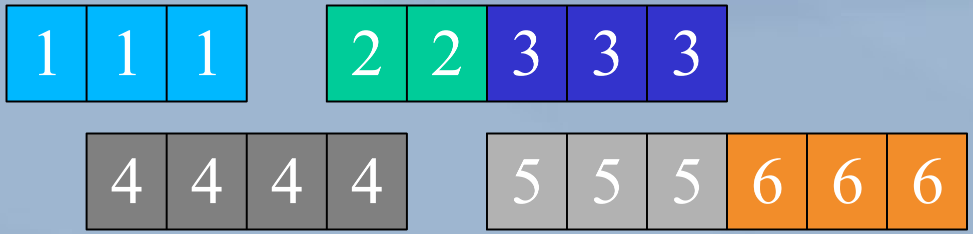
- Split an IP packet into fragments
- Minimum fragment size is 8 bytes

IP stacks handle this different ways

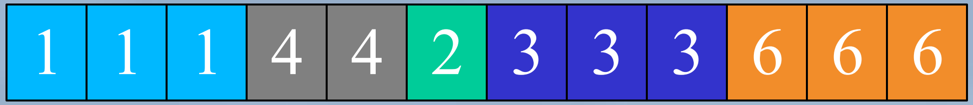
- Overlaps, duplicates, gaps, oh my!
- Abuse differences to evade IDS

Fragmentation Models

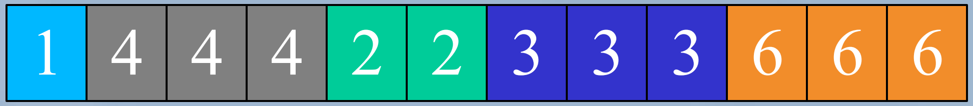
- Paxson & Handley



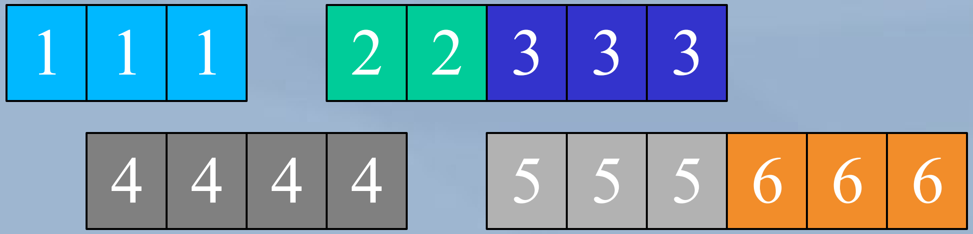
- BSD



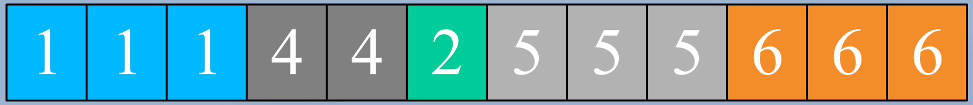
- BSD-Right



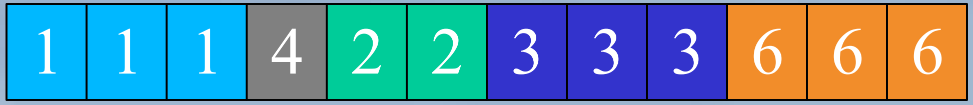
Fragmentation Models



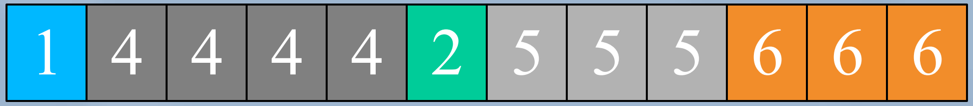
- Linux



- First (Windows)

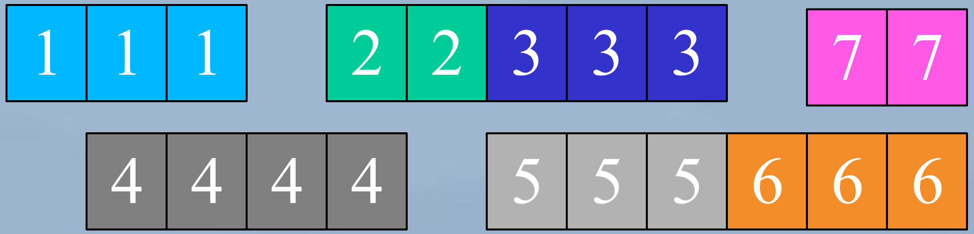


- Last (IOS)

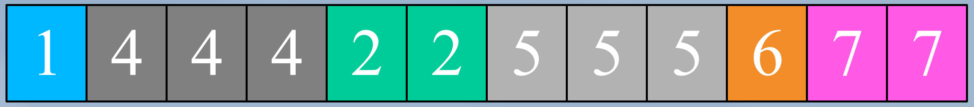


Fragmentation - Complications

- Novak/Sturges Model
- NONE - Drop frags (New IOS)

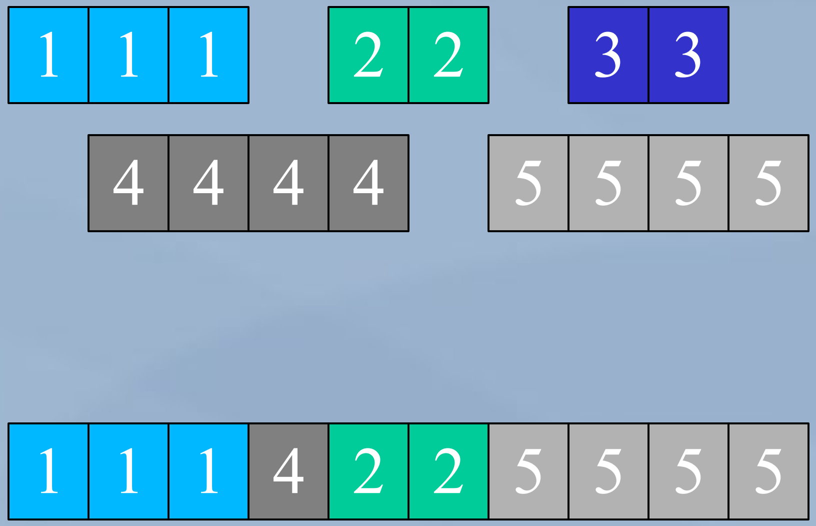


- Fragmentation - BSD-Right



Fragmentation - Windows/Solaris

- Windows / Solaris



Application Modeling - Layer 5/6

Millions of applications

- Protocol code differences
- Vendor specific extensions
- Error condition handling

Fun tricks for every protocol!

- HTTP, FTP, SMTP, DNS
- SunRPC, DCERPC, SMB

Application Modeling - Layer 5/6

Sidestep, Robert Graham of ISS

- SunRPC fragmentation
- FTP telnet option negotiation
- HTTP URI encoding

Many new techniques

- Vendor-specific URI encoding
- Evasion for 'deep' protocols
- Client-side attack evasion

SMB Evasions

SMB is a transport protocol

- Remote file access
- System administration
- Network authentication
- Remote procedure calls

SMB Evasions

SMB based vulnerabilities

- Malware propagation
- Remote registry access
- Authentication attacks
- DCERPC transport
  - MS04-011, MS04-007
  - MS05-039, MS06-025
  - Distributed COM

SMB Evasions

What is an IDS to do?

- Signature-only
- State track + signature
- State + context + signature
- Complete protocol emulation
  - What version of the protocol?
  - What version-specific options?
  - What vendor-specific options?

SMB Evasions (before & after)

Segmented read and write operations

- Independent of TCP and IP layers
- IDS must track length and offset
- Evade DCERPC signatures
- Evade malware signatures
- Offset value ignored for pipes

Demonstration

SMB Evasions (before & after)

Data and parameter padding

- Offset value determines location
- Used to align fields in memory
- Abused to fill with bogus data
- Max padding is ~4000 bytes...
  - **Insert fake SMB requests**
  - **Trigger low-risk signatures**

Demonstration

SMB Evasions (before & after)

The SMB ECHO command

- Simple command echos data
- Useful to trick SMB state engines
- Max size is greater than MTU...
  - Insert fake SMB responses
  - Trigger low-risk signatures

SMB Evasions (before & after)

SMB Transaction “PIPE” string

- Normally just \PIPE\
- Not validated by the OS
- Max length is ~4000 bytes
- Evade almost all Trans signatures
- Also useful for state engine attacks

Demonstration

SMB Evasions (before & after)

SMB CreateAndX Path Names

- Paths are normalized by target
- Trivial to obfuscate with \\\\\\\\\\\\\\
- Evade many DCERPC signatures
- Evade malware signatures

Demonstration

SMB Evasions (before & after)

Unicode & Non-Unicode Strings

- Evade signatures with Unicode off
- All Unicode-based evasions apply
- Remember the IIS Unicode bug?
- Same thing applies to SMB paths

Demonstration

SMB Evasions (before & after)

Evasion with AndX Chains

- Multiple commands per request
- Login, open file, write, close, logoff
- Also useful for state engine attacks

Evasion with Request Stacking

- Pile all SMBs into one send()
- Side effect of Nagle algorithm

DCERPC Evasion

Lets talk DCERPC

- Supports multiple transports
  - TCP, HTTP, UDP, SMB (4+ ways)
- Multiple ways to represent data
  - Big endian or little endian byte order
  - Unicode, EBCDIC, or ASCII strings
  - IEEE, VAX, Cray, or IBM floats
- Supports fragmentation
  - IP + TCP + SMB + DCERPC = ?
- Excellent source of new vulns :-)

DCERPC Evasion

DCERPC Basics

- Connect to the transport
- Bind to specific UUID and version
- Call function by number

Function parameters

- Encoding specified by client
- Uses the NDR encoding system

DCERPC Evasion

DCERPC Bind evasions

- Bind to multiple UUIDs at once
- Bind to one UUID then AlterContext
- Bind with authentication field

DCERPC Evasion

DCERPC Call evasions

- Fragment data across many requests
- Encrypt data with packet privacy
- Append random data to NDR stub
- Prepend an Object ID

DCERPC Evasion

DCERPC Transport evasion

- RPC over HTTP via RpcProxy
- One-packet UDP function calls
  - Uses the idempotent flag

Ports and processes

- Shared processes share pipes
- Choose which named pipe to use
- Everyone loves \BROWSER

DCERPC - NDR Strings

“ABCDE” in Little Endian ASCII

- Len + Offset + TotalLen + string + null pad to 32bit boundary

```
"\x05\x00\x00\x00”
"\x00\x00\x00\x00”
"\x05\x00\x00\x00”
"ABCD”
"E\x00\x00\x00"
```

- Use non-NULLs for padding!

DCERPC - NDR Strings

Empty string "" in Little Endian ASCII

- Len + Offset + TotalLen + string + pad to 32bit boundary

```
"\x00\x00\x00\x00"
"\x00\x00\x00\x00"
"\x00\x00\x00\x00"
"\x00\x00\x00\x00"
```

- Or on some services

```
"\x00\x00\x00\x00"
```

DCERPC - ISystemActivator

Blaster NDR stub

```
*(DWORD *)(buf2+0x10)=*(DWORD *)(buf2+0x10)+len-0xc;
*(DWORD *)(buf2+0x80)=*(DWORD *)(buf2+0x80)+len-0xc;
*(DWORD *)(buf2+0x84)=*(DWORD *)(buf2+0x84)+len-0xc;
*(DWORD *)(buf2+0xb4)=*(DWORD *)(buf2+0xb4)+len-0xc;
*(DWORD *)(buf2+0xb8)=*(DWORD *)(buf2+0xb8)+len-0xc;
*(DWORD *)(buf2+0xd0)=*(DWORD *)(buf2+0xd0)+len-0xc;
*(DWORD *)(buf2+0x18c)=*(DWORD *)(buf2+0x18c)+len-0xc;
```

DCERPC - ISystemActivator

How did vendors look for this attack?

- MEOW prefixes for objects

```
4d45 4f57 0400 0000 a201 0000 0000 0000    MEOW............
c000 0000 0000 0046 3803 0000 0000 0000    .......F8.......
c000 0000 0000 0046 0000 0000 f005 0000    .......F........
e805 0000 0000 0000 0110 0800 cccc cccc    ................
c800 0000 4d45 4f57 e805 0000 d800 0000    ....MEOW........
```

- Long Paths
  - \\TOO_LONG_PATH_HERE\

DCERPC - ISystemActivator Path

Contains 8 objects, bad one is #7

- Paths everywhere!
- One object allows ~1Mb of padding!
- All Windows path rules also apply

Text Protocols: Header Folding

Header parsing is ambiguous

- HTTP, SMTP, iCal, Email
- EvilHeader: Bar Biz; boo
- What does your application do?
  - "EvilHeader: Bar Bi\n ;boo"
  - "EvilHeader: Bar Biz\n boo"

Client-Side Attack Evasion

Juicy targets for many reasons

– No firewall, rich text, scripting, many bugs

So many evasion options

- Unicode
- Javascript
- Objects
- Compression
- Encryption

Unicode

Which Unicode?

- utf-16le, utf-16be, utf32-le, utf32-be,
- utf-7, utf-8
- HTTP
  - Content-Type: text/html; charset: utf-16be
- Oops
  - Start with "\xFE\xFF", forces utf-16be

UTF-8 Overlong Strings

- Encode the letter "A"
  - `41, c1a1, e081a1, f08081a1, f8808081a1, fc808080 81a1`
- "Invalid" overlong strings
  - 2 bytes
    - `c121, c161, c1a1, c1e1`
  - 3 bytes
    - `e00101, e00141, e00181, e001c1, e04101, e04141, e04181, e041c1, e08101, e08141, e08181, e081c1, e0c101, e0c141, e0c181, e0c1c1`
- 125 ways in ONE character set!

Common Javascript Evasions

<script>document.write(”EVIL") </script>

- <body onLoad= ”document.write('EVIL')”;>
- document.write(unescape('%45%56%49%4C'));
- <font style=“background: url(javascript:document.write(‘EVIL));”>

Uncommon Javascript Evasions

- <font style="background: url(jav as c ript:alert('evil'));">
- <scr\xFE\xFFipt> alert('CVE-2006-2783'); </s\xFE\xFFript>
- **Using PCRE to strip javascript?**
  - Unicode (default doesn't support it…)
  - Rejects overlong strings, 0xFF, or 0xFE

Base64 your HTML

```
<OBJECT ID="w00t" TYPE="text/html"
  DATA="data:text/html;base64,ZXZpb
  CB0ZXh0">aww, too bad!</OBJECT>
```

- Equivalent to "evil text”
- Don't write signatures for this!
- Spaces matter
  - `"foo”," foo"," foo”`
  - `IGZvbw==, ICBmb28=, ICAgZm9v`

Compression Issues

- Zip Bombs
  - 100Mb => 100k - GZIP
- Who writes rules for GZIP output?
  - WMF header
- Arbitrary sized headers in GZIP
  - name, comment
- Three compression algorithms
  - gzip, deflate, compress

SSL your attacks

Encryption is fun

- Purchase a certificate ($$$)
- Compromise and hijack existing cert
- Convince the user to ignore warnings
- Use SSL wrapped CGI proxy server!

  https://www.fsurf.com/index.php?q=http://IP:8080/foo.pls

  https://proxify.com/u?http://IP:8080/foo.wmf

Attacking the IDS

Find the failure points

- Alert management
- Hardware limitations
- Session tracking
- Pattern matching
- Signature strength

IDS Alert Management

Attack the software

- Flood the alert system
- Nikto is great for this!
- Multiple alerts per packet?
  - One IDS triggers ~1050 per packet!

Attack the user

- Hide the real attack in the flood
- Abuse UI limitations to hide events

IDS Hardware Limitations

Gigabit Ethernet limits

- 1,000,000,000 bits
- 125000000 bytes
- 1602564 packets
- 1.602 packets per microsecond
- Oh, full duplex...
- 3.205 packets per microsecond

IDS Hardware Limitations

PC hardware limitations

- PCI/PCI-X limits
  - 33Mhz: 32/64 = 133/266 MB/s
  - 66Mhz: 32/64 = 266/532 MB/s
  - 100Mhz: 64 = 800MB/s

- Software interrupt limits
  - Intel Pro/1000 Server / 3Ghz P4/Xeon
  - 680,000pps RX | 840,000pps TX
  - **348Mbps** capture w/64b packets

  * Poll mode bypasses interrupt limits

IDS Hardware Limitations

PC hardware realities

- Typical Dell 1U appliance
  - Dual Intel Pro/1000 cards
  - 3.0Ghz Xeon
- 760Mbps max capture mode
- 380Mbps max inline mode
- The ICSA report agrees!
  - ISS G400 Proventia rated at 350Mbps

IDS Hardware Limitations

Network hardware realities

- FastPath vs SlowPath
  - Minimum processing on FastPath
  - SlowPath used for exceptions
- Find the SlowPath
  - Management services
  - Encryption and authentication
  - IP fragment processing

IDS Hardware Limitations

Shared cores for hardware

- A “core” is licensed for a chip
  - Provides common networking features
  - Routing, reassembly, switching, etc
- Quickest way to add a feature
  - Common choice for quick development
  - Just as buggy as any other software
  - Any flaw applies to multiple vendors :-)

IDS Hardware Limitations

Memory allocation

- Static blocks preferred over allocator
- Block must hold entire packet
- Split into “buckets” based on size
- Stream a specific packet pattern
  - Try 63, 65, 129, 257, 1025, 2049
  - Allocate all blocks in a given bucket
  - Force exceptions and pass-through

Session Tracking Limitations

Hash Collisions

- Crosby & Wallach
- key = srcip ^ dstport
- 2^16 srcip/dstports hash equally
- data[key] -> Linked List ip/port
- Force walking the linked list
- 43.78 minutes for 65k PACKETS

Session Tracking Limitations

Splay Trees

- Self-balancing binary tree
- O(log(n)) amortized over time
- Worst Case = Sorted List
- O(n) to rebalance from worst case

Demonstration

Attacking Pattern Matchers

- Find the most expensive operation
  - Force it to repeat over and over
- Trigger exception processing
  - Use invalid characters, recursion, etc
- Inject termination characters
  - Use terminator strings to fail a match
  - Depends on the signature and protocol

Attacking Pattern Matchers

```
char * search(char *buf, int buflen, char *string,
 int stringlen) {
        char *ptr = buf;
        int i = 0;
        while ( (i + stringlen) < buflen ) {
            if ( memcmp(ptr, string, stringlen) == 0 ) {
                return ptr;
            }
            i++; ptr++;
        }
        return NULL;
    }
```

Attacking Pattern Matchers

search(data, datalen, "evilfoo!", 8);

- Maximize work done by memcpy
- Send “evilfoo” * 8
- 48 calls to memcpy
- 96 to 384 memory operations[0]
- 2000 ms on a 65k packet of evilfoo

[0] Depending on platform, alignment, and libc implementation

Attacking Pattern Matchers

/.*From=[^&]{165,}.*/

- .*
  - Match any amount of any character
- From=
- [^&]{165,}
  - 165 or more bytes of anything but &
- Force repeated backtrack
  - “From=“ repeating, “&” at byte 165

Demonstration

Attacking the Signatures

- Difference between IDS and application
  - isspace
    - \t, \n, \v, \f, \r or “ “
  - Newlines
    - \r, \n, \r\n
- Force signature engine to stop early
  - Hit memory limits
    - PCRE_CONFIG_POSIX_MALLOC_THRESHOLD
  - Hit recursion limits
    - PCRE_CONFIG_STACKRECURSE
  - Hit maxiumum failure limits
    - PCRE_CONFIG_MATCH_LIMIT

Extracting signatures

- **Blackbox signature discovery**
  - Create protocol template, set boundaries
  - Enable block mode in IPS product
  - Flood request permutations and create sig :-)
- **Direct memory access**
  - Hardware bus monitoring
  - Root the box and dump the process
- **Poor cryptography**
  - Key has to accessible somewhere

Conclusion

Everything can be evaded

- At what layer?
- At what cost?
- At what speed?

Contact

Brian Caswell

- bmc[at]shmoo.com
- http://www.shmoo.com/~bmc/

H D Moore

- hdm[at]metasploit.com
- http://metasploit.com/