Using Guided Missiles in Drive-bys

Automatic browser fingerprinting and exploitation with the Metasploit Framework:

Browser Autopwn

James Lee

Browser Autopwn

- Auxiliary module for the Metasploit Framework
- Fingerprints a client
- Determines what exploits might work
- Used to suck
- Now it doesn't

Outline

- Intro
- Cluster bombs
- Guided missiles
  - Fingerprinting and targeting
- Stealth
- Demos
- Commercial comparison

whoami

- James Lee
- egypt
- Co-Founder, Teardrop Security
- Developer, Metasploit Project

My Involvement in MSF

- Started submitting patches and bug reports in 2007
- HD gave me commit access in April 2008
  - Broke the repo April 2008

The Metasploit Framework

- Created by HD Moore in 2003
  - ncurses based game
  - Later became a real exploit framework in perl
- Rewritten in ruby in 2005
  - Which is way better than python
- Extensible framework for writing exploits

I <3 MSF

- Modular payloads and encoders
- Many protocols already implemented
- Many non-exploit tools
- All kinds of exploits
  - Traditional server-side
  - Client-sides

Why Clientsides

- Karmetasploit
- Any other tool that gets you in the middle
- Users are weakest link, blah, blah, blah
- See Chris Gates

Client Exploits in MSF

- Extensive HTTP support
  - Heapspray in two lines of code
  - Sotirov's .NET DLL, heap feng shui
- Wide range of protocol-level IDS evasion
- Simple exploit in ~10 lines of code

Simple Exploit

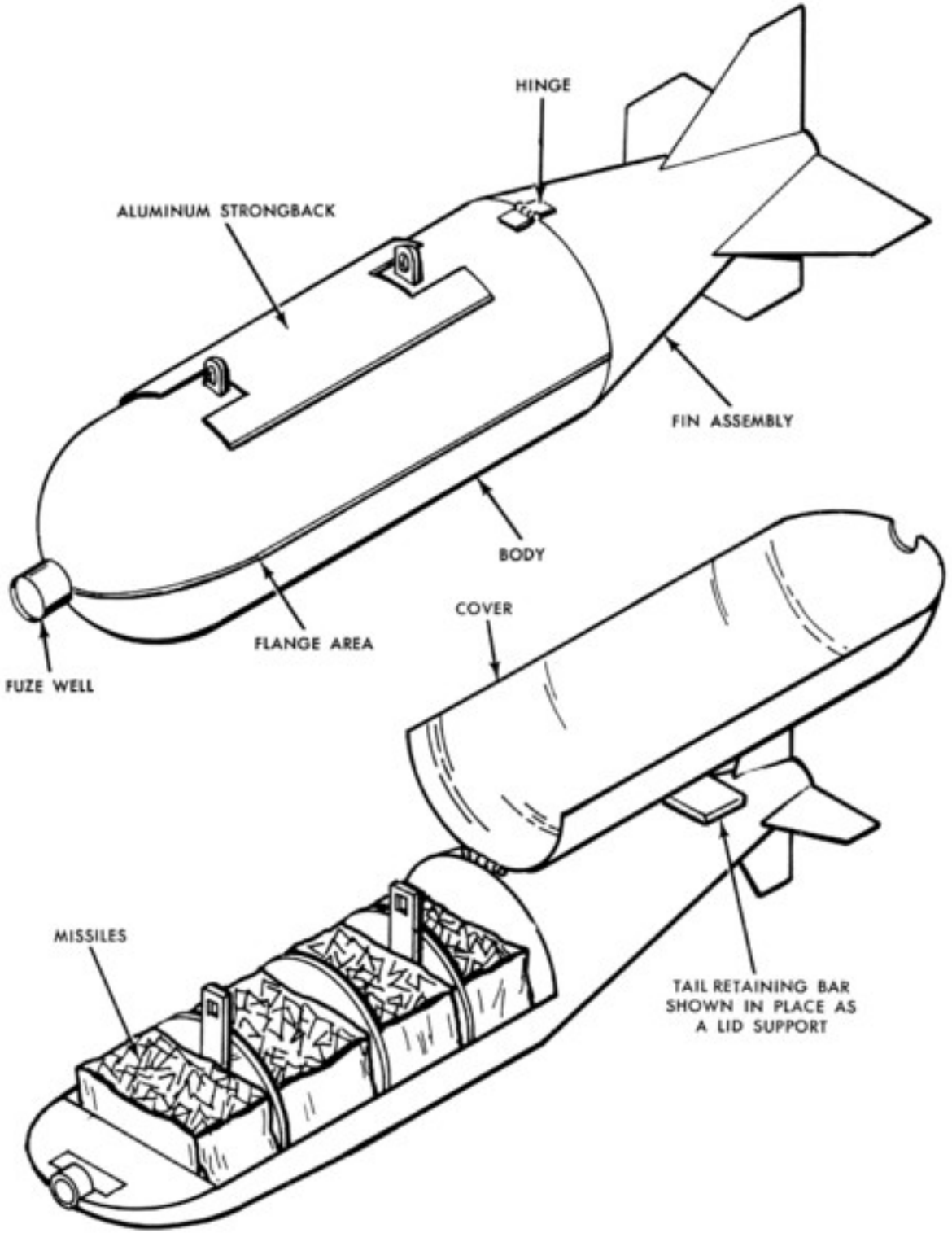
```
content = “<html><body>
<object id='obj' classid='...'></object><script>
#{js_heap_spray}
sprayHeap(#{payload.encoded}, #{target.ret}, 0x4000);
obj.VulnMethod(#{[target.ret].pack(“V”)*1000});
</script></body></html>“

send_response(client, content)
```

Or Arbitrarily Complex

- ani_loadimage_chunksize is 581 lines of code
- As of June 28, MSF has 85 browser exploit modules

Problem

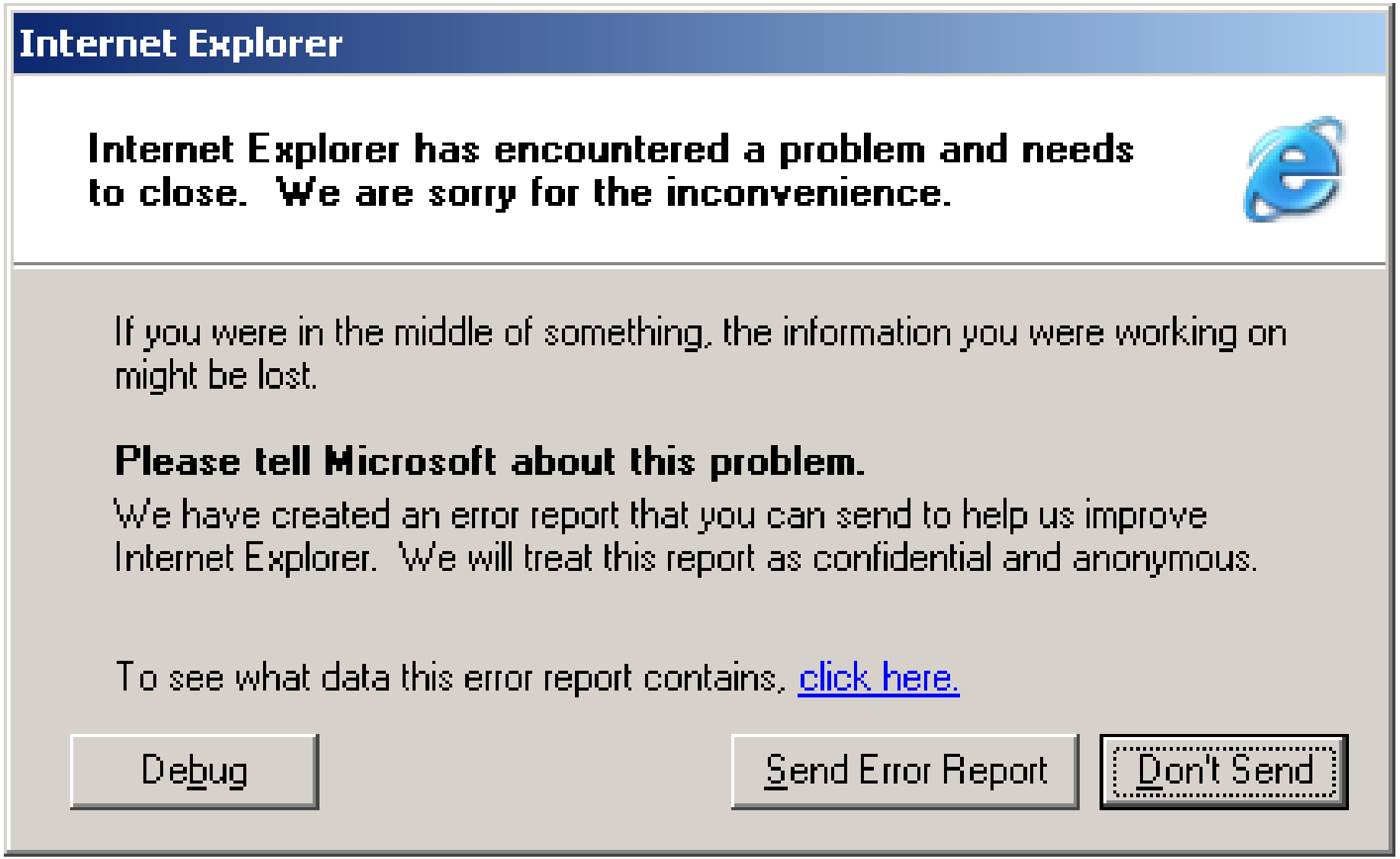

HINGE
ALUMINUM STRONGBACK
FIN ASSEMBLY
BODY
COVER
FLANGE AREA
FUZE WELL
MISSILES
TAIL RETAINING BAR
SHOWN IN PLACE AS
A LID SUPPORT

Solution

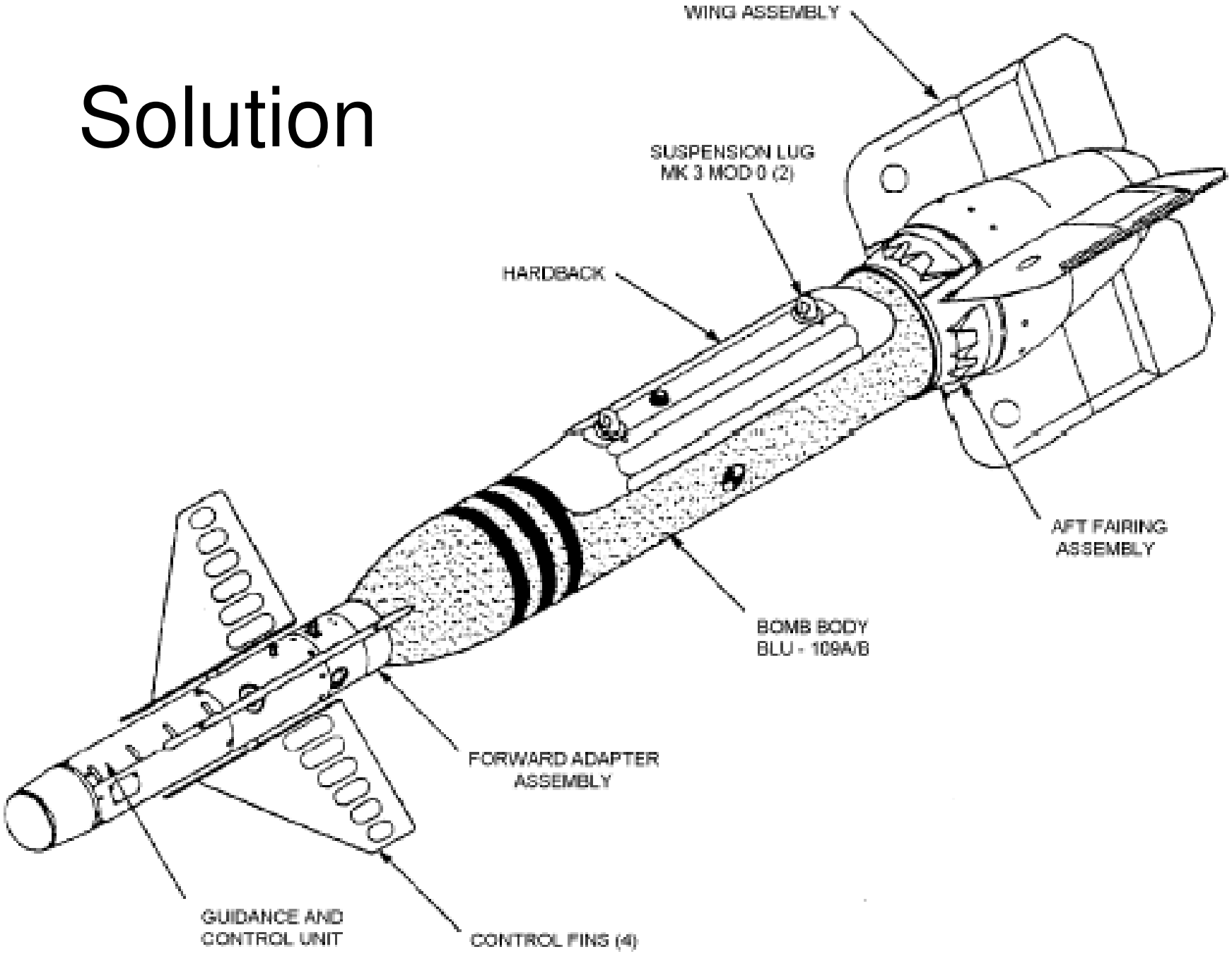
Cluster Bomb Approach

- Is it IE? Send all the IE sploits
- Is it FF? Send all the FF sploits
- Originally exploits were ad-hoc
  - Pain in the ass when new sploits come out

Problem


Internet Explorer
Internet Explorer has encountered a problem and needs to close. We are sorry for the inconvenience.
If you were in the middle of something, the information you were working on might be lost.
Please tell Microsoft about this problem.
We have created an error report that you can send to help us improve Internet Explorer. We will treat this report as confidential and anonymous.
To see what data this error report contains, click here.
Debug
Send Error Report
Don't Send

Solution


WING ASSEMBLY
SUSPENSION LUG
MK 3 MOD 0 (2)
HARDBACK
AFT FAIRING
ASSEMBLY
BOMB BODY
BLU - 109A/B
FORWARD ADAPTER
ASSEMBLY
GUIDANCE AND
CONTROL UNIT
CONTROL FINS (4)

Guided Missile Approach

- Better client and OS fingerprinting
  - less likely to crash or hang the browser
- Only send exploits likely to succeed
  - Browser is IE7? Don't send IE6 sploits, etc.

Fingerprinting the Client

- User Agent
  - Easy to spoof
  - Easy to change in a proxy
  - A tiny bit harder to change in JS

Fingerprinting the Client

- Various JS objects only exist in one browser
  - window.opera, Array.every
- Some only exist in certain versions
  - window.createPopup, Array.every, window.Iterator
- Rendering differences and parser bugs
  - IE's conditional comments

Internet Explorer

- Parser bugs, conditional comments
  - Reliable, but not precise
- ScriptEngine*Version()
  - Almost unique across all combinations of client and OS
  - Brought to my attention by Jerome Athias

Opera

- window.opera.version()
  - Includes minor version, e.g. “9.61”

Hybrid Approach for FF

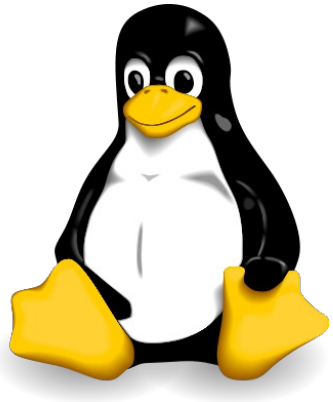
- Existence of document.getElementsByClassName means Firefox 3.0
- If User Agent says IE6, go with FF 3.0
- If UA says FF 3.0.8, it's probably not lying, so use the more specific value

Safari

- Still in progress
- Existence of window.console
  - If Firebug is installed on FF, shows up there, too
- Availability of window.onmousewheel
  - Defaults to null, so have to check typeof

Fingerprinting the OS

- User Agent
- Could use something like p0f
- From the server side, that's about it



Internet Explorer

- Again, ScriptEngine*Version()
- Almost unique across all combinations of client and OS, including service pack

Opera

- Each build has a unique opera.buildNumber()
- Gives platform, but nothing else

Firefox

- navigator.platform and friends are affected by the User Agent string
- navigator.oscpu isn't
  - “Linux i686”
  - “Windows NT 6.0”

Others

- Really all we're left with is the User Agent
- That's okay, most don't lie
  - And those that do are likely to be patched anyway
- Generic, works everywhere when UA is not spoofed

Future Fingerprinting

- QuickTime
- Adobe
- Less well-known third party stuff

ActiveX

- `“new ActiveXObject()”` works if you have the class name
- Otherwise, IE doesn't seem to have a generic way to tell if an ActiveX object got created
  - document.write(“<object ...>”)
  - document.createElement(“object”)

Solution

- typeof(obj.method)
  - 'undefined' if the object failed to initialize
  - 'unknown' or possibly a real type if it worked

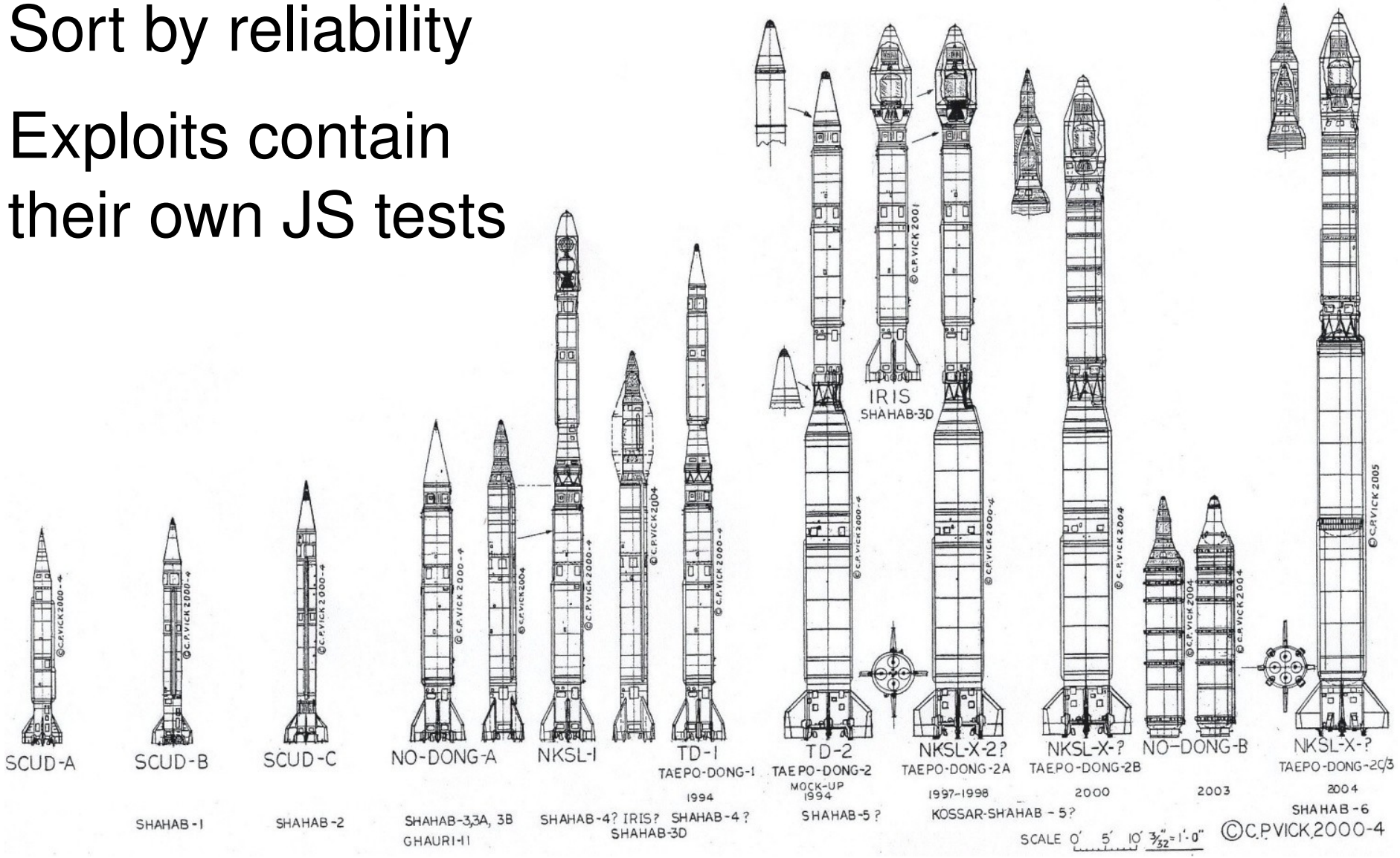
Target Acquired

What is it Vulnerable to?

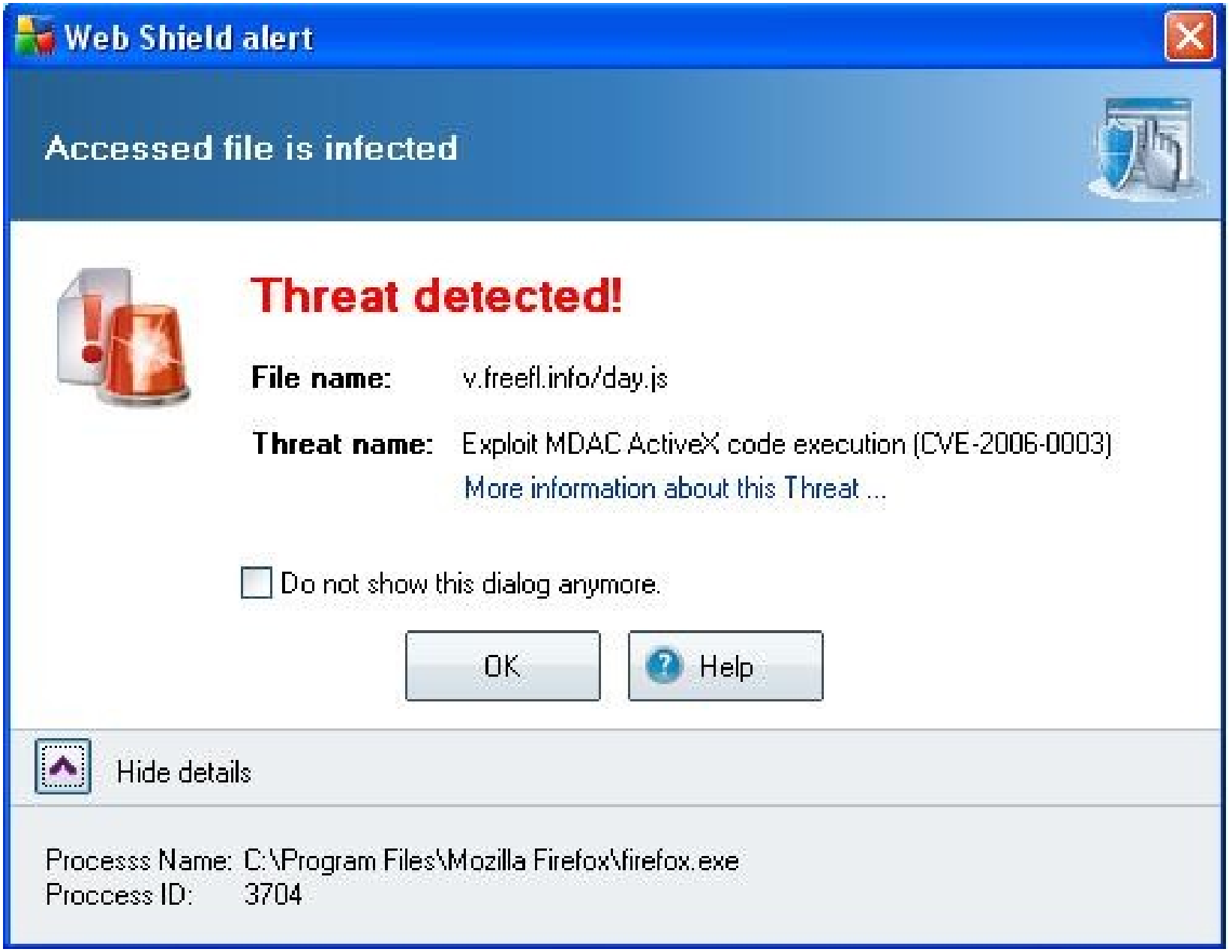
- Coarse determination server-side
  - JavaScript builds fingerprint, sends it back to the server
  - Server sends sploits that match the browser and OS, possibly version
- Fine determination client-side
  - `navigator.javaEnabled` exists, try `mozilla_navigatorjava`

Select a Missile

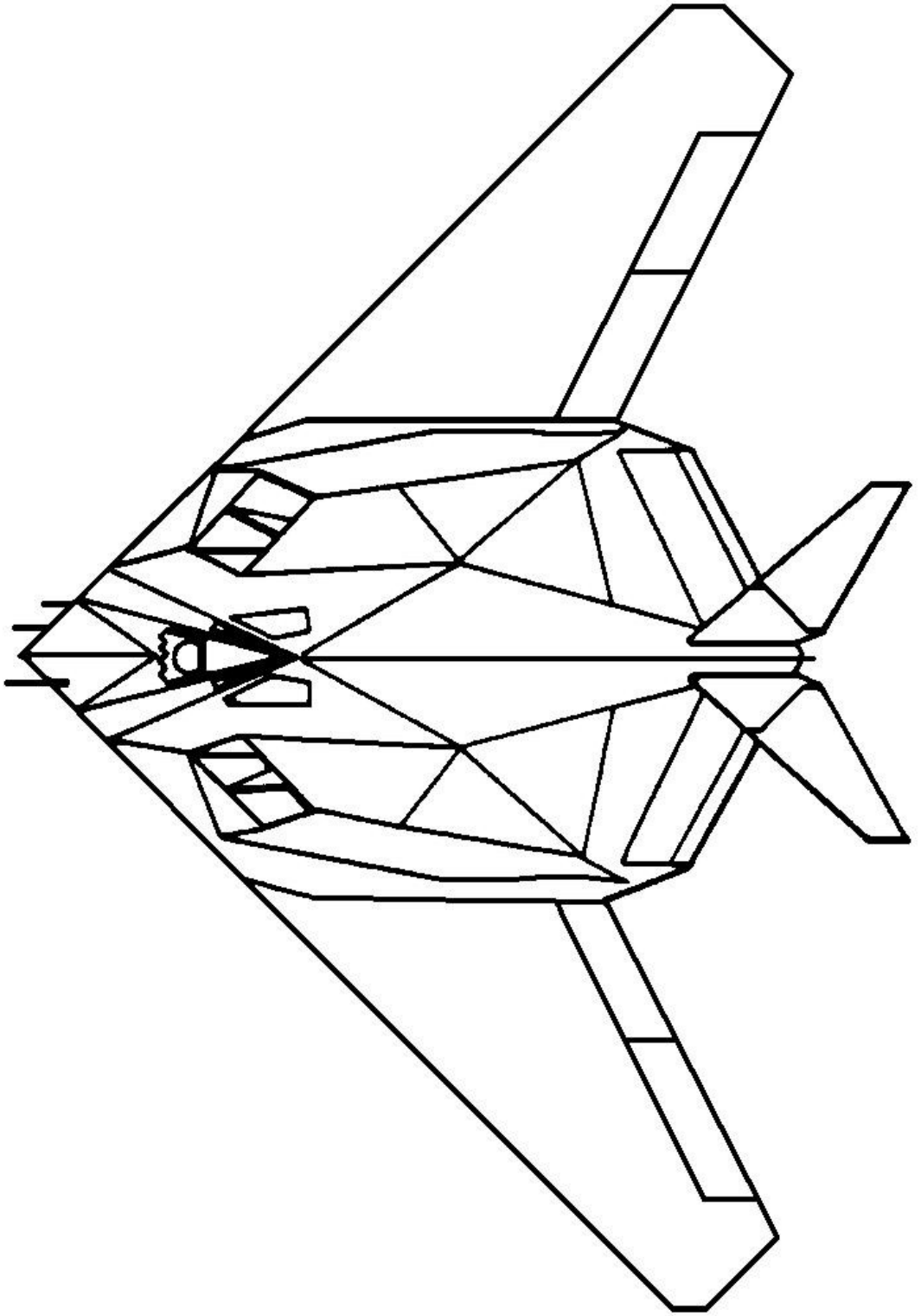
- Sort by reliability
- Exploits contain their own JS tests



Problem


Web Shield alert
Accessed file is infected
Threat detected!
File name: v.freefl.info/day.js
Threat name: Exploit MDAC ActiveX code execution (CVE-2006-0003)
More information about this Threat ...
Do not show this dialog anymore.
OK
Help
Hide details
Processs Name: C:\Program Files\Mozilla Firefox\firefox.exe
Proccess ID: 3704

Solution



Obfuscation

- Randomize identifiers
- Build strings from other things
- JSON / AJAX
- Obfuscation is not crypto

Encryption

- Put a key in the URL
  - Not available in the stand-alone script
- Simple XOR is enough to beat AV and NIDS
- If they figure it out, it's easy to make the crypto stronger

Demonstrations

And we're back...

- I hope that worked
- Now how do YOU make exploits work within this framework?

Writing Exploits

- Add autopwn_info() to top of exploit class
- :ua_name is an array of browsers this exploit will work against
- :vuln_test is some javascript to test for the vulnerability (unless it's ActiveX)
  - Usually comes directly from the exploit anyway

Example: mozilla_navigatorjava

```
include Msf::Exploit::Remote::BrowserAutopwn
autopwn_info({
    :ua_name    => HttpClients::FF,
    :javascript => true,
    :rank       => NormalRanking,#reliable memory corruption
    :vuln_test  => %Q|
        if (
           window.navigator.javaEnabled &&
           window.navigator.javaEnabled()
        ){
            is_vuln = true;
        }
        |,
})
```

Example: ms06_067_keyframe

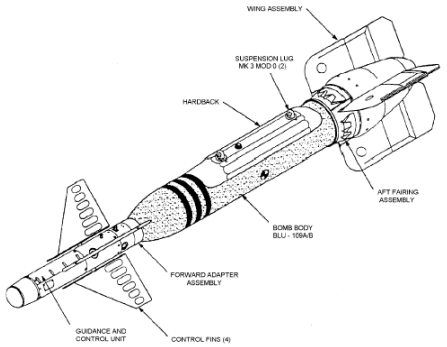
```
include Msf::Exploit::Remote::BrowserAutopwn
autopwn_info({
    :ua_name     => HttpClients::IE,
    :javascript => true,
    :os_name     => OperatingSystems::WINDOWS,
    :vuln_test   => 'KeyFrame',
    :classid     => 'DirectAnimation.PathControl',
    :rank        => NormalRanking #reliable memory corruption
})
```

Example: winzip_fileview

```
include Msf::Exploit::Remote::BrowserAutopwn
autopwn_info({
    :ua_name     => HttpClients::IE,
    :javascript => true,
    :os_name     => OperatingSystems::WINDOWS,
    :vuln_test   => 'CreateFolderFromName',
    :classid     => '{A09AE68F-B14D-43ED-B713-BA413F034904}',
    :rank        => NormalRanking #reliable memory corruption
})
```

Browser Autopwn Summary

- Reliable Target Acquisition
- Smart Missile Selection
- Stealthy from an AV perspective
- Easy to extend
- Detection results stored in a database



Commercial Comparison

- Mpack
- Firepack
- Neosploit
- Luckysploit

Mpack, Firepack

- Hard to acquire
- Old exploits
- Detection is only server-side
- Hard to change or update exploits
- Obfuscation + XOR

Neosploit

- Compiled ELFs run as CGI
- Unless you get the source or do some RE, you won't really know what it does

Luckysploit

- Real crypto (RSA, RC4)
- Even harder to acquire

Browser Autopwn

- Easy to write new exploits or take out old ones
- Free (three-clause BSD license)
- Easy to get (http://metasploit.com)
- Not written in PHP
- OS and client detection is client-side, more reliable in presence of spoofed or borked UA

Future

- More flexible payload selection
- Stop when you get a shell
  - Maybe impossible in presence of NAT/proxies
- Easier-to-use JS obfuscation
- UAProf for mobile devices
- Integration with MetaPhish

Download it

- svn co http://metasploit.com/svn/framework3/trunk
- Submit patches to msfdev@metasploit.com

Thanks

- hdm, valsmith, tebo, mc, cg, Dean de Beer, pragmatk
- Everybody who helped with testing
- Whoever created ActiveX