Defense by Numb3r5

Making problems for script k1dd13s and scanner monkeys

@ChrisJohnRiley

podcaster
blogger
dirtysec
bad-programmer
researcher
twitter
Metasploit
Not-an-Expert
c22
eurotrash
con-junkie
PTES
scared
pentester

“The wisest man, is he who knows, that he knows nothing”

Socrates: Apology, 21d

edge case

WARNING

This talk contains:

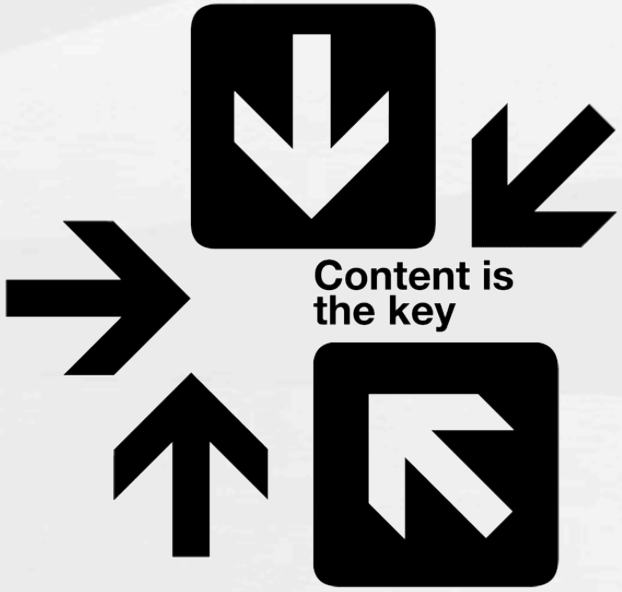
- Numbers
- Bad Jokes
- Traces of peanuts
- Did I mention numbers?

TL;DR

Goals for this talk

Describe the defensive uses of HTTP status codes

1) What
2) Why
3) How
4) Goals
5) Bringing it together
6) Review



#1

[WHAT?]

HTTP Status Codes

```
HTTP/1.1 206 Partial content
Date: Wed, 15 Nov 1995 06:25:24 GMT
Last-Modified: Wed, 15 Nov 1995 04:58:
Content-Range: bytes 21010-47021/47022
Content-Length: 26012
Content-Type: image/gif
```

Seems like such a *Small* detail

… *small* detail,

big impact

HTTP Status Codes

- Majority part of RFC 2616 (HTTP/1.1)
- 5 *main* classes of response
  - 1XX *informational*
  - 2XX *success*
  - 3XX *redirection*
  - 4XX *client error*
  - 5XX *server error*

HTTP Status Codes

- Proposed RFC* for 7XX codes
- Examples:
  - 701 *Meh*
  - 719 *I am not a teapot*
  - 721 *Known unknowns*
  - 722 *Unknown unknowns*
  - 732 *Fucking Unic□de*

* https://github.com/joho/7XX-rfc

#1.1

BASICS

AKA: THE BORING THEORY BIT

1XX Informational

- Indicates response received
- Processing is not yet completed
    - 100 Continue
    - 101 Switching Protocols
    - 102 Processing (WebDAV RFC 2518)

2XX Success

- Indicates response received
- Processed and understood
  - 200 OK
  - 201 Created
  - 202 Accepted
  - 203 Non-Authoritative Information
  - 204 No Content

2XX Success (cont.)

- 205 Reset Content
- 206 Partial Content
- 207 Multi-Status (WebDAV RFC 4918)

Codes not supported by Apache

- 208 Already Reported
- 226 IM Used
- 250 Low on Storage Space

3XX Redirection

- Action required to complete request
  - 300 Multiple Choices
  - 301 Moved Permanently
  - 302 Found (Moved Temporarily)
  - 303 See Other
  - 304 Not Modified

3XX Redirection (cont.)

- 305 Use Proxy
- 306 Switch Proxy (unused)
- 307 Temporary Redirect

Codes not supported by Apache

- 308 Permanent Redirect

4XX Client Error

- Client caused an error
  - 400 Bad Request
  - 401 Unauthorized
  - 402 Payment Required
  - 403 Forbidden
  - 404 Not Found
  - 405 Method Not Allowed

4XX Client Error (cont.)

- 406 Not Accessible
- 407 Proxy Authentication Required
- 408 Request Timeout
- 409 Conflict
- 410 Gone
- 411 Length Required

4XX Client Error (cont.)

- 412 Precondition Failed
- 413 Request Entity Too Large
- 414 Request-URI Too Long
- 415 Unsupported Media Type
- 416 Request Range Not Satisfiable
- 417 Expectation Failed
- 418 I'm a Teapot (IETF April Fools RFC 2324)

4XX Client Error (cont.)

- 419 / 420 / 421 Unused
- 422 Unprocessable Entity (RFC 4918)
- 423 Locked (RFC 4918)
- 424 Failed Dependency (RFC 4918)
- 425 No Code / Unordered Collection
- 426 Upgrade Required (RFC 2817)

4XX Client Error (cont.)

Codes not supported by Apache

- 428 Precondition Required
- 429 Too Many Requests
- 431 Request Header Fields Too Large
- 444 No Response (NGINX)
- 449 Retry With (Microsoft)
- 450 Blocked by Win. Parental Controls
- 451 Unavailable For Legal Reasons
- 494 Request Header Too Large (NGINX)
- 495 Cert Error (NGINX)
- 496 No Cert (NGINX)
- 497 HTTP to HTTPS (NGINX)
- 499 Client Closed Request (NGINX)

5XX Server Error

- Server error occurred
  - 500 Internal Server Error
  - 501 Not Implemented
  - 502 Bad Gateway
  - 503 Service Unavailable
  - 504 Gateway Timeout
  - 505 HTTP Version Not supported

5XX Server Error (cont.)

- 506 Variant Also Negotiates (RFC 2295)
- 507 Insufficient Storage (WebDAV RFC 4918)
- 508 Loop Detected (WebDAV RFC 5842)
- 509 Bandwidth Limit Exceeded (apache ext.)
- 510 Not Extended (RFC 2274)

Codes not supported by Apache

- 511 Network Authentication Required (RFC 6585)
- 550 Permission Denied
- 598 Network Read Timeout Error (Microsoft Proxy)
- 599 Network Connection Timeout Error (Microsoft Proxy)

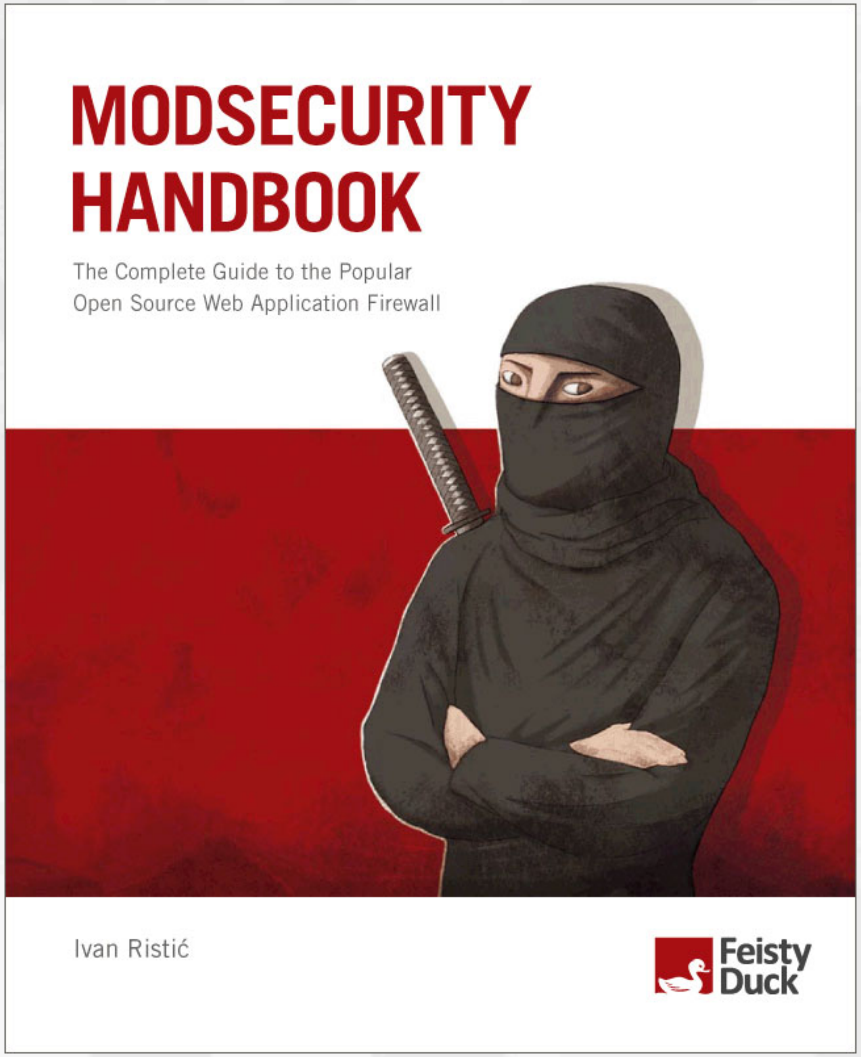
OMG Enough with the numb3rs already!!!!

BORING

#2

[WHY?]

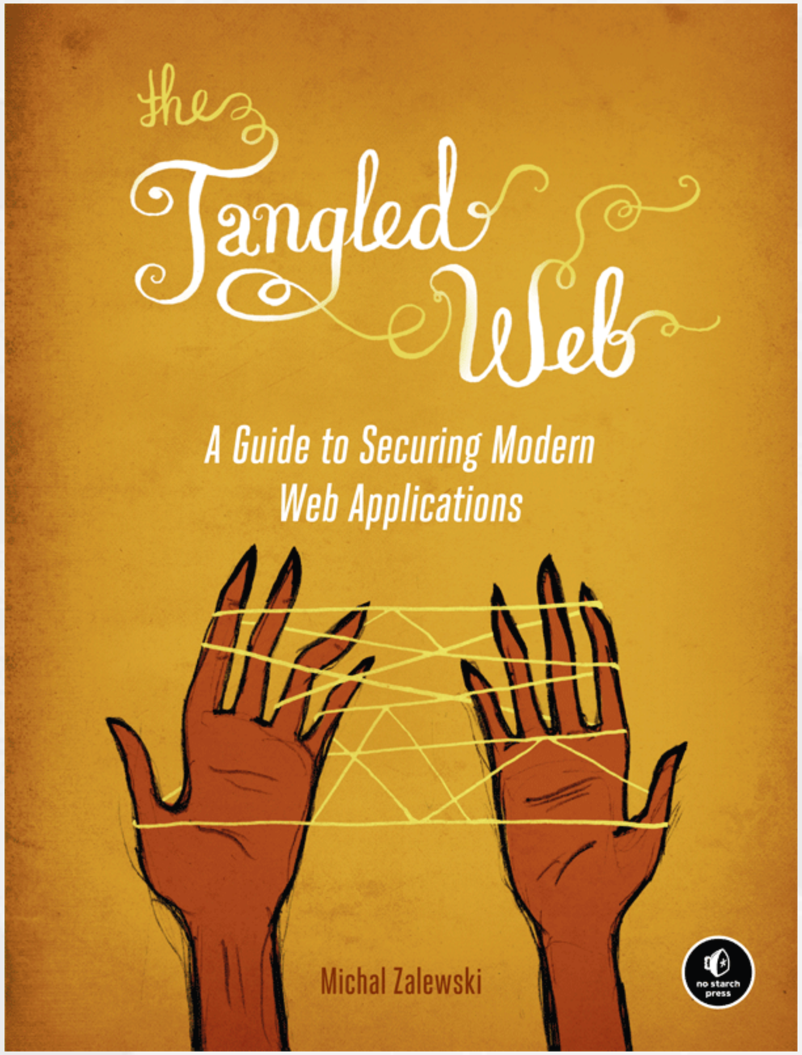
It started as a simple idea…


MODSECURITY HANDBOOK

The Complete Guide to the Popular Open Source Web Application Firewall

Ivan Ristić

Feisty Duck


the Tangled Web

A Guide to Securing Modern Web Applications

Michal Zalewski

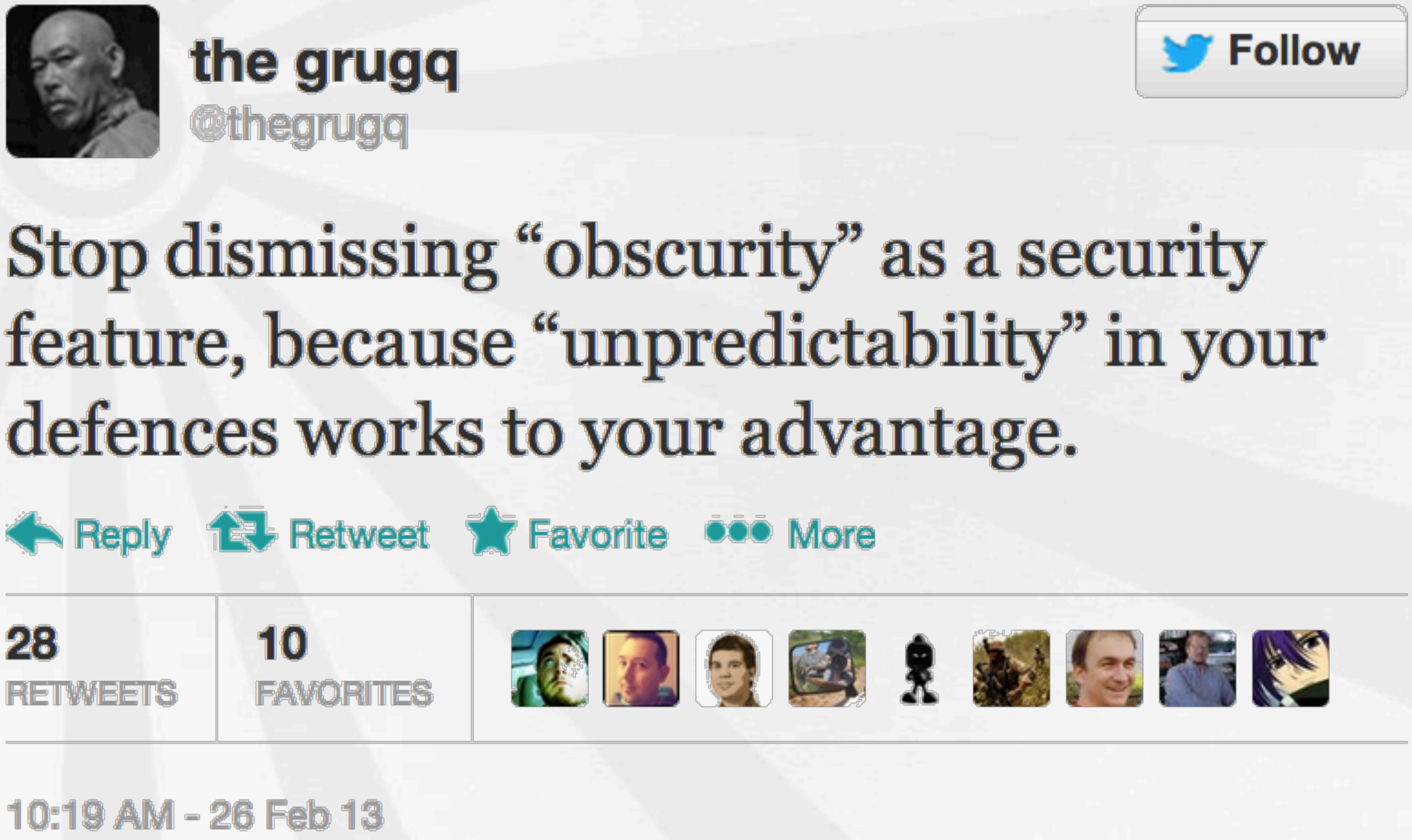
no starch press

… and started to think

SCREW WITH SCANNERS

… AND SCRIPT K1DD13S

THAT SOUNDS LIKE FUN!


the grugq
@thegrugq

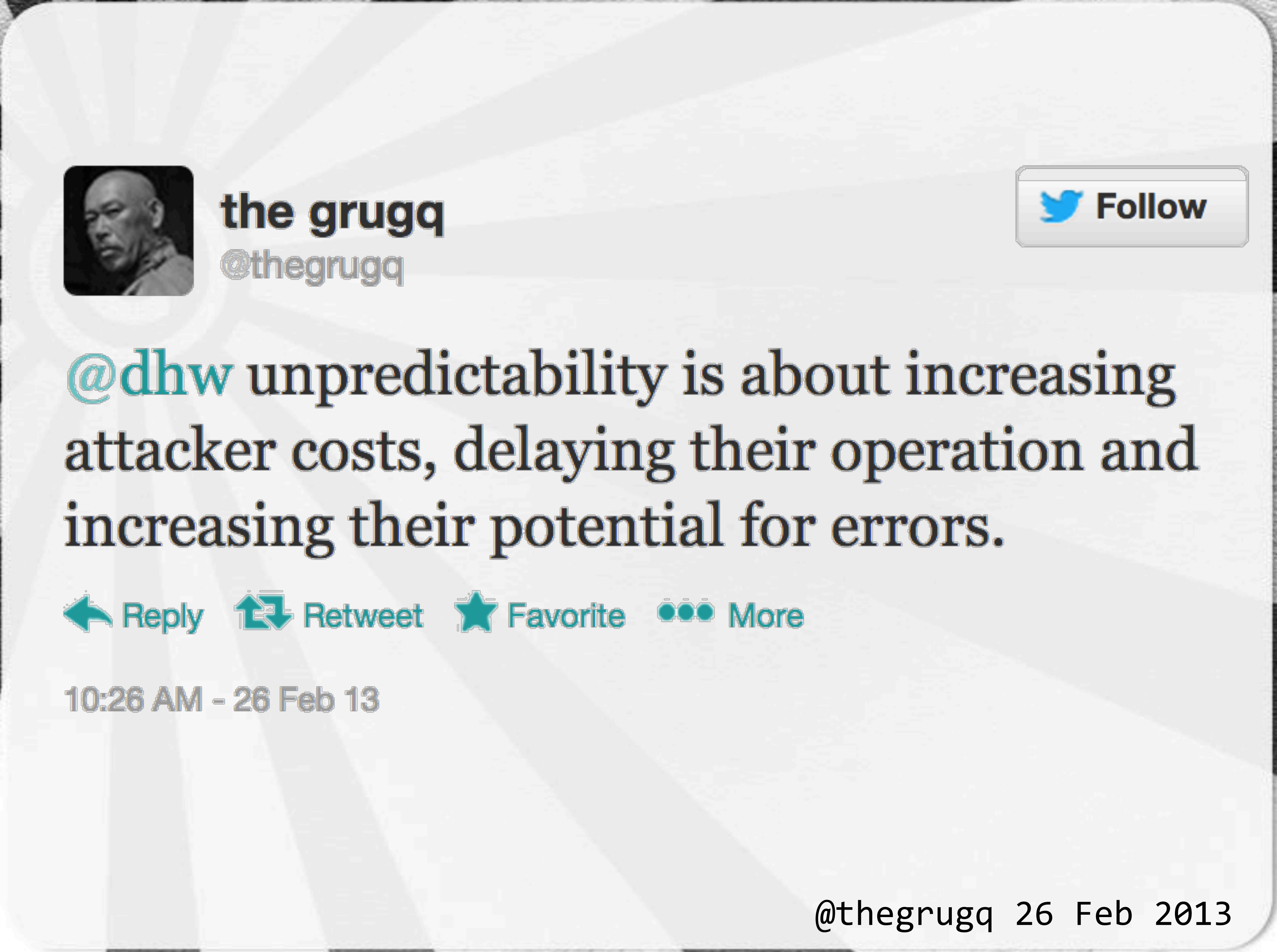
Stop dismissing "obscurity" as a security feature, because "unpredictability" in your defences works to your advantage.

28 RETWEETS

10 FAVORITES

10:19 AM - 26 Feb 13

@thegrugq 26 Feb 2013



the grugq
@thegrugq

@dhw unpredictability is about increasing attacker costs, delaying their operation and increasing their potential for errors.

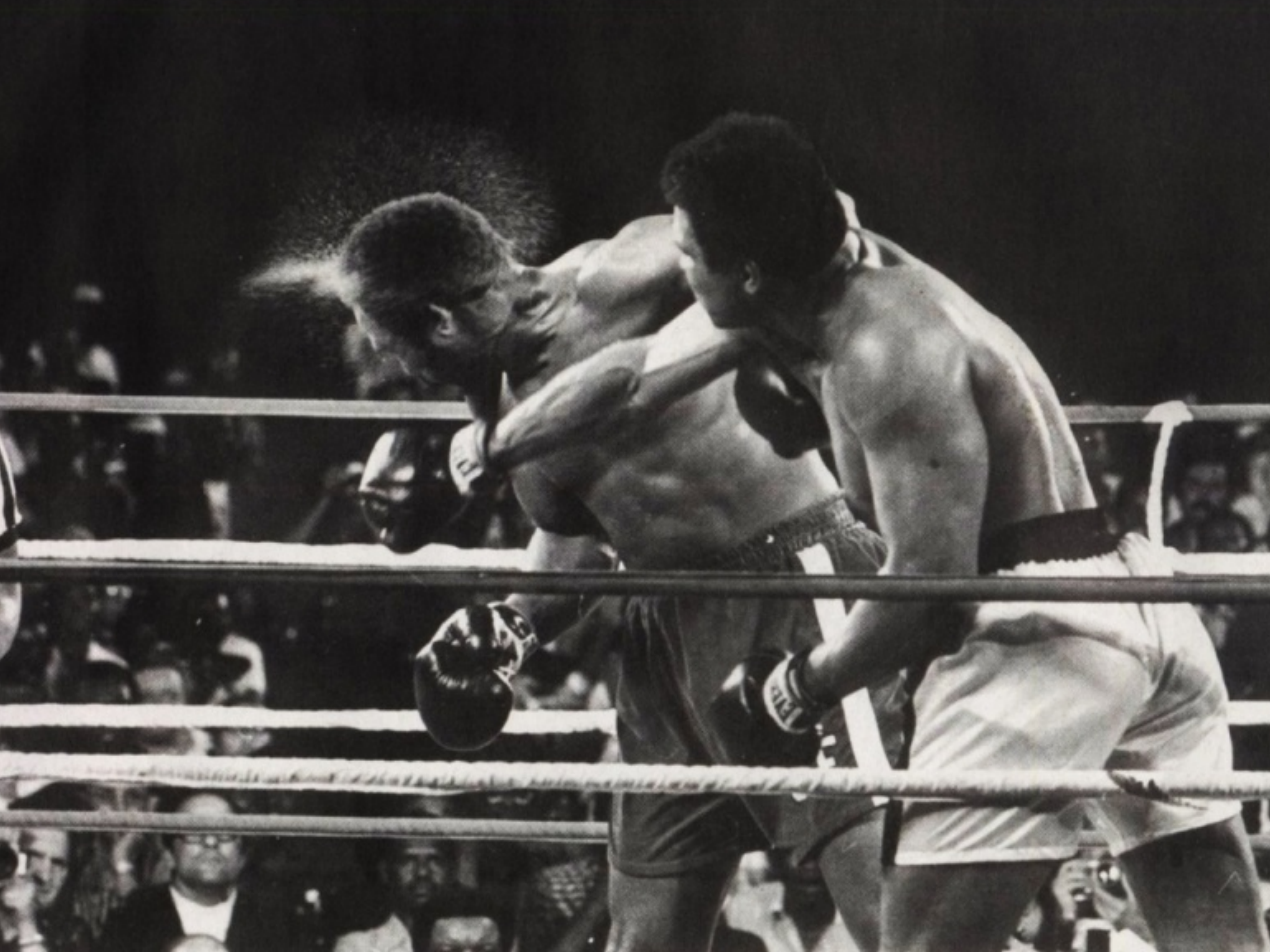
10:26 AM - 26 Feb 13

@thegrugq 26 Feb 2013

Increase attacker costs

$ $ $

Waste Attacker time



Prior Art

- When the tables turn (2004)
    - *Roelof Temmingh, Haroon Meer, Charl van der Walt*
    - http://slideshare.net/sensepost/strikeback
- Stopping Automated Attack Tools (2006)
    - *Gunter Ollmann*
    - http://www.technicalinfo.net/papers/StoppingAutomatedAttackTools.html

Prior Art

- mod-security mailing list (2006)
  - Status Code 503 together w/ Retry-After header
  - *Ryan Barnett*
  - http://bb10.com/apache-mod-security-user/2006-12/msg00042.html

```
SecFilterDefaultAction "deny,log,status:503"
SecFilter ".*"
Header set Retry-After "120"
```

#3

[HOW?]

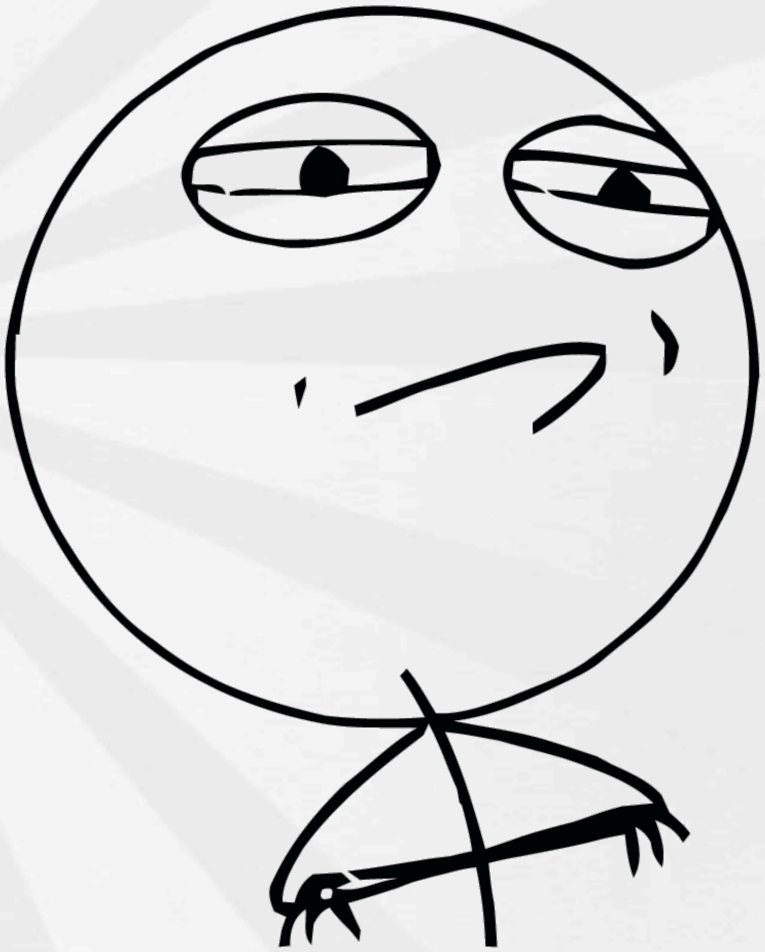
Browsers have to be <u>flexible</u>

THIS LEADS TO INTERPRETATION

… which leads to the dark-side

RFCs…

They're more of a guideline really



What could possibly go wrong!

#3.1

TESTING

THE HOW OF THE THING!

- Restricted research to the big 3
  - Internet Explorer
  - Chrome / Chromium
  - Firefox

No… Safari isn’t in the top ~~10~~ 3

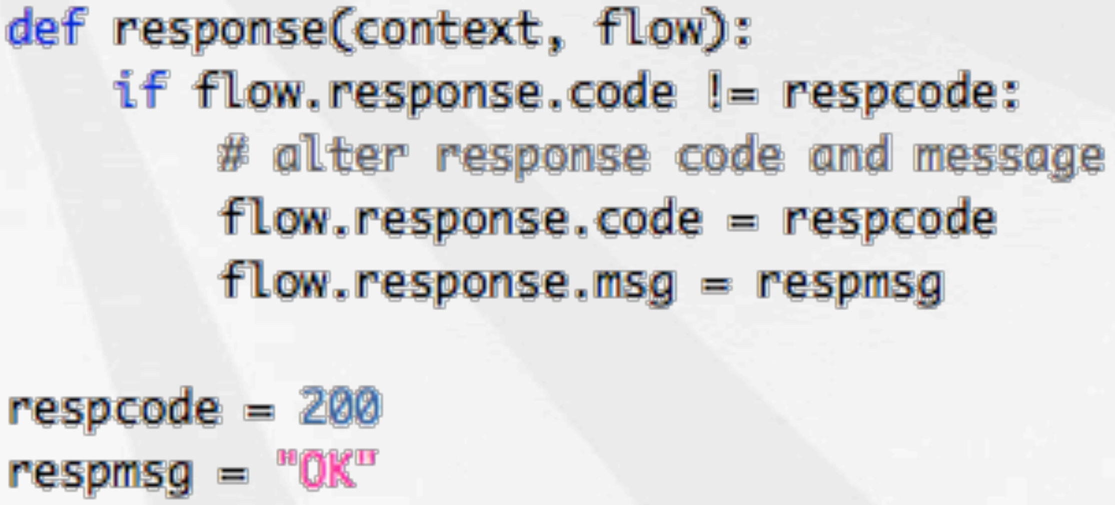
Opera jumped…

…or was it pushed?

Lynx

The unrealistic option

- MITMproxy / MITMdump
  - Python-based
  - Simple to setup proxy / reverse proxy
  - Script-based actions



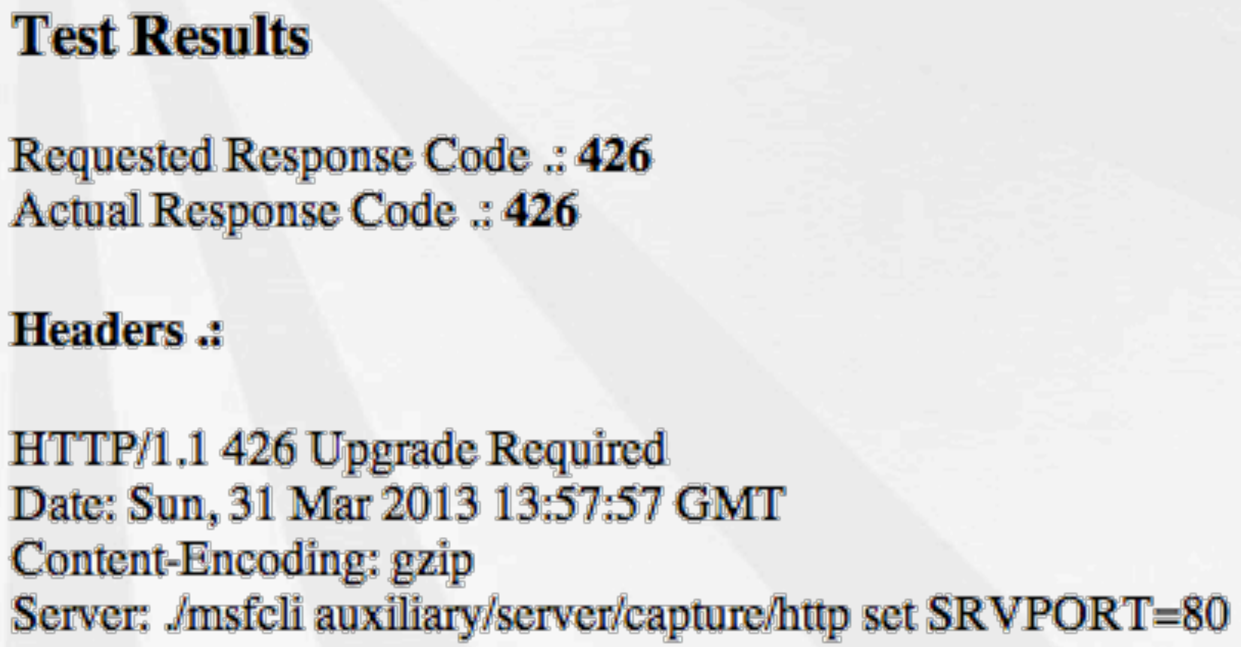
```
def response(context, flow):
    if flow.response.code != respcode:
        # alter response code and message
        flow.response.code = respcode
        flow.response.msg = respmsg

respcode = 200
respmsg = "OK"
```

- PHP
  - Ability to set response code
    - Must be at the top of the PHP code
  - Can be added to php.ini
    - auto-prepend-file = /full/path
  - Limited by web-server (apache)

```
# set response code
Header($_server[“SERVER_PROTOCOL”]. ” $status_code”);
```

- Testing browsers automatically
  - Created PHP file to set status code
    - http://c22.cc/POC/respcode.php?code=*XXX*



Test Results

Requested Response Code .: **426**
Actual Response Code .: **426**

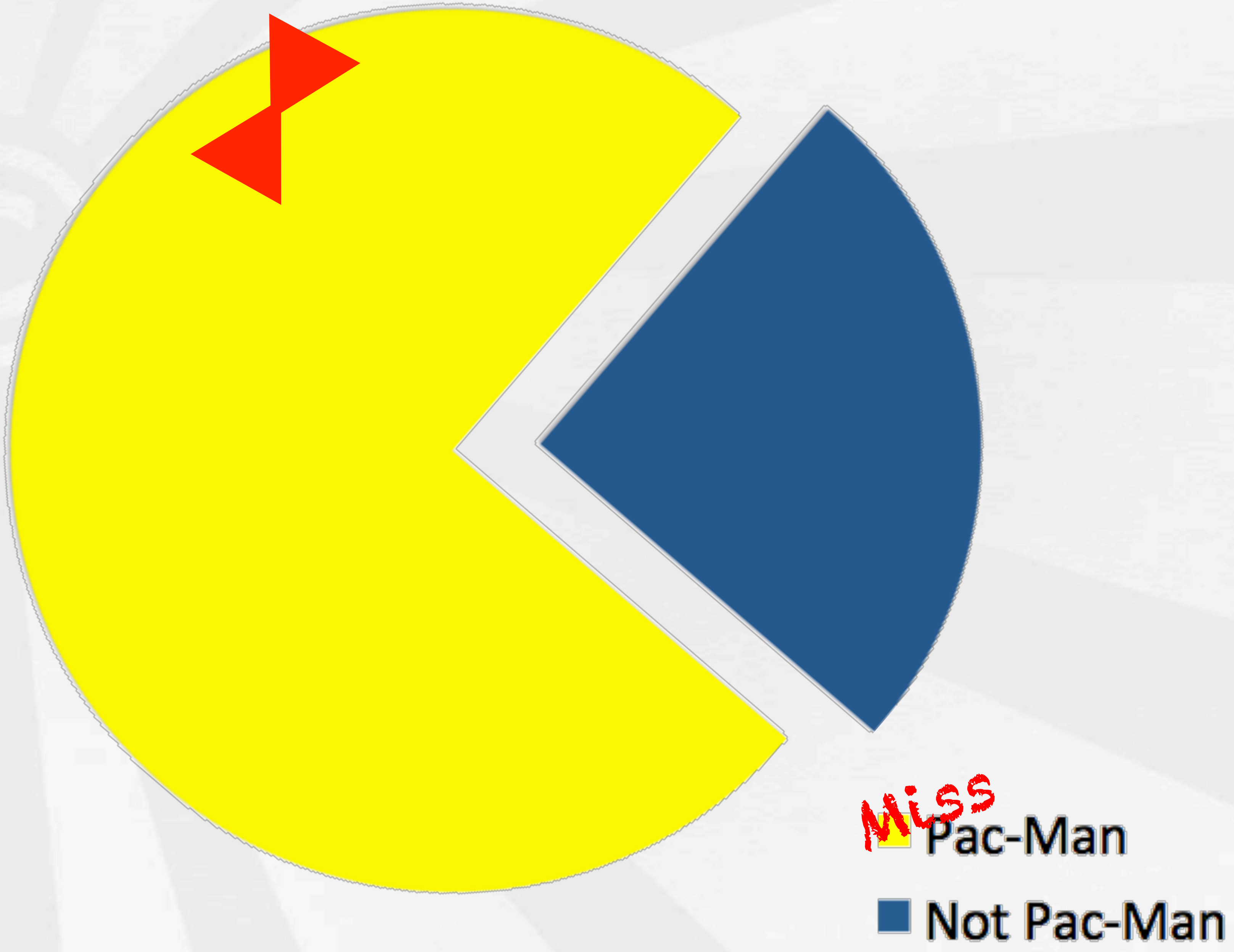
Headers .:

```
HTTP/1.1 426 Upgrade Required
Date: Sun, 31 Mar 2013 13:57:57 GMT
Content-Encoding: gzip
Server: ./msfcli auxiliary/server/capture/http set SRVPORT=80
```

#3.2

BROWSERS

... AND THEIR STATUS CODE HABITS


Miss
🟨 Pac-Man
🟦 Not Pac-Man

Status Code	HTML	iFrame	JavaScript	HTML	iFrame	JavaScript	HTML	iFrame	JavaScript
100	☒	☒	☒	☒	d/load	☒	☒	☒	☒
101	☒	☒	☒	☒	d/load	☒	☒	☒	☒
102	☒	☒	☒	☒	d/load	☒	☒	☒	☒
200									
201									
202									
203									
204	☒	☒	☒	☒	☒	☒	☒	☒	☒
205	☒	☒	☒	☒	☒	☒			
206									
207									

Status Code	HTML	iFrame	JavaScript	HTML	iFrame	JavaScript	HTML	iFrame	JavaScript
300			☒						
301			☒				☒	☒	☒
302			☒				☒	☒	☒
303			☒				☒	☒	☒
304	☒	☒	☒	☒	☒	☒	☒	☒	☒
305			☒						
306			☒						
307			☒				☒	☒	☒

Status Code	HTML	iFrame	JavaScript	HTML	iFrame	JavaScript	HTML	iFrame	JavaScript
400			☒			☒		☒	☒
401			☒			☒			☒
402			☒			☒			☒
403			☒			☒		☒	☒
404			☒			☒		☒	☒
405			☒			☒		☒	☒
406			☒			☒		☒	☒
407			☒	Proxy	Proxy	Proxy			☒
408	☒	☒	☒			☒		☒	☒
409			☒			☒		☒	☒
410			☒			☒		☒	☒
411			☒			☒			☒
↓			☒			☒			☒
426			☒			☒			☒

Status Code	HTML	iFrame	JavaScript	HTML	iFrame	JavaScript	HTML	iFrame	JavaScript
500			☒			☒		☒	☒
501			☒			☒		☒	☒
502			☒			☒			☒
503			☒			☒			☒
504			☒			☒			☒
505			☒			☒		☒	☒
506			☒			☒			☒
507			☒			☒			☒
508			☒			☒			☒
509			☒			☒			☒
510			☒			☒			☒

Loading... Please Wait

Browsers handle most things just like they handle a **200 OK**?

YEP…

MOSTLY

- HTML Responses
  - Almost all response codes are rendered by the browser correctly
- iFrames
  - Some special cases for **IE**, but other browsers handle this the same as HTML

- JavaScript/CSS
  - Limited accepted status codes
    - Limited 3XX support
      - Chrome is the exception here
    - No support for 4XX/5XX codes

So we know what browsers interpret *differently*

What do browsers have in *common*?

- 1XX code handling
  - Retries
  - Confusion
    - Chrome / IE6 try to download the page!
    - Fun on Android… (never ending download)
  - Times outs (eventually)

- 204 No Content
  - Um, no content!
- 304 Not Modified
  - Again, no content returned

#3.3

WHAT ABOUT HEADERS?

Just because the RFC says a specific status code must have an associated header…

…doesn’t mean it *HAS* to

- Redirection codes (301-304, 307)
  - No Location header, no redirect
- 401 Unauthorized
  - No WWW-Authenticate header, no authentication prompt
- 407 Proxy Authentication Required
  - No Proxy-Authenticate header, no prompt

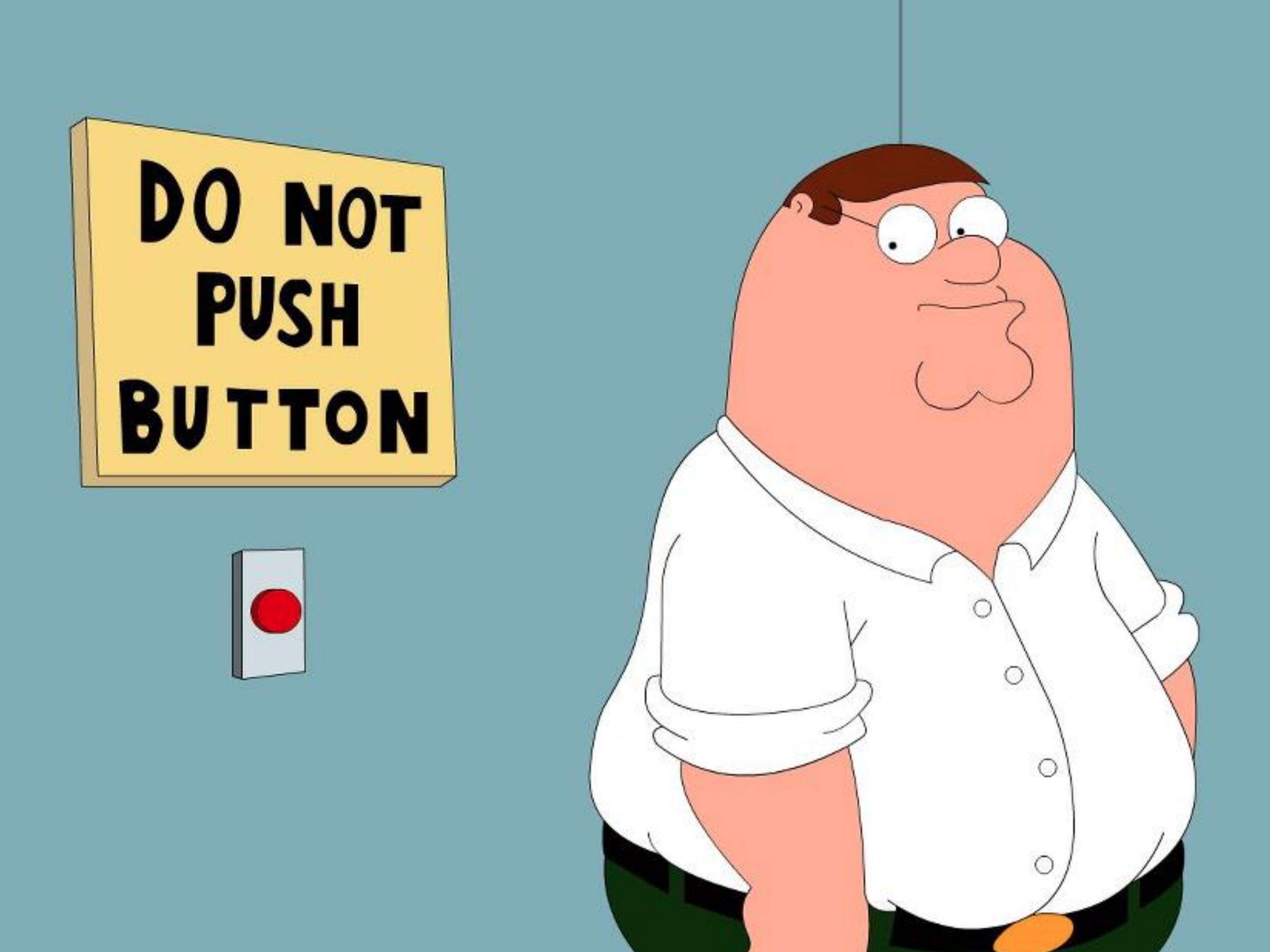
Just because the RFC says a specific status code <u>shouldn't</u> have an associated header…

…doesn’t mean it *can’t*

- 300 Multiple Choices w/ Location Header
  - Firefox / IE6 follows the redirect
  - Chrome doesn't
- More research needed in this direction
  - Most headers are uninteresting / ignored

Each browser handles things a little differently

I wonder what we can do with that!


DO NOT PUSH BUTTON

#4

[GOALS]

- Each browser handles things differently
  - Use known conditions
    - Handled codes
    - Unhandled codes
  - Browser weirdness

#4.1

Browser Fingerprinting

Firefox

- Doesn’t load JavaScript returned with a 300 ‘Multiple Choices’ status code
  - Other browsers tested DO (IE/Chrome)

- Request JavaScript from server
- Response Status: 300 Multiple Choices
- If JavaScript doesn’t run in the browser
  - Firefox

Chrome

- Loads JavaScript returned with a 307 ‘Temporary Redirect’ status code
  - Other browsers tested DON’T (IE/FF)

- Request JavaScript from server
- Response Status: 307 Temporary Redirect
- If JavaScript runs in the browser
  - Chrome

Internet Explorer

- Loads JavaScript returned with a 205 ‘Reset Content’ status code
    - Other browsers tested DON’T (FF/Chrome)

- Request JavaScript from server
- Response Status: 205 Reset Content
- If JavaScript runs in the browser
    - Internet Explorer

Browser Fingerprinting

DEMO

Recycle Bin
Immunity Debugger
WinDbg (X64)
Python (comma...
IDLE (Python GUI)
Notepad++
installers
Google Chrome
Mozilla Firefox
Internet Explorer
11:41 AM
Autopan Disabled

- Other options to fingerprint browsers
  - 300 Redirect (Chrome)
  - 305 / 306 JavaScript (Firefox)
  - 400 iFrame (Internet Explorer)
  - …

POC Script → http://c22.cc/POC/fingerprint.html

User-Agents

can be spoofed

BROWSER TRAITS *CAN’T*

#4.2

Proxy detection

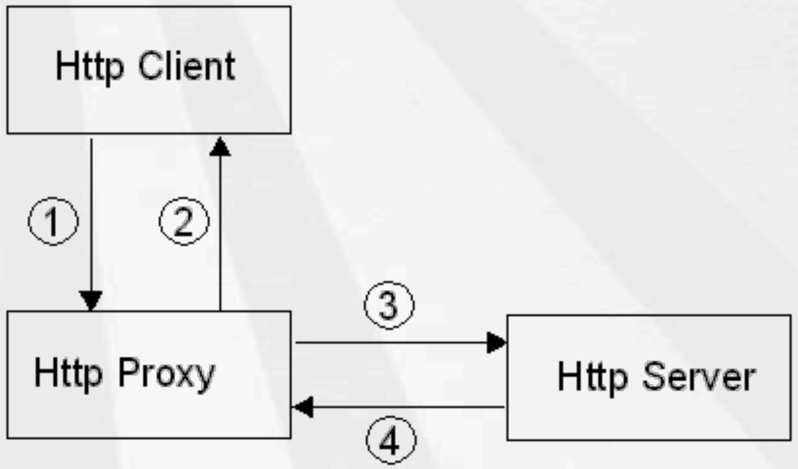
Chrome Proxy Detection

- Chrome handles proxy config differently
    - 407 status code isn't rendered
    - Unless an HTTP proxy is set!

    - Allows us to detect if an HTTP proxy is set
    - Just not which proxy
        - Can only detect HTTP proxies ;(

Chrome Proxy Detection

- Request page from server
- Response Status: 407 Proxy Authentication
  - w/o Proxy-Authenticate header
- If Chrome responds HTTP proxy is set



BONUS STAGE

Side-Effect: Owning Proxies

- Privoxy 3.0.20 (CVE-2013-2503)
  - 407 Proxy Authentication Required
    - w/ Proxy-Authenticate header
  - User prompted for user/pass
    - Prompt appears to be from Privoxy
  - Privoxy passes user/pass to remote site
    - Profit???

Side-Effect: Owning Proxies

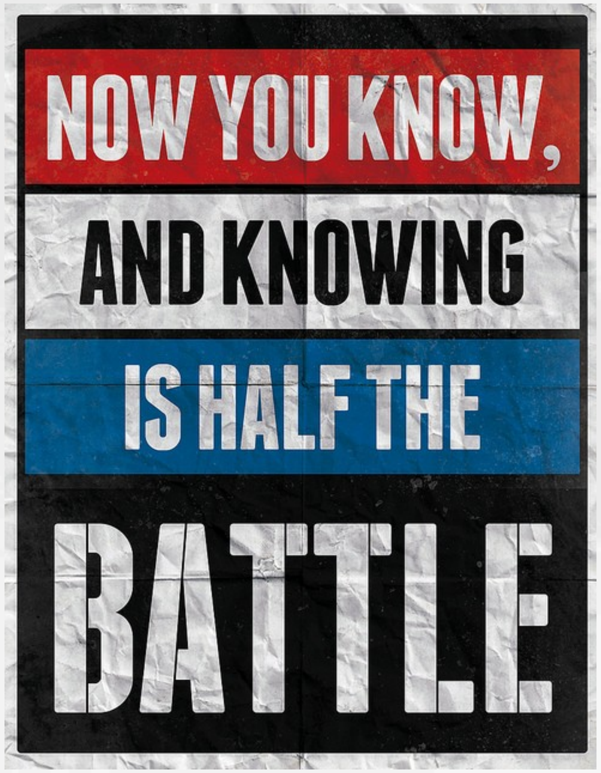
- Not just Privoxy that’s effected
  - Any transparent proxy
    - e.g. Burp, ZAP, …
  - Not really a vuln for most
    - Works as designed!

#5

[BRINGING IT ALL TOGETHER]

What we have

- Status codes all browsers treat as content
- Status codes all browsers can't handle
  - 1XX, etc..
- Lots of browser quirks



What can we do

- F*ck with things
- Screw with scanner monkeys
- Make RFC lovers cry into their beer
- Break things in general

Let’s try to…

- Use what we’ve discovered to…
  - Break spidering tools
  - Cause false positives / negatives
  - Slow down attackers
    - The fun way!
  - Blocking successful exploitation

#5.1

Breaking Spiders

Simplistic view of spiders

- Access target URL
- Read links **/** functions
- Test them out
- If true: continue
  - What is **TRUE**?

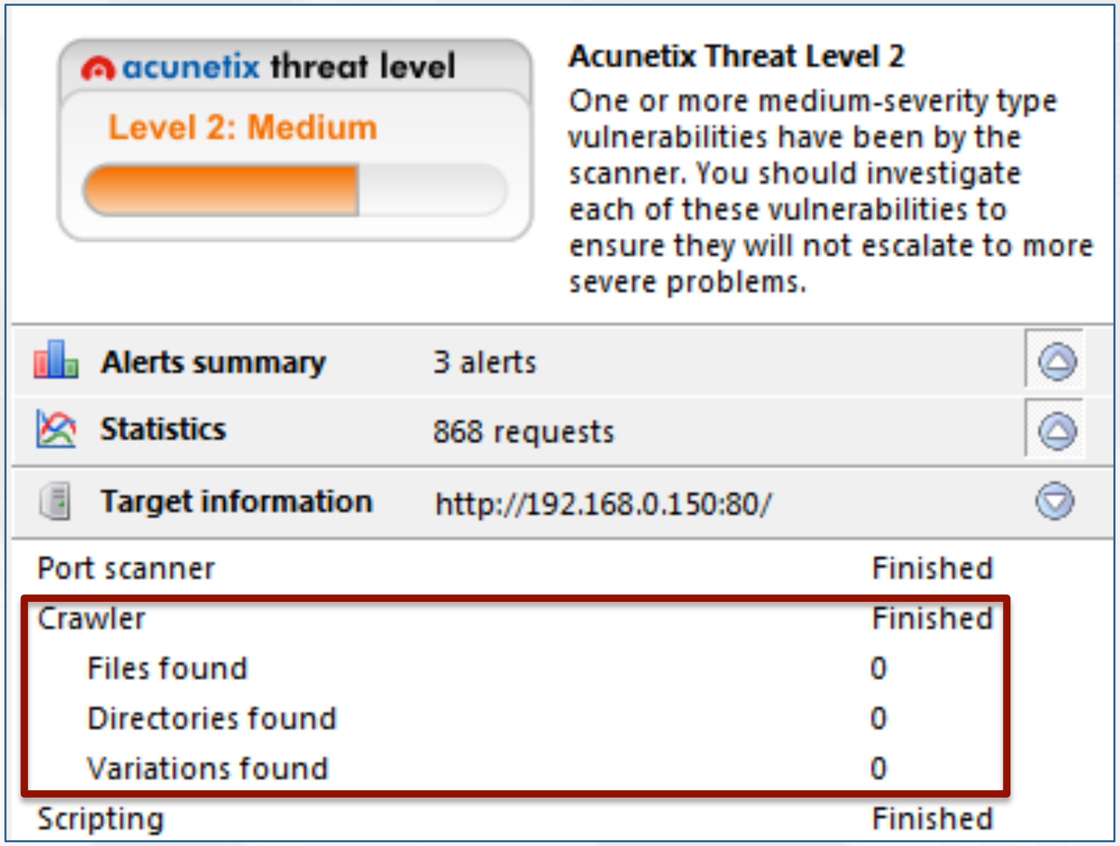
- What happens if:
  - Every response is
    - 200 OK
    - 404 Not Found
    - 500 Internal Server Error

200 OK

- IF 200 == True:
  - Problems!
  - Never-ending spider

404 Not Found

- IF 404 == False:
  - What website?



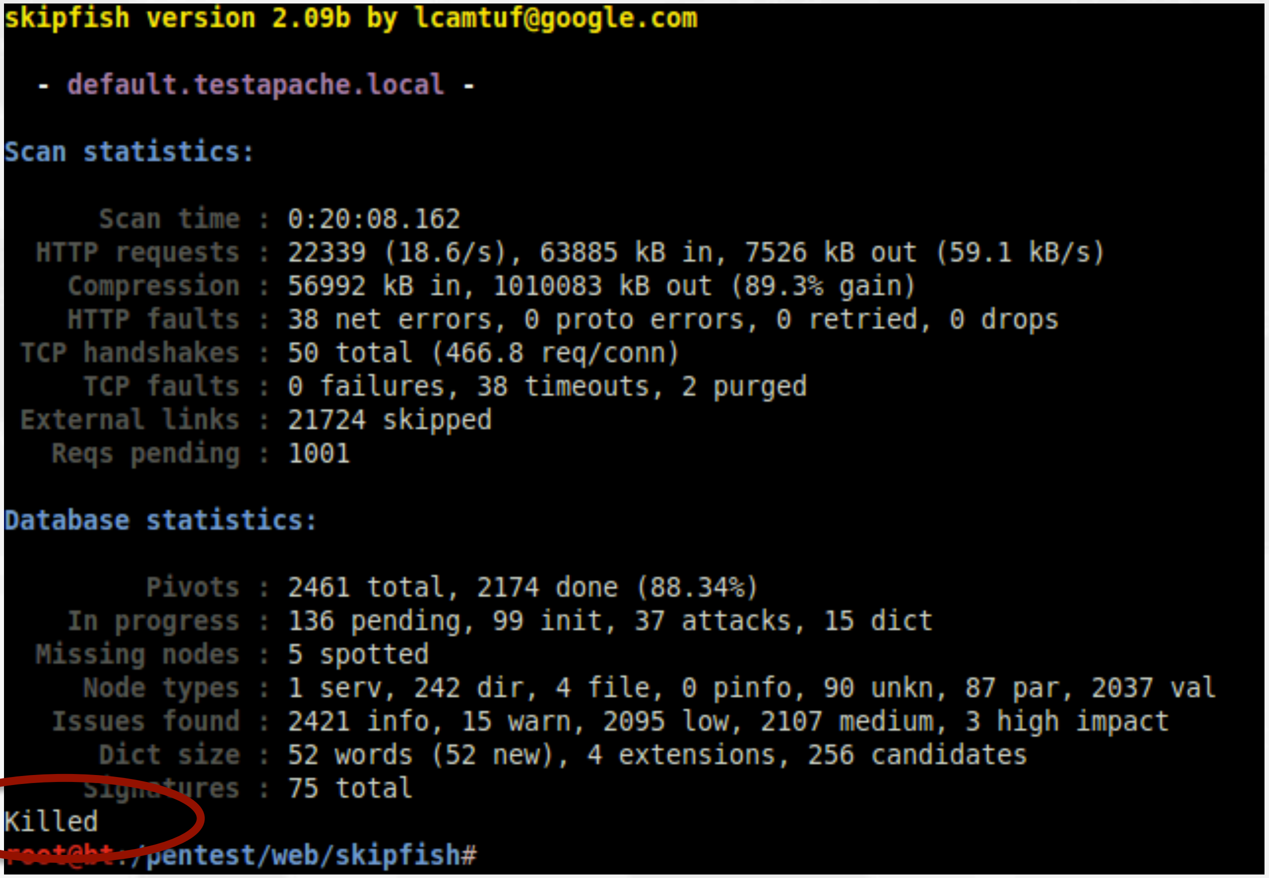
Acunetix Threat Level 2

One or more medium-severity type vulnerabilities have been by the scanner. You should investigate each of these vulnerabilities to ensure they will not escalate to more severe problems.

Alerts summary	3 alerts
Statistics	868 requests
Target information	http://192.168.0.150:80/
Port scanner	Finished
Crawler	Finished
Files found	0
Directories found	0
Variations found	0
Scripting	Finished

500 Internal Server Error

- Skipfish != happy fish



```
skipfish version 2.09b by lcamtuf@google.com

  - default.testapache.local -

Scan statistics:

      Scan time : 0:20:08.162
  HTTP requests : 22339 (18.6/s), 63885 kB in, 7526 kB out (59.1 kB/s)
    Compression : 56992 kB in, 1010083 kB out (89.3% gain)
    HTTP faults : 38 net errors, 0 proto errors, 0 retried, 0 drops
 TCP handshakes : 50 total (466.8 req/conn)
     TCP faults : 0 failures, 38 timeouts, 2 purged
 External links : 21724 skipped
   Reqs pending : 1001

Database statistics:

         Pivots : 2461 total, 2174 done (88.34%)
    In progress : 136 pending, 99 init, 37 attacks, 15 dict
  Missing nodes : 5 spotted
     Node types : 1 serv, 242 dir, 4 file, 0 pinfo, 90 unkn, 87 par, 2037 val
   Issues found : 2421 info, 15 warn, 2095 low, 2107 medium, 3 high impact
      Dict size : 52 words (52 new), 4 extensions, 256 candidates
     Signatures : 75 total
Killed
root@bt:~/pentest/web/skipfish#
```

#5.2

False Positives / Negatives

- Most scanners use status codes
  - At least to some extent
    - Initial match (prior to more costly regex)
    - Speed up detection
    - Easy solution

- What happens if:
  - Every response is
    - 200 OK
    - 404 Not Found
    - 500 Internal Server Error
    - raNd0M*

* Using codes that are accepted by all browsers as content

Vulnerability Baseline

- w3af
  - Information Points → 79
  - Vulnerabilities → 65
  - Shells → 0 shells ☹
  - Scan time → 1h37m23s

Every response 200 OK

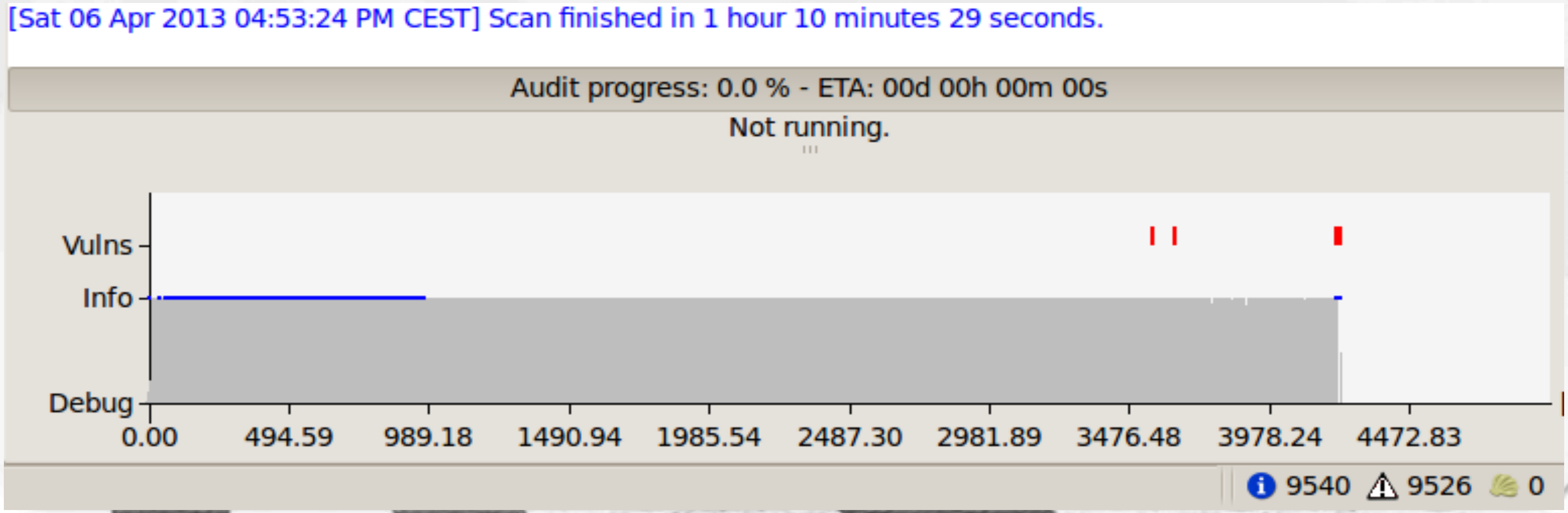
- No change in discoveries
  - All points discovered - per baseline
    - 79 Information Points
    - 65 Vulnerabilities
    - 0 Shells
  - Scan time → 9h56m55s
    - Lots more to check ;)

Every response 404 Not Found

- Less to scan == Less to find
  - False negatives
    - 44 Information Points (-35)
    - 37 Vulnerabilities (-28)
- Scan time → 7m13s
  - Much quicker scan
  - Less paths traversed

Every response 500

- Server Error == OMG VULN SANDWICH!
  - False positives+++
    - 9540 Information points (+9461)
    - 9526 Vulnerabilities (+9461)


[Sat 06 Apr 2013 04:53:24 PM CEST] Scan finished in 1 hour 10 minutes 29 seconds.

Audit progress: 0.0 % - ETA: 00d 00h 00m 00s
Not running.

Random Status Codes

- Multiple test runs
  - All tests produced False positives**++**
    - avg. 619 Information points (+540)
    - avg. 550 Vulnerabilities (+485)
- Avg. scan time → 11m37s
  - Often much quicker scans
  - Lots of variation in scan times

Random Status Codes

- Skipfish + $random_status = chaos
  - False Positives + False Negatives
  - Scan jobs killed (due to lack of scanner resources)
- Scan times
  - 1st scan time → 10h3m35s
  - 2nd scan time → 0h0m4s
  - 3rd scan time → 16h47m41s

#5.3

Slowing attackers down!

What does your **WAF** really do?

- OMG Attack
- Block / Return error
  - 403, 500, …
- Profit???

Why?

Remember that list of status codes *browsers* don't handle well?

Yeah well, *scanners* don’t usually handle them well *either*!

Especially the 1XX codes

- Remember LaBrea tarpit?
  - Tim Liston 2001 *
  - Designed to slow spread of Code Red
  - Slows down scans / attackers

* http://labrea.sourceforge.net

I 1 D 1 E 1 A 1

How about an HTTP Tarpit!

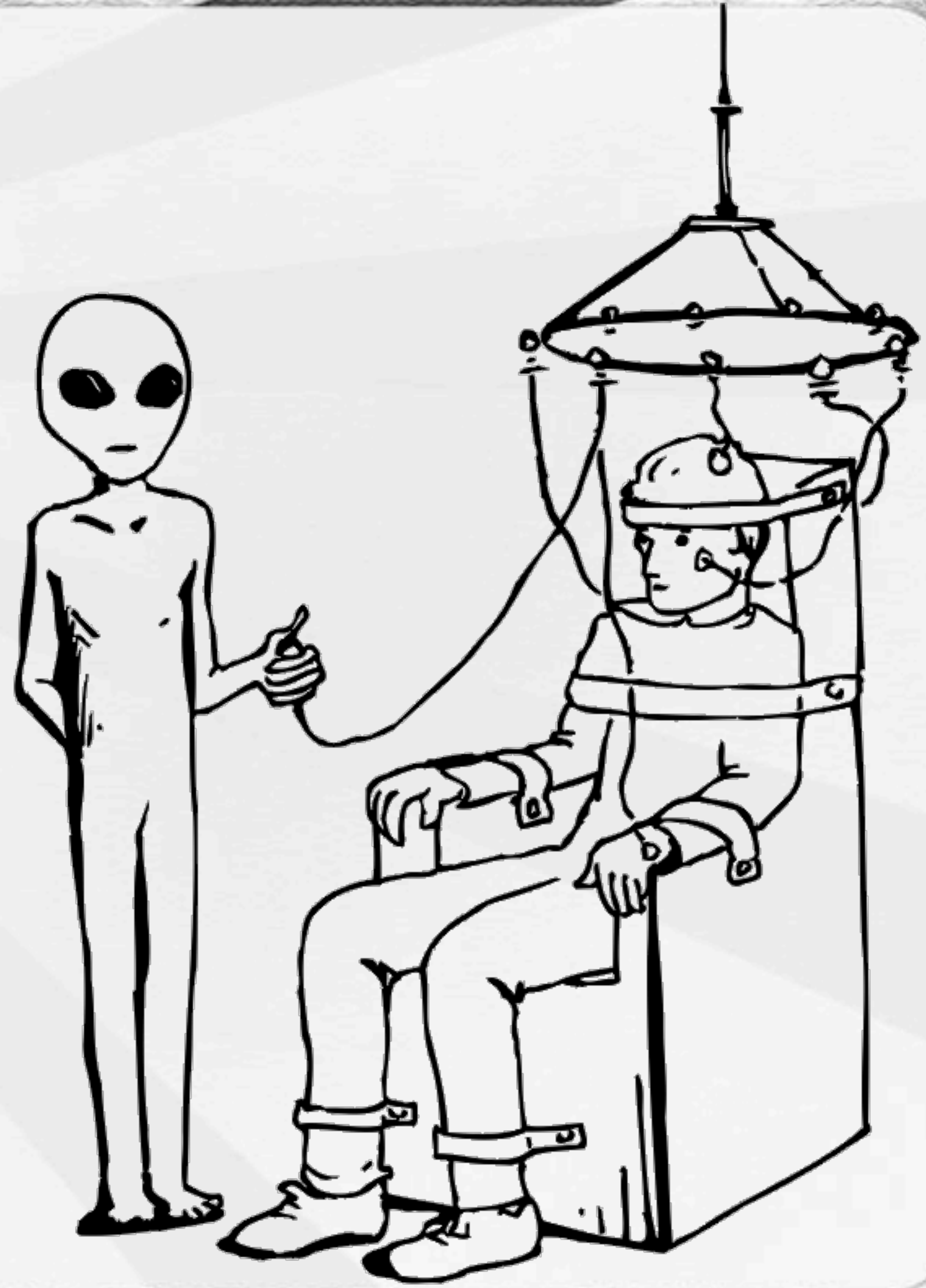
CASE IN POINT by Tom Fishburne

WHERE DID ALL THIS T.A.R. COME FROM?
CaseCentral

HTTP Tarpit Scenario

- WAF detects scan / attack
- Adds source IP to “naughty” list
- Rewrite all responses from the server
  - 100|101|102 status codes only (random)
  - 204|304 might also be useful (no content)

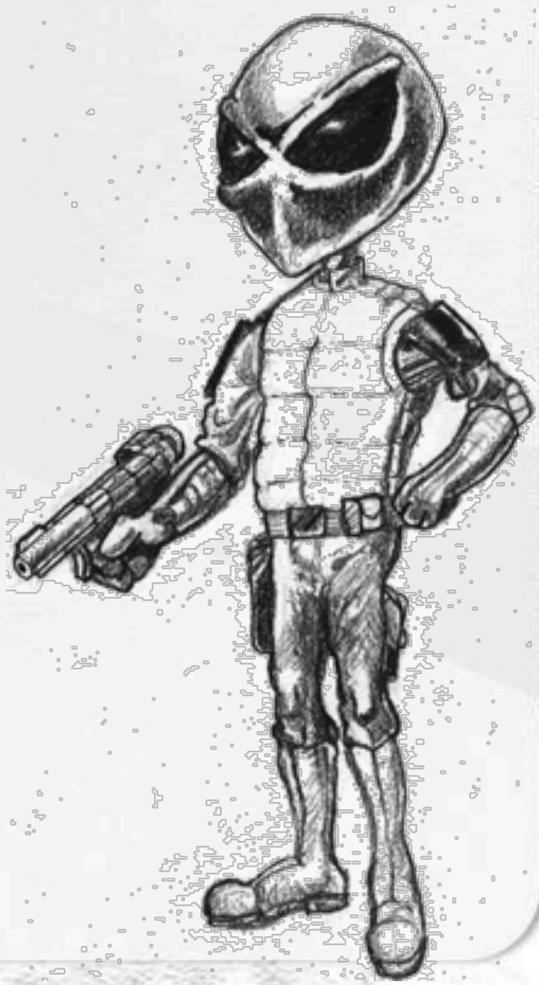
Let's do some science!*



* Science not included

Nikto

vs. the HTTP Tarpit



Baseline	HTTP Tarpit
Scan time	
2m 18s	14h 33m 2s
Findings	
18	10

W3AF

vs. the HTTP Tarpit

Baseline	HTTP Tarpit
Scan time	
1h 37m 23s	18m 10s
Findings	
65	0

SKIPFISH

vs. the HTTP TARPIT

Baseline	HTTP Tarpit
Scan time	
18m 10s	05s
Findings	
Low: 2519 Med: 2522 High: 12	Low: 0 Med: 0 High: 3

Acunetix

vs. the HTTP Tarpit

acunetix

Baseline	HTTP Tarpit
Scan time	
1h 19m	33m
Findings	
Info: 1104 Low: 30 Med: 32 High: 24	Info: 3 Low: 3 Med: 1 High: 0

HTTP Tarpit Results

- HTTP Tarpit Results *
  - Slow down scans
    - Nikto: 340x as long
    - Others give up quicker ;)
  - Unreliable / aborted scans
    - Up to 100% less findings

* Not scientifically sound ;)

#5.4

Blocking successful exploitati0n

We’ve made it hard to *find* the vulnerabilities

We’ve made it *time consuming* for attackers

Now let's stop the sk1dd13s using Metasploit to pop $hells

Q: How often does Metasploit reference status codes?

```
rgrep -E 'res[p|ponse]?\.code' *
```

→ 958 *

* Not scientifically sound ;)

Lots of dependency on status codes*

* yep, even the stuff I wrote

```
if (res.code < 200 or res.code >= 300)
   case res.code
   when 401
      print_warning("Warning: The web site
      asked for authentication: #{res.headers
      ['WWW-Authenticate'] || res.headers
      ['Authentication']}")
   end
   fail_with(Exploit::Failure::Unknown,
   "Upload failed on #{path_tmp}
   [#{res.code} #{res.message}]")
end
```

No match,

No shell*

* exploit dependent

#6

[REVIEW]

- Using status codes to our benefit is fun
  - … and useful!
- Browsers can be quirky
- Scanners / attack toolkits are sometimes set in their ways
  - Take the easy route
  - Easy to fool

- WAFs need to get more offensive about their defense
  - More than just blocking a request
    - Even if you use a snazzy message
  - Hacking back is bad
  - Slowing down known attacks is good
  - Make life harder for skiddies is pricele$$

- Current tools are much the same as APT
  - APT (Adequate Persistent Threat)
  - Only as advanced as they NEED to be

…because screwing with sk1dd13s is fun!

#6.1

Implementation

GHETTO

Ghetto Implementation

- PHP (the lowest common denominator)
  - auto-prepend-file
  - Limited to resources PHP handles
- MITMdump
  - MITMproxy == memory hog
  - Reverse proxy mode

APPROVED

- Usable implementation
  - Nginx as reverse proxy
    - Requires: ngx_lua
    - ngx.status = XXX
    - Bugs in non-git version
      - 203, 305, 306, 414, 505, 506 return *nil*

https://github.com/ChrisJohnRiley/Random_Code/blob/master/nginx/nginx.conf

STAY IN LANE
FUTURE
PAST

- Ease adoption
  - Implement into mod-security
    - Not a simple task
    - Already been discussed many times
    - Help wanted ;)

#6.2

Countering this research

- Less reliance on status codes
- More reliance on content / headers
  - Pros
    - Better matching / intelligence
  - Cons
    - Slower? (regex matching)
    - More resource intensive

Nikola Hadjinikolov ®

Questions?

Code / Scripts available

http://github.com/ChrisJohnRiley/Random_Code

What doesn’t kill you, makes you smaller!

Thanks for coming

http://blog.c22.cc

@ChrisJohnRiley | contact@c22.cc