Defense by Numb3r5

Making problems for script kidd13s and scanner monkeys

@ChrisJohnRiley
“THE WISEST MAN, IS HE WHO KNOWS, THAT HE KNOWS NOTHING”

Socrates: Apology, 21d
edge case
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
WHAT?
HTTP Status Codes
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 Unicode*

* 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!!!!
It started as a simple idea...
... and started to think
SCREW WITH SCANNERS
... AND SCRIPT
K1DD13S
THAT SOUNDS LIKE FUN!
Stop dismissing “obscurity” as a security feature, because “unpredictability” in your defences works to your advantage.
@dhw unpredictability is about increasing attacker costs, delaying their operation and increasing their potential for errors.
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

  - Gunter Ollmann
Prior Art

- mod-security mailing list (2006)
  - Status Code 503 together w/ Retry-After header
  - Ryan Barnett

SecFilterDefaultAction "deny,log,status:503"
SecFilter ".*"
Header set Retry-After "120"
HOW?
BROWSERS HAVE TO BE FLEXIBLE
This leads to interpretation... which leads to the dark-side
RFCs...

They’re more of a guideline really
WHAT COULD POSSIBLY GO WRONG!
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

```python
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)

```php
# set response code
Header($_server["SERVER_PROTOCOL"] . " $status_code");
```
- Testing browsers automatically
  - Created PHP file to set status code

**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
BROWSERS

... AND THEIR STATUS CODE HABITS
<table>
<thead>
<tr>
<th>Status Code</th>
<th>HTML</th>
<th>iFrame</th>
<th>JavaScript</th>
<th>HTML</th>
<th>iFrame</th>
<th>JavaScript</th>
<th>HTML</th>
<th>iFrame</th>
<th>JavaScript</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>d/load</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>101</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>d/load</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>102</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>d/load</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>200</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>201</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>202</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>203</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>204</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>205</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>206</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>207</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Status Code</td>
<td>HTML</td>
<td>iFrame</td>
<td>JavaScript</td>
<td>HTML</td>
<td>iFrame</td>
<td>JavaScript</td>
<td>HTML</td>
<td>iFrame</td>
<td>JavaScript</td>
</tr>
<tr>
<td>-------------</td>
<td>------</td>
<td>--------</td>
<td>------------</td>
<td>------</td>
<td>--------</td>
<td>------------</td>
<td>------</td>
<td>--------</td>
<td>------------</td>
</tr>
<tr>
<td>300</td>
<td></td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>301</td>
<td></td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>302</td>
<td></td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>303</td>
<td></td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>304</td>
<td></td>
<td>✗</td>
<td>✗</td>
<td></td>
<td></td>
<td>✗</td>
<td></td>
<td></td>
<td>✗</td>
</tr>
<tr>
<td>305</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>306</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>307</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Status Code</td>
<td>HTML</td>
<td>iFrame</td>
<td>JavaScript</td>
<td>HTML</td>
<td>iFrame</td>
<td>JavaScript</td>
<td>HTML</td>
<td>iFrame</td>
<td>JavaScript</td>
</tr>
<tr>
<td>-------------</td>
<td>------</td>
<td>--------</td>
<td>------------</td>
<td>------</td>
<td>--------</td>
<td>------------</td>
<td>------</td>
<td>--------</td>
<td>------------</td>
</tr>
<tr>
<td>400</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>401</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>402</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>403</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>404</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>405</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>406</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>407</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Proxy</td>
</tr>
<tr>
<td>408</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Proxy</td>
</tr>
<tr>
<td>409</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Proxy</td>
</tr>
<tr>
<td>410</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Proxy</td>
</tr>
<tr>
<td>411</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Proxy</td>
</tr>
<tr>
<td>426</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Proxy</td>
</tr>
<tr>
<td>Status Code</td>
<td>HTML</td>
<td>iFrame</td>
<td>JavaScript</td>
<td>HTML</td>
<td>iFrame</td>
<td>JavaScript</td>
<td>HTML</td>
<td>iFrame</td>
<td>JavaScript</td>
</tr>
<tr>
<td>-------------</td>
<td>------</td>
<td>--------</td>
<td>------------</td>
<td>------</td>
<td>--------</td>
<td>------------</td>
<td>------</td>
<td>--------</td>
<td>------------</td>
</tr>
<tr>
<td>500</td>
<td></td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>501</td>
<td></td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>502</td>
<td></td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>503</td>
<td></td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>504</td>
<td></td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>505</td>
<td></td>
<td></td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>506</td>
<td></td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>507</td>
<td></td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>508</td>
<td></td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>509</td>
<td></td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>510</td>
<td></td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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
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 shouldn’t 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
GOALS
Each browser handles things differently

- Use known conditions
  - Handled codes
  - Unhandled codes
- Browser weirdness
Browser Fingerprinting
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
- 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
#4.2 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 ;(
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!
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?
500 Internal Server Error

- Skipfish != happy fish

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

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#
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)
Random Status Codes

- Multiple test runs
  - All tests produced False positives++
    - avg. 619 Information points (+540)
    - avg. 550 Vulnerabilities (+485)
- Avg. scan time $\rightarrow$ **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 $\rightarrow$ 10h3m35s
  - 2nd scan time $\rightarrow$ 0h0m4s
  - 3rd scan time $\rightarrow$ 16h47m41s
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
How about an HTTP Tarpit!
WHERE DID ALL THIS T.A.R. COME FROM?
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
<table>
<thead>
<tr>
<th></th>
<th>Baseline</th>
<th>HTTP Tarpit</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scan time</strong></td>
<td>2m 18s</td>
<td>14h 33m 2s</td>
</tr>
<tr>
<td><strong>Findings</strong></td>
<td>18</td>
<td>10</td>
</tr>
</tbody>
</table>
W3AF vs. the HTTP TARPIT
<table>
<thead>
<tr>
<th></th>
<th>Baseline</th>
<th>HTTP Tarpit</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scan time</strong></td>
<td>1h 37m 23s</td>
<td>18m 10s</td>
</tr>
<tr>
<td><strong>Findings</strong></td>
<td>65</td>
<td>0</td>
</tr>
</tbody>
</table>
SKIPFISH vs. the HTTP TARPIT
<table>
<thead>
<tr>
<th></th>
<th>Baseline</th>
<th>HTTP Tarpit</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scan time</strong></td>
<td>18m 10s</td>
<td>05s</td>
</tr>
<tr>
<td><strong>Findings</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low:</td>
<td>2519</td>
<td>Low: 0</td>
</tr>
<tr>
<td>Med:</td>
<td>2522</td>
<td>Med: 0</td>
</tr>
<tr>
<td>High:</td>
<td>12</td>
<td>High: 3</td>
</tr>
</tbody>
</table>
ACUNETIX vs. the HTTP TARPIT
<table>
<thead>
<tr>
<th>Baseline</th>
<th>HTTP Tarpit</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scan time</strong></td>
<td></td>
</tr>
<tr>
<td>1h 19m</td>
<td>33m</td>
</tr>
<tr>
<td><strong>Findings</strong></td>
<td></td>
</tr>
<tr>
<td>Info: 1104</td>
<td>Info: 3</td>
</tr>
<tr>
<td>Low: 30</td>
<td>Low: 3</td>
</tr>
<tr>
<td>Med: 32</td>
<td>Med: 1</td>
</tr>
<tr>
<td>High: 24</td>
<td>High: 0</td>
</tr>
</tbody>
</table>
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 ;)

* HTTP Tarpit Results *

---

* Not scientifically sound ;)

---
Blocking successful exploitation
We’ve made it hard to *find* the vulnerabilities
We’ve made it time consuming for attackers
Now let's stop the sk\textcolor{red}{1dd13}s using Metasploit to pop \textcolor{red}{\$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
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 priceless$$
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!
Implementation
Ghetto Implementation

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

STAY IN LANE

FUTURE

PAST
Ease adoption
  - Implement into mod-security
    - Not a simple task
    - Already been discussed many times
    - Help wanted ;}
Countering this research
- Less reliance on status codes
- More reliance on content / headers
  - Pros
  - Better matching / intelligence
  - Cons
  - Slower? (regex matching)
  - More resource intensive
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