Server fingerprinting

How I broke most famous recon tools and made the script kiddies sad

@0x48piraj

$ whoami

- Undergraduate student at --REDACTED-- University
- Google Code In Contribution Winner, GSoC ‘19 Intern
  OWASP Foundation, Open source enthusiast
- Independent security researcher, have given chronic nightmares to Canon (CVE-2019-14339), Motorola, IIT-B, IIT-M, Cornell, United Nations, Mozilla Firefox (Android & iOS), Opera Mini (Android), NodeJS modules, Samsung, LG Electronics, Dutch & Indian Govt.
- cat > bio.info

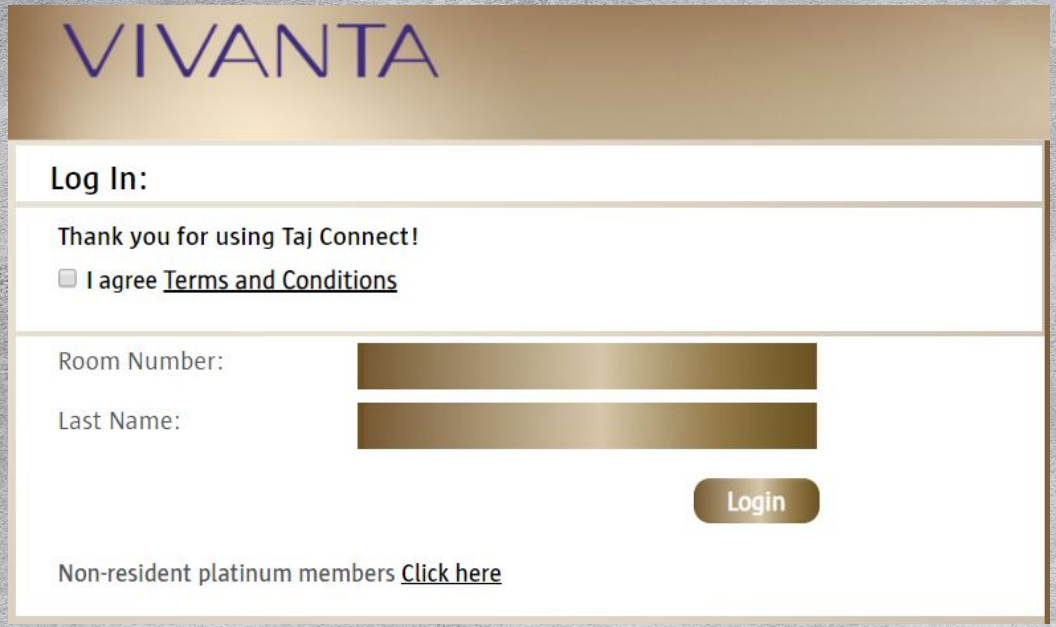
… more uninteresting information goes here

CTRL + Z

[1]+ Stopped cat > bio.info

$ whoami, cont.

Came in yesterday, got in my lovely room (thanks, BSides team), went on to “review” my presentation, my network was struggling with poor signals and thought about using Vivanta’s (SSID: Vivanta_Dwarka) awesome WiFi Network.



$ whoami, cont.

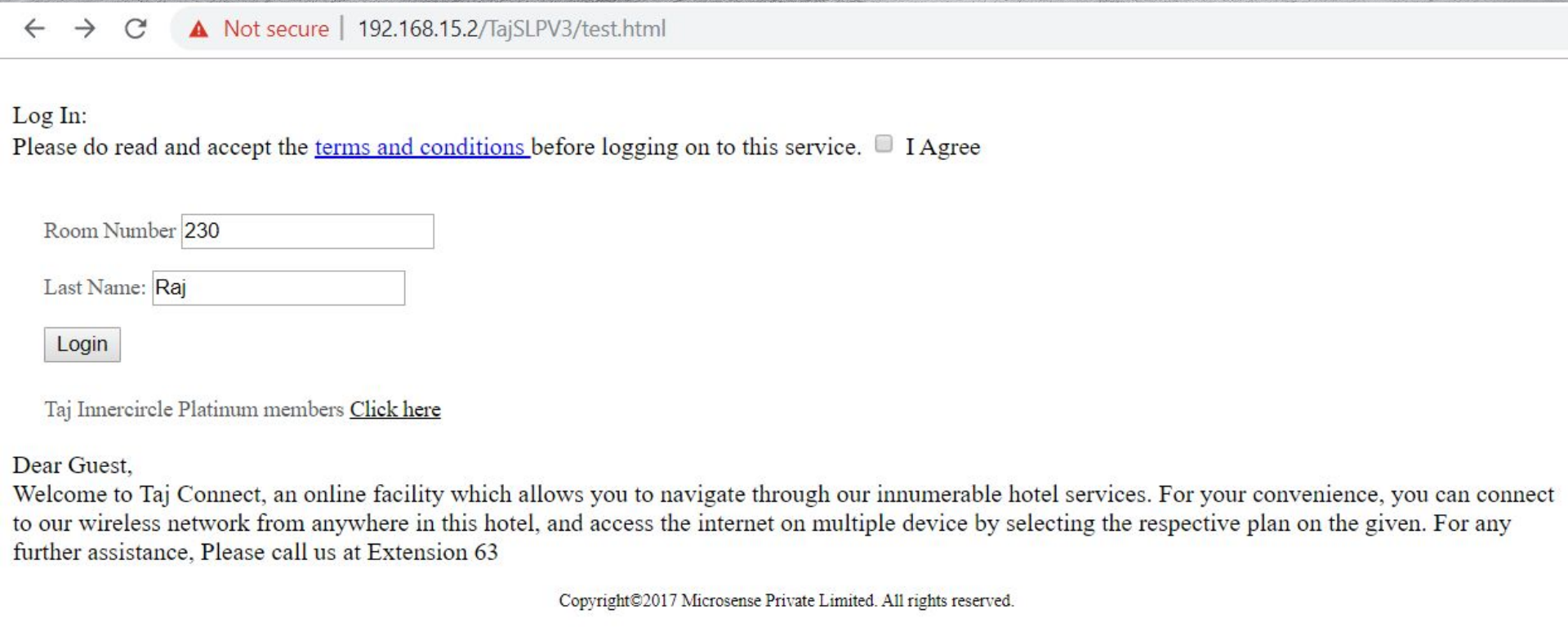
Eh, just room number and last name?

Numerous natural options popped in instantly -

1. Social engineering (I’m looking for “xyz”, can you please help?)
2. Bruteforce attack (is it 203?, 302?, 230? Yes!)
3. Dictionary attack
4. Social engg. + Bruteforce (Excuse me, can you look for “xyz”?)
5. Information leakage (dumpster diving?)
6. Exploring the target web-app looking for vulnz

$ whoami, cont.

left-over test code found during brute-forcing obvious directories.



$ whoami, cont.

Fortunately or unfortunately (depends on your perspective), the error prevented further exploitation.

During testing, I also found Trace.axd which said,

Trace Error

Description: The current trace settings prevent trace.axd from being viewed remotely (for security reasons) It could, however, be viewed by browsers running on the local server machine.

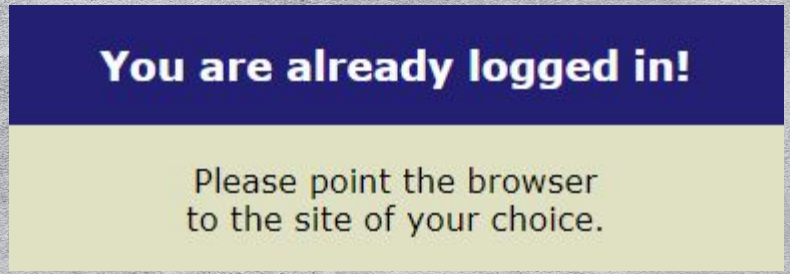
Details: To enable trace.axd to be viewable on remote machines, please create a <trace> tag within the configuration file located in the root directory of the current web application. This <trace> tag should then have its "localOnly" attribute set to "false".

```
<configuration>
    <system.web>
        <trace localOnly="false"/>
    </system.web>
</configuration>
```

$ whoami, cont.

Found multiple endpoints during my assessment -

- 192.168.15.2 (*http://192.168.15.2/TajSLPV3/*)
- 124.153.110.196 (*http://124.153.110.196/TajMemberNew/**)
- 192.168.15.3 (very interesting)
- More ...



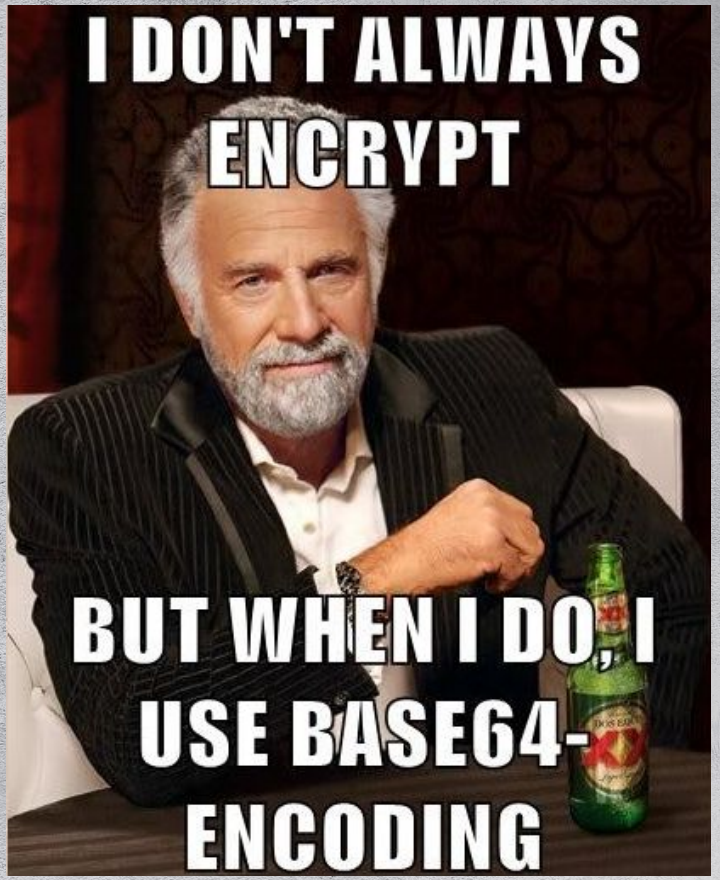
$ whoami, cont.

Essentially, the login structure of the URL was,

http://192.168.15.2/TajSLPV3/mifilogin.aspx?encry=VUk9MDJkMjI2JlVVUkw9aHR0cDovLzE---REDATED---NTkmT1M9aHR0cDovLzE5Mi4xNjguMTUuMy8mU0M9MTg4ODc=

Yeah, so it's basically,

http://192.168.15.2/TajSLPV3/mifilogin.aspx?encry=BASE64_CHUNK



$ whoami, cont.

Decoding the BASE64_CHUNK gave something like,

When Unauthenticated

UI=02d226&UURL=http://192.168.15.3:1111/usg/userok.htm&MA=D0F88C
AAF6AB&RN=80&PORT=80&RAD=no&PP=no&PMS=no&SIP=10.0.2.159&
OS=http://192.168.15.3/&SC=18887

When Authenticated

&UI=02d226&UURL=http://192.168.15.3:1111/usg/userok.htm&MA=D0F88
CAAF6AB&RN=80&PORT=80&RAD=no&PP=no&PMS=no&SIP=10.0.2.110
&OS=http://www.msftconnecttest.com/redirect&SC=25869&logid=7C2A313
960890101201966114382 7

$ whoami, cont.

Demystifying the process flow

1) User connects to the WiFi

2) When certain user launches browser, he/she gets redirected by the Nomadix to Taj's portal page, Nomadix redirects the user through a user ID for the session *(their MAC address)*

Eg: http://IP_ADDRESS:1111/usg/userok.htm?MA=

Yep, the MA parameter holds device's MAC address.

$ whoami, cont.

3) The client website checks to see if certain MAC address has been captured before if it has, immediately makes a POST to Nomadix, if the MAC address has not been captured before then it asks for user info (e.g. Room, Last name).

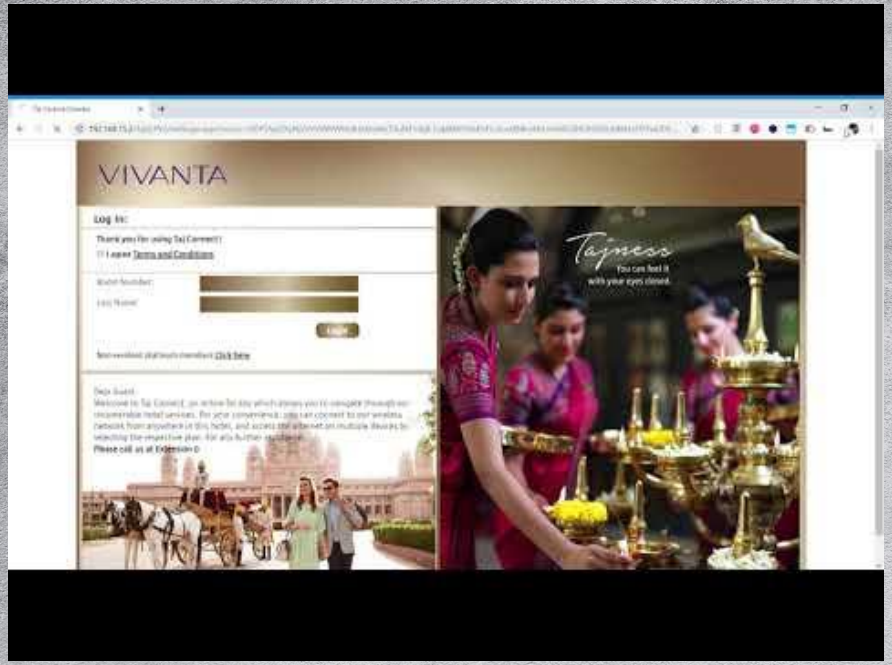
On successful authentication the Web server tells the Nomadix Gateway that certain user can use the internet (service levels are defined). This is done from the web server as a POST to one of the following addresses:

http://:1111/usg/command.xml
http://:1112/usg/command.xml

4) The Nomadix then manages certain user's session until it expires.

$ whoami, cont.

Testing time, capturing the encry= parameter from an authenticated user, i.e. my laptop, decoding, modifying the MAC address, encoding again, then taking another device, and triggering our newly crafted payload results in:

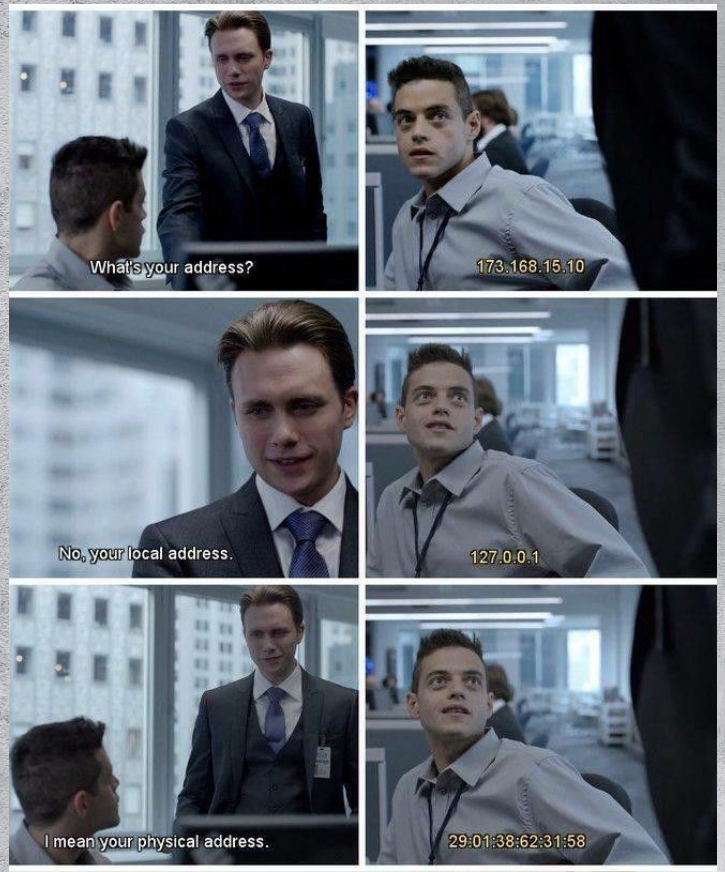


$ whoami, cont.

And we all know how to collect lots and lots of MAC addresses, as MAC addresses are sent unencrypted. The reason for this is, MAC addresses are part of the OSI Data Link layer (level 2) and are visible in packets even if encryption such as WEP / WPA2 is used.

Commands:

```
wlan=`iw dev | awk '$1=="Interface"{print $2}'`
tshark -a duration:100 -i $wlan -T fields -e
eth.src | sort | uniq
```



$ whoami, cont.

The following “greet message” pretty much covers as our Addendum

Dear Guest,

Welcome to Taj Connect, an online facility which allows you to navigate through our *innumerable hotel services*. For your convenience, you can connect to our wireless network from anywhere in this hotel, and access the internet on multiple devices by selecting the respective plan on the given device.


WARNING
NOW ENTERING THE PONY ZONE

This talk contains

- How famous reconnaissance tools work
- Current fingerprinting methods
- Bad jokes, obviously
- Screwing with script kiddies

  (everyone's dream, right?)

This talk contains

- How famous reconnaissance tools work

Providing
"defensive strategies"
to companies

DISCLAIMER

This research presented herein was conducted and completed as an independent researcher, none of the research presented herein was conducted under the auspices of my current organisation.

The views, information and opinions expressed in this presentation and it's associated research paper are mine only and do not reflect the views, information or opinions of my current organisation.

So, what's going on?

There are several different vendors and versions of web servers on the market today.

Knowing the version and type of a running web server allows testers to determine known vulnerabilities and the appropriate exploits to use during testing.

Okay, but, what is server fingerprinting?

That very process is called web application fingerprinting, identifying the type/version of the web server/application.

More on “how”

By knowing how each type of web server responds to specific payload (i.e. request) and keeping this information in a fingerprint database, a tester can send these payloads to the web server, analyze the response, and compare it to the database of known signatures revealing useful information. (i.e. tech-stack/server/app/OS)

Let’s not confuse

Many types of fingerprinting methods.

- Operating system or say, TCP/IP stack (Nmap)
- Web-stack (Wappalyzer, BuiltWith, etc.)
- Protocol based
  - e.g. HTTP protocol behaviour (best*)
  - e.g. HTTP response header
- --INSERT--

*with statistical analysis, combined with fuzzy logic techniques

Let’s talk about OS fingerprinting

Collection of configuration attributes from a remote device during standard layer 4 network communications. The combination of parameters may then be used to infer the remote machine's operating system. Two types of methods are used,

1. Active fingerprinting
2. Passive fingerprinting

Active/Passive

Active fingerprinting is accomplished by sending specially crafted packets to a target machine and then noting down its response and analyzing the gathered information to determine the target OS.

Passive fingerprinting is based on sniffing traces from the remote system and determining the operating system of the remote host.
(not used by Nmap)

TCP/IP fields which are used in fingerprinting,

- Initial packet size (16 bits)
- Initial TTL (8 bits)
- Window size (16 bits)
- Max segment size (16 bits)
- Window scaling value (8 bits)
- "don't fragment" flag (1 bit)
- "sackOK" flag (1 bit)
- "nop" flag (1 bit)

Just inspecting the Initial TTL and window size fields is often enough in order to successfully identify an operating system.

Generalized Nmap Working

Nmap fingerprints a system in three steps,

- Performs a port scan to find a set of open and closed TCP and UDP ports.
- Generates specially formed packets, sends them to the remote host, and listens for responses. (performs 7 tests)
- Uses the results from the tests to find a matching entry in its database of fingerprints.

Fooling OS fingerprinting tools ‘how-to’

For fooling Nmap, or any other OS fingerprinting tool, we will have to make patches to the Linux kernel because the aim is to change Linux TCP/IP stack behavior, and if we want to achieve it, we need to do it in the kernel layer.

Existing Research

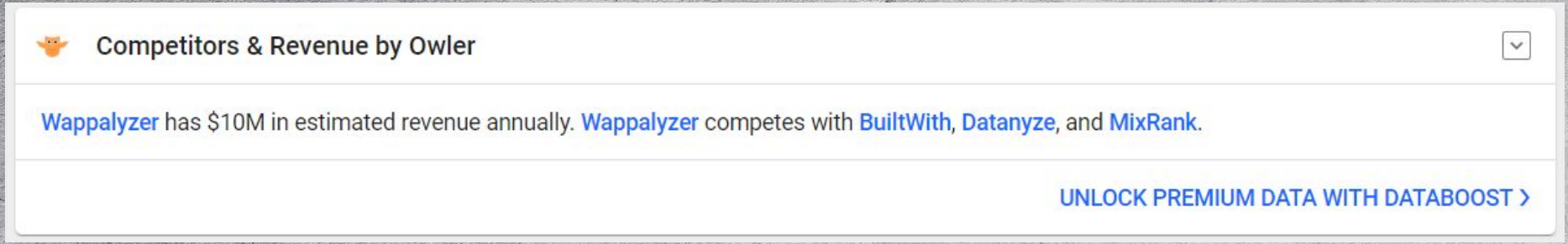
- A practical approach for defeating Nmap OS-Fingerprinting by David Barroso Berrueta (Google “defeat-nmap-osdetect”)
- Defeating TCP/IP Stack Fingerprinting by Matthew Smart G., Robert Malan and Farnam Jahanian from University of Michigan (9th USENIX Security Symposium Paper 2000)

There are cool modules available such as IP Personality (don’t use Fingerprint fucker) which work out-of-the-box so writing from scratch doesn’t makes too much sense.

Wappalyzer

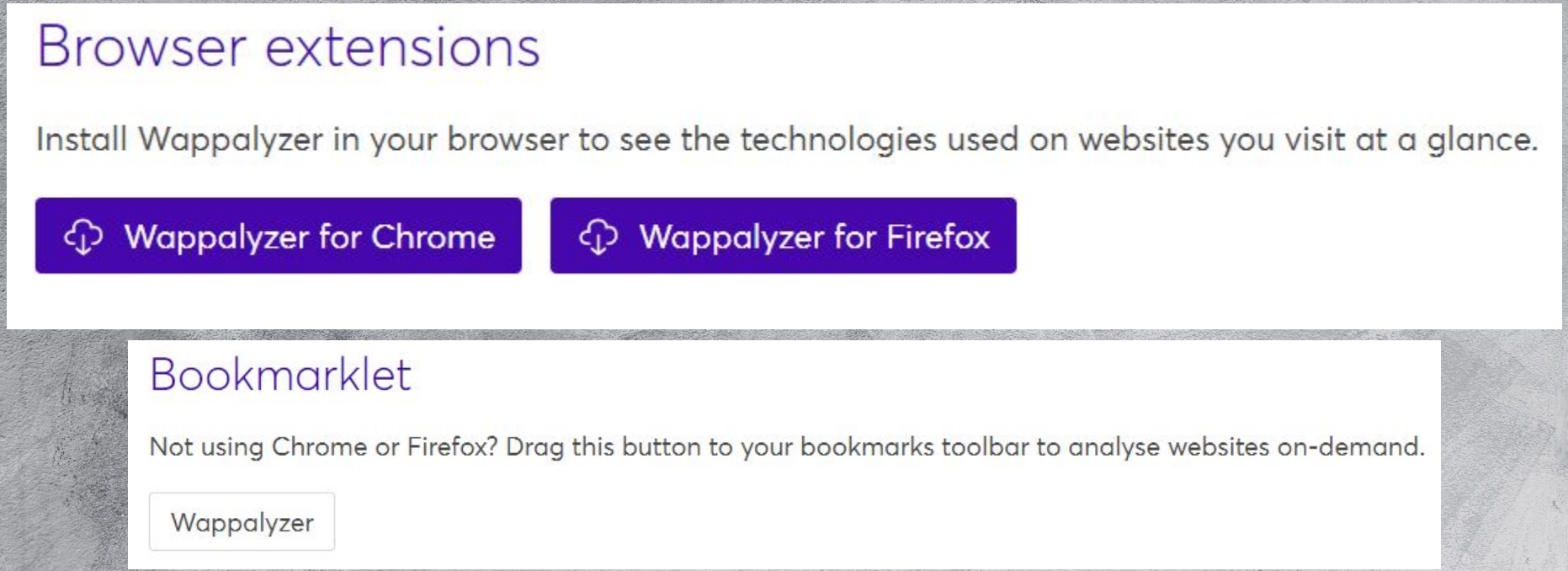
Wappalyzer is a cross-platform utility that uncovers the technologies used on websites.

In the last six months, Wappalyzer identified 1,139 technologies
42,189,411,946 times on **63,214,050** websites



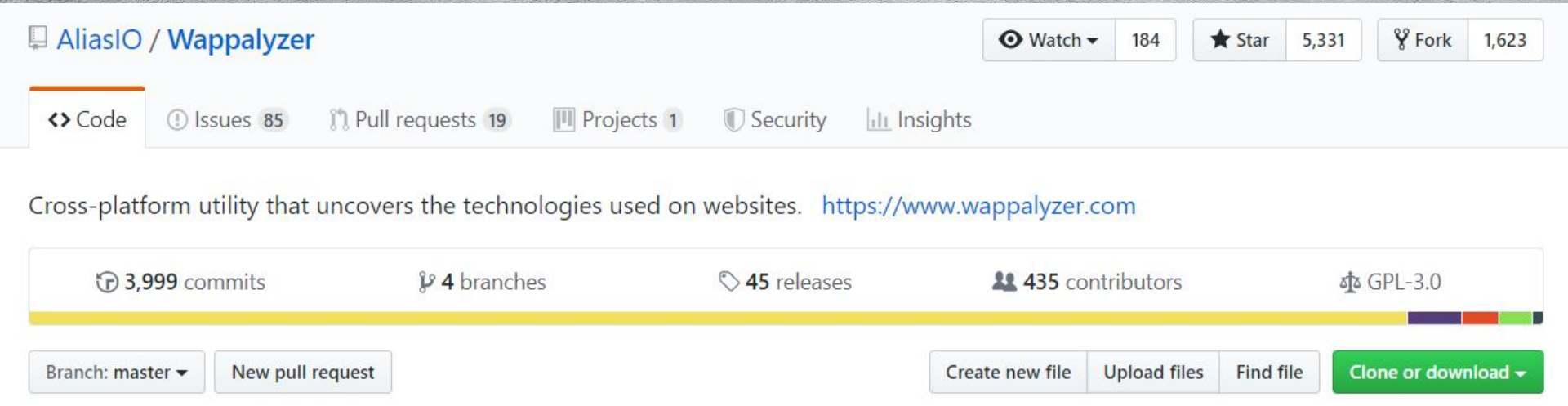
The favourite tool for script kiddies

Wappalyzer has Chrome and Firefox extensions.



Open-source <3

Wappalyzer is free and open-source. The source code is available under the GPL v3 licence.



Deep diving into the code

- Code structure of the Wappalyzer
- Understanding the repository at high level
- Internal workings
- Finding vulnerabilities
- Attacking the application

Structure

```
Dell@DESKTOP-H0F6GFV MINGW64 ~/Desktop/wapparalyser/Wappalyzer (master)
$ ls -la
total 96
drwxr-xr-x 1 Dell 197121     0 Oct 10 01:23 ./
drwxr-xr-x 1 Dell 197121     0 Oct 10 00:48 ../
-rw-r--r-- 1 Dell 197121   156 Oct 10 01:23 .editorconfig
-rw-r--r-- 1 Dell 197121    24 Oct 10 01:23 .eslintignore
-rw-r--r-- 1 Dell 197121   102 Oct 10 01:23 .eslintrc.js
drwxr-xr-x 1 Dell 197121     0 Oct 10 01:23 .git/
-rw-r--r-- 1 Dell 197121   441 Oct 10 01:23 .gitattributes
-rw-r--r-- 1 Dell 197121   214 Oct 10 01:23 .gitignore
-rw-r--r-- 1 Dell 197121   767 Oct 10 01:23 .travis.yml
drwxr-xr-x 1 Dell 197121     0 Oct 10 01:23 bin/
drwxr-xr-x 1 Dell 197121     0 Oct 10 01:23 build/
-rw-r--r-- 1 Dell 197121   616 Oct 10 01:23 Dockerfile
-rw-r--r-- 1 Dell 197121   684 Oct 10 01:23 issue_template.md
-rw-r--r-- 1 Dell 197121 35819 Oct 10 01:23 LICENSE
-rw-r--r-- 1 Dell 197121 62382 Oct 10 01:23 npm-shrinkwrap.json
-rw-r--r-- 1 Dell 197121   397 Oct 10 01:23 package.json
drwxr-xr-x 1 Dell 197121     0 Oct 10 01:23 patches/
-rw-r--r-- 1 Dell 197121  1098 Oct 10 01:23 README.md
-rwxr-xr-x 1 Dell 197121   467 Oct 10 01:23 run*
-rw-r--r-- 1 Dell 197121  1883 Oct 10 01:23 schema.json
drwxr-xr-x 1 Dell 197121     0 Oct 10 01:23 src/

Dell@DESKTOP-H0F6GFV MINGW64 ~/Desktop/wapparalyser/Wappalyzer (master)
$ ls -la src/drivers/webextension/
total 20
drwxr-xr-x 1 Dell 197121    0 Oct 10 01:23 ./
drwxr-xr-x 1 Dell 197121    0 Oct 10 01:23 ../
-rw-r--r-- 1 Dell 197121  103 Oct 10 01:23 .gitignore
drwxr-xr-x 1 Dell 197121    0 Oct 10 01:23 _locales/
drwxr-xr-x 1 Dell 197121    0 Oct 10 01:23 css/
drwxr-xr-x 1 Dell 197121    0 Oct 10 01:23 html/
drwxr-xr-x 1 Dell 197121    0 Oct 10 01:23 images/
drwxr-xr-x 1 Dell 197121    0 Oct 10 01:23 js/
-rw-r--r-- 1 Dell 197121 1897 Oct 10 01:23 manifest.json
-rw-r--r-- 1 Dell 197121  356 Oct 10 01:23 npm-shrinkwrap.json
-rw-r--r-- 1 Dell 197121   66 Oct 10 01:23 package.json
-rw-r--r-- 1 Dell 197121  280 Oct 10 01:23 yarn.lock

Dell@DESKTOP-H0F6GFV MINGW64 ~/Desktop/wapparalyser/Wappalyzer (master)
$
```

Deep diving into the code

Interesting files to look into:

- Wappalyzer/src/wappalyzer.js
- Wappalyzer/src/apps.json
- Wappalyzer/src/drivers/*.*
  - Wappalyzer/src/drivers/webextension/js/*.*

Deep diving into the code

Wappalyzer/src/wappalyzer.js

```
/**
 * Parse apps.json patterns
 */
parsePatterns(patterns) {
  if (!patterns) {
    return [];
  }

  let parsed = {};

  // Convert string to object containing array containing string
  if (typeof patterns === 'string' || patterns instanceof Array) {
    patterns = {
      main: asArray(patterns),
    };
  }

  Object.keys(patterns).forEach((key) => {
```

```
/**
 * Analyze HTML
 */
analyzeHtml(app, html) {
  const patterns = this.parsePatterns(app.props.ht

  if (!patterns.length) {
    return Promise.resolve();
  }

  return asyncForEach(patterns, (pattern) => {
    if (pattern.regex.test(html)) {
      addDetected(app, pattern, 'html', html);
    }
  });
}
```

```
/**
 * Calculate confidence total
 */
getConfidence() {
  let total = 0;

  Object.keys(this.confidence).forEach((id) => {
    total += this.confidence[id];
  });

  this.confidenceTotal = Math.min(total, 100);

  return this.confidenceTotal;
}
}
```

```
/**
 * Analyze script tag
 */
analyzeScripts(app, scripts) {
  const patterns = this.parsePatterns(app.props.script);

  if (!patterns.length) {
    return Promise.resolve();
  }

  return asyncForEach(patterns, (pattern) => {
    scripts.forEach((uri) => {
      if (pattern.regex.test(uri)) {
        addDetected(app, pattern, 'script', uri);
      }
    });
```

```
    /**
     * Analyze meta tag
     */
    metaTags) {
    = this.parsePatterns(app.props.meta);
    = [];

    meta) {
    e.resolve();

    h((match) => {
    atterns).forEach((meta) => {
    ew RegExp(`(?:name|property)=["']${meta}["']`, 'i');

    match)) {
    tent = match.match(/content=("|')([^"']+)("|')/i);

    push(asyncForEach(patterns[meta], (pattern) => {
    tent && content.length === 4 && pattern.regex.test(content[2])) {
    tected(app, pattern, 'meta', content[2], meta);

    all(promises);
```

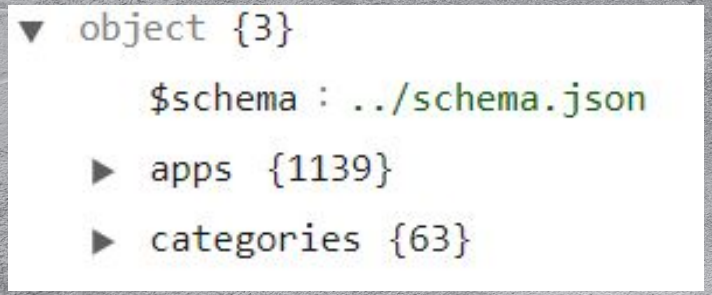
Deep diving into the code

Wappalyzer/src/apps.json

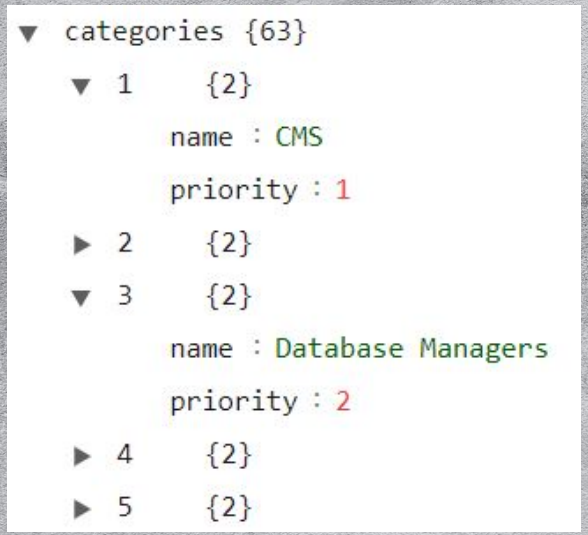
```
 GNU nano 3.1                                    apps.json

{
  "$schema": "../schema.json",
  "apps": {
    "1C-Bitrix": {
      "cats": [
        1
      ],
      "headers": {
        "Set-Cookie": "BITRIX_",
        "X-Powered-CMS": "Bitrix Site Manager"
      },
      "html": "(?:<link[^>]+components/bitrix|(?:src|href)=\"/bitrix/(?:js|templates))",
      "icon": "1C-Bitrix.png",
      "implies": "PHP",
      "script": "1c-bitrix",
      "website": "http://www.1c-bitrix.ru"
    },
    "91App": {
      "cats": [
        6
      ],
      "icon": "91app.png",
      "script": "https\\:\\/\\/track\\.91app\\.io\\/track\\.js\\?",
      "website": "https://www.91app.com/"
    },
    "3dCart": {
      "cats": [
        1,
        6
      ],
      "cookies": {
        "3dvisit": ""
      },
      "headers": {
        "X-Powered-By": "3DCART"
      },
      "icon": "3dCart.png",
      "script": "(?:twlh(?:track)?\\.asp|3d_upsell\\.js)",
      "website": "http://www.3dcart.com"
    },
    "A-Frame": {
      "cats": [
        25
      ],
      "html": "<a-scene[^<>]*>",
      "icon": "A-Frame.svg",
      "implies": "three.js",
      "js": {
        "AFRAME.version": "^(.+)$\\;version:\\1"
      },
      "script": "/?([\\d.]+)?/aframe(?:\\.min)?\\.js\\;version:\\1",
      "website": "https://aframe.io"
    },
                                [ Read 13935 lines ]
^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos  M-U Undo  M-A Mark Text  M-] To Bracket  M-Q Previous  ^B Back  ^◄ Prev Word
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell ^_ Go To Line M-E Redo  M-6 Copy Text  ^Q Where Was  M-W Next  ^F Forward  ^► Next Word
```

Structure of apps.json

root node



categories sub-node



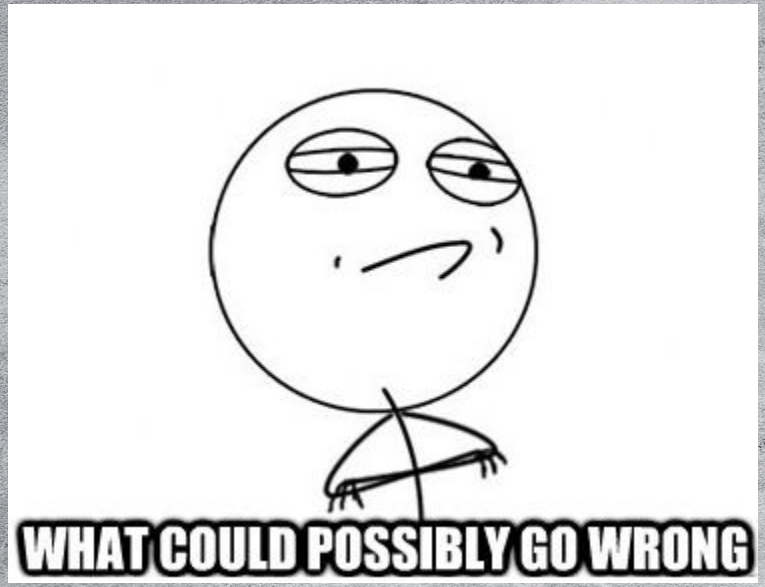
apps sub-node

Pattern matching is CoOL

From seeing the code, it's trivial to conclude that the application uses custom regex/regexp to parse specific features, validates using apps.json which holds lots and lots of "patterns", uses fancy function named getConfidence() to calculate the confidence score.

- AliasIO/Wappalyzer/src/wappalyzer.js#L105

???


WHAT COULD POSSIBLY GO WRONG

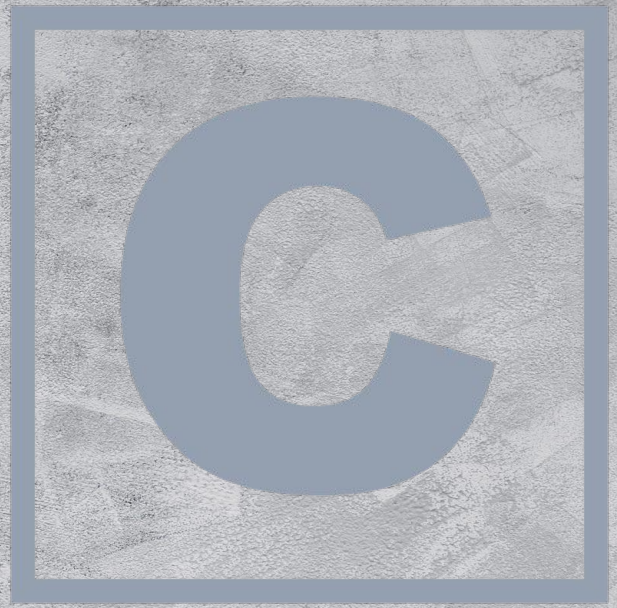
Testing slash Fuzzing

Which programming language to write the fuzzer ?

- C, Java
- Javascript
- PHP
- Python

Choosing language

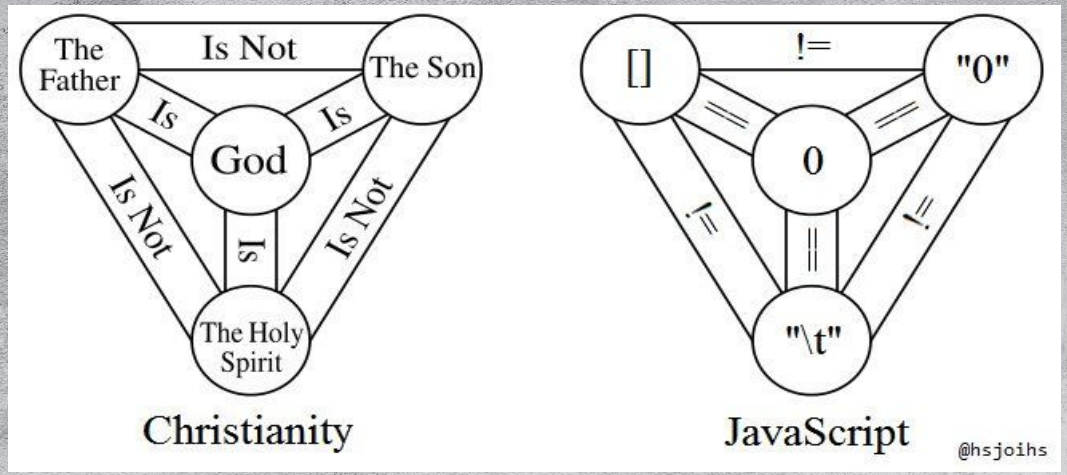
Java™



Sorry for this slide, had too much of coke

Choosing language

thank u, next



Choosing language

JS Object Model is just great.

- Strings
  - "WAT" + 1 >>> "WAT1"
  - "WAT" - 1 >>> NaN
- Dict
  - [] + {} >>> [object Object]
  - {} + [] >>> 0
- [] == [] >>> false (Obviously.)

Choosing language

The carbon-dated option

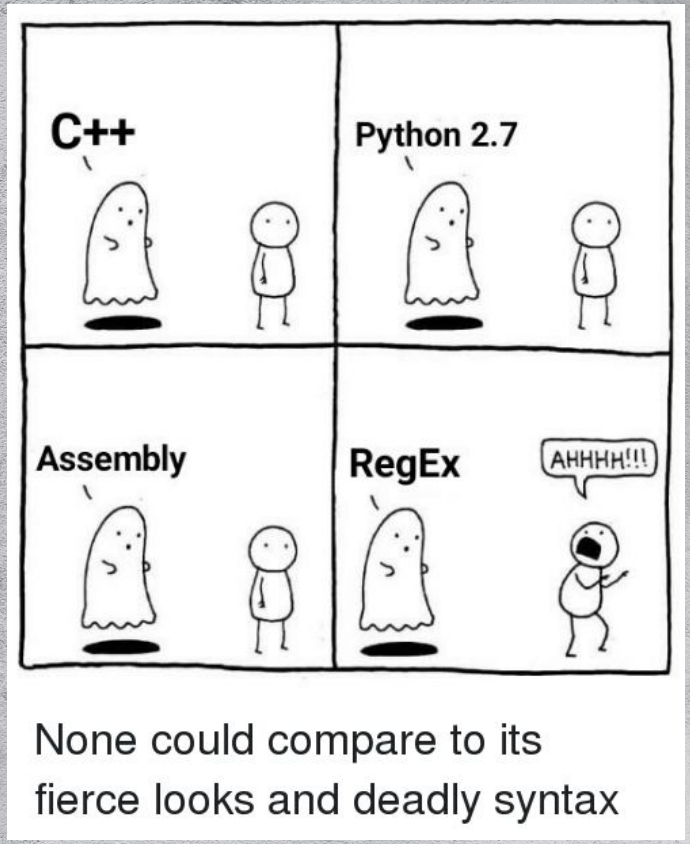
Let’s get Pythonic

Regex - Regex

Found Python module named rstr

It has a xeger() method which allows creating a random string from a regular expression.

https://bitbucket.org/leapfrogdevelopment/rstr



Esoteric rstr bug

Unfortunately, it has a bug, which causes what I call “length explosion”, basically the length of string generated from various RegEx(s) tend to go up and up. I fixed i, and then found Xeger (https://pypi.org/project/xeger/)

```
>>> from xeger import Xeger
>>> x = Xeger(limit=10)  # default limit = 10
>>> x.xeger("/json/([0-9]+)")
u'/json/15062213'
```

bug in action

```
>>> len('Warp/583054472440781110899315707516776271021861074.64582092977883255483304725522407252733055221443436300011607433258814501737175324469904685918057470.964498415559528906857015312705.758272282446344519159
257526178834503307545592062980624528864171897496648369.857837069252664281325271817606018455672399585594333291.931932785486919045080954542162141086675589276340096684069822732499234738638678491631265754191767.227
615795335121691035203919525165856938864487198062195373553414739707729.321298540729811547120583715048124392853966559781382245285613348743419222139 52.57594466663584206799414927517761979778742978525982941459 3.0023
373461573885331803677624010506643982569964759219735449989517085453 6.01243739133.80147537064660674716307 4.743140635274695707888646296238633542911282407822678688484416863202 8.304841569164093355383617943004802553
669073621989355021223109892065695283523096771 2.476757873013691112378100934120114164498842448668690655655166220063775785.245765721402513454917598040695799908694580503358870163607056407511476159370.054254093600825
533861940194588003910608591048.695458998355819735929506301660987637315192499421 53.90643906141065400834678715928672758052099836276022537168903222885 76.224276.160526686599737220879250855950409960150888156059685976
695316238640872268957877449451517.75591018404722770943634284999245058578023038 12.673077758475659391272645201024355902174513113753879658063514375723290999963.102692836935688349569799040345660199050079344464194104492503267066474850 31.495478878678143496486621592522823236949941668 9.112656837494314199129528024788872367631276583636343588611368154693490109097.499059276360574305178624356596815281413067219739076507858537507359144842887963236 6.923866059626816384792983139302510966977927709396632112061380079 60.872284399789664737805927992.353204206806773803495545581.038598035623060629234995765173724084437002274912698269523692643565895827
754390711769743314 9.419849415313002726530950498427187036879124105998681182961669621573729091965946770 22.454100569879631416112618516338999379518840004142 5.255144822 1.77122884 8798799893461882720821421240341999563
937459439313486817385566067315785817667526759536764303471 5.05864 7.795067565054092401458063062700124180 2870829765779302463643034355209722838638785821 40.941222664292255038159278210678743374417 44.303125557099847614642
060968547003541444327081602621927233365124539936127328 86.913351639072662873379968858072935959489271039421349265578 5.375108343188569415457914657061033805672892087922119 3 53.804737043850238182166839533613590902
532202948680991626834981 2.235092827149934678201992716851253341115976969156535209711147907016 26.601356415840284230497634235743783758321827692570147648 37.752192365089740474 42.551890017721232910702219431668593059
530449446101792285333107666557637723652387805 1.140777624708164041772415919837105202517999235688138202071743790561584330333875139370959629923 0.75882 1.187641240836005545446632992052 5.544304199435376430582866220950422
550073812958008743962960649943749608604633982295414849546514251835 4.231463148768168257892875204257692476970117272425191297703747288735878712 6.154087123310719931050456114575175659 6.518055763969683815240136551870470
706587301900943899625259 6.604399081328729368806721 53.314492933549080000618274181279465812841450890400418302687063074107 0.043698703423507478314964699585525507135939305246 3.930842166293013266465800439487360228168960
392089957764966260870000191810106433264709 2.263069497817285089773310790403735543176945224667003022941324 6.76498026874 4022937870895119071650166424144352892955576517397759750823946883611873170573 2.255368210502953340
916444939222933536326 1.113153371 7.748139916507291804899457 5.120502413041589701654790098870252170098972724773916883393778539429 5.999243641942573603140643368501437754761954663712303934927455599736674 9.766846037810
293707767056553681554480586167851490738857248222252526467235389912211156662418696463550 8.877328036242764553342255106729966089677117853490989491449376153905542813711 4.472554722082495727 8.390263677428262758660051060
141349527204717209743826 4.819545262920813881619914945118801 4.089503716225711406782794269793755860921662375 5.940391561713349485191394476472994569754682994910189991125567140105000499544845778557068665 35.667839729507
474655553306396047986562173853453247985056722317503385502185503409293874375415207001 0.529149000156196 7.805235418336665539067106125065018714047266773033872453878731245 3.357121736810894788071045 8.855042764790811779919
928047673512771993513384229854 5.696208814558301069373328978493385717157355570495885976813353237709884637670990082 5.589009746108823989826805432638942 4.096179564100497217606481375259415680843984322101193034940331
736416840 4.45727741314 2.268357944358702185041174736319752900192121326927550373318446980355663108899 0.065772107329226875185 1.440067414021868814799174786018287772497413820092407518274692048347737877655212958 36.27612
795560834771851930557097542021299174817697211174194132 6.677894852330498826106441077651399305765968254302052158310 0.060788032311679741283619 9;version:583054472440781110899315707516776271021861074.645820929778832555
483304725522407252733055224143436300011607433258814501737153246899046859180574700.964498415559528906857015312705.758272282446344519159257526178834503075455592062980624528864171897496648369.857837069252664281325
271817606018455672399585594333291.931932785486919045080954542162141086675589276340096684069822732249923473843867849163126575419176 7.227615795335121691035203919525165856938864487198062195373553414739707729.321298
540729811547120583715048124392853966559781382245285613348743419222139 52.575944666635842067994149275177619779778429785259829414593.00233734615738853318036776240105066439825699647592197354499895170854536.012437391
33.801475370646606747163074.743140635274695707888646296238633542911282407822678688484416863202 8.304841569164093355383617943004802553366907362198935502122310989206569528352309677 12.476757873013691112378100934120
114164498842448668690655655166220063775785.245765721402513454917598040695799908694580503358870163607075640751147615937 0.054254093600825338619401945880039106085910 48.695458998355819735929506301660987637315192499421
2153.906439061410654060834678715928672758052099836276022537168903222885 76.224276.160526686599737220879250855950409960150888156059685976953162386408722689578774494515 17.755910184047227709436342849992450585780230812.
812.673077758475659391272645201024355902174513113753879658063514375723290999963.102692836935688349569799040345660199050079344464194104492503267066474850 31.495478878678143496486621592522823236949941668 9.112656837
749431419912952802478887236763123765836364588611368154693490109097.499059276360574305178624356596815281413067219739076507858537507359144842887963236 6.923866059626816384792983139302510966977927709396632112061380
07960 8.722284399789664737805927992.353204206806773803495545581.038598035623060629234995765173724084437002274912698269523692643565895827754390711769743314 9.419849415313002726530950498427187036879124105998681182961
669621573729091965694677022 4.454100569879631416112618516338999379518840004142 2.255144822 1.77122884 8798799893461882720821421240341999563937459439313486817385566067315785817667526759536764303471 5.05864 7.795067565054092
401458063062700124180 2870829765779302463643034355209722838638785821 40.941222664292255038159278210678743374417 44.303125557099847614642060968547003541444327081602621927233365124539936127328886 9133516390726628733
996885807293595948927103942134926557858 18.375108343188569415457914657061033805672892087922119 3 53.804737043850238182166839533613590902532202948680991626834981 2.235092827149934678201992716851253341115976969156535
2097111479070162 6.601356415840284230497634235743783758321827769257014764837 7.752192365089740474 42.551890017721232910702219431668593053953044944610179228533310766655763772336523878 05.140777624708164041772415919837
105202517999235688138202071743790561584330333875139370958629923 0.75882 1.187641240836005545446632992052 5.544304199435376430582866220950425500738129580087439629606949943749608604633982295414849546514258 3.2314631
876816825789285720425769247697011727242519129770374728873587812 5.154087123310719931050456114575175659 6.518055763969683815240136551870470658730190094389962529 6.604399081328729368806721 53.314492933549080000618274181
812794658128414508904004183026870630741 07.043698703423507478314964699585525507135939305246 3.930842166293013266465800439487360228168960392089957764966260870000191810106433264709 2.263069497817285089773310790403735543
317694522466700302294132 6.76498026874 4022937870895119071650166424144352892955576517397759750823946883611877317057 3.255368210502953349164449392229335363216 1.113153371 7.748139916507291804899457 5.120502413041589701
654790098870252170098972724773916883393778539429 5.999243641942573603140643368501437754761954663712303934927455599736674 9.766846037810293707767056553681554480586167851490738857248222252526467235389912211156662
1869463550 8.877328036242764553342255106729966089677117853490989491449376153905542813711 4.472554722082495727 8.390263677428262758660051061413495272047172097438264 8.819545262920813881619914945118801 0.089503716225711406
678279426793755860921662375 5.940391561713349485191394476472994569754682994910189991125567140105000499544845778557068665 35.667839729507474655553306396047986562173853453247985056722317503385502185503409293874375415207001 0.529149000156196
7.805235418336665539067106125065018714047266773033872453878731245 3.357121736810894788071045 8.855042764790811779919928047673512771993513384229854 5.696208814558301069373328978493385717157355570495885976813353237709884637670990082 5.589009746108823989826805432638942 4.096179564100497217606481375259415680843984322101193034940331736416840 4.45727741314 2.268357944358702185041174736319752900192121326927550373318446980355663108899 0.065772107329226875185 1.440067414021868814799174786018287772497413820092407518274692048347737877655212958 36.27612795560834771851930557097542021299174817697211174194132 6.677894852330498826106441077651399305765968254302052158310 0.060788032311679741283619'
9970
>>>
```

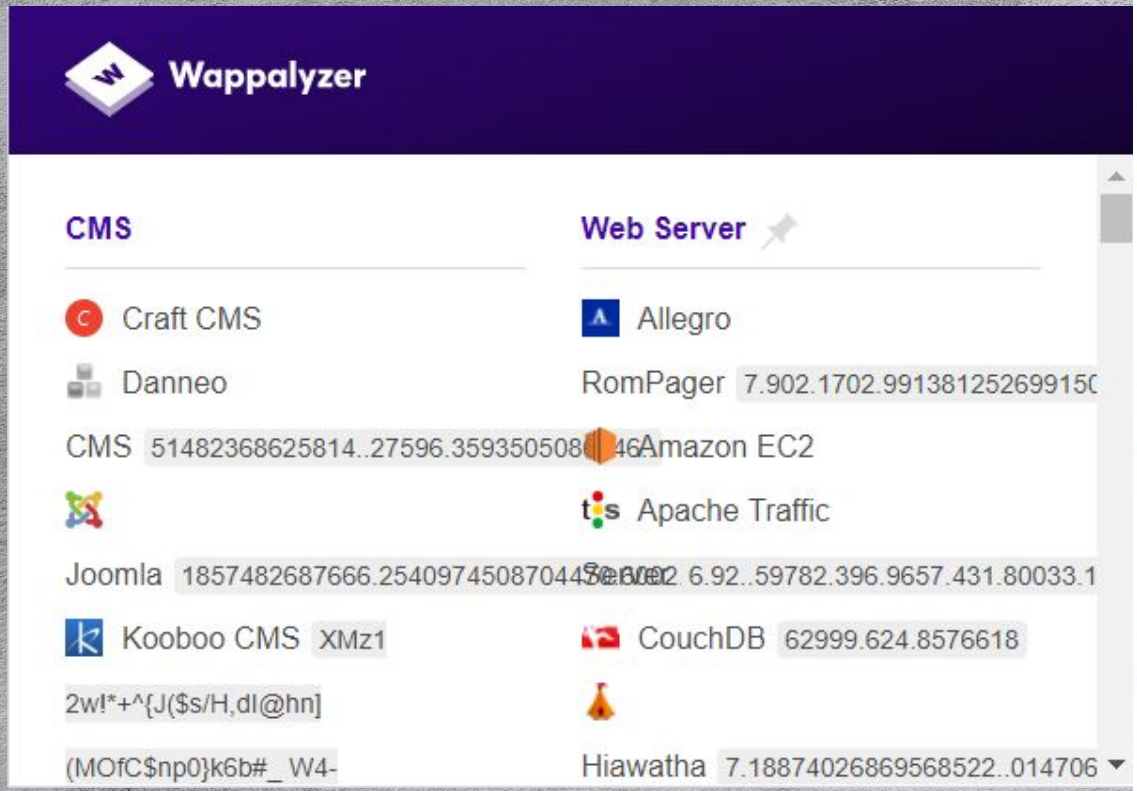
Initial testing

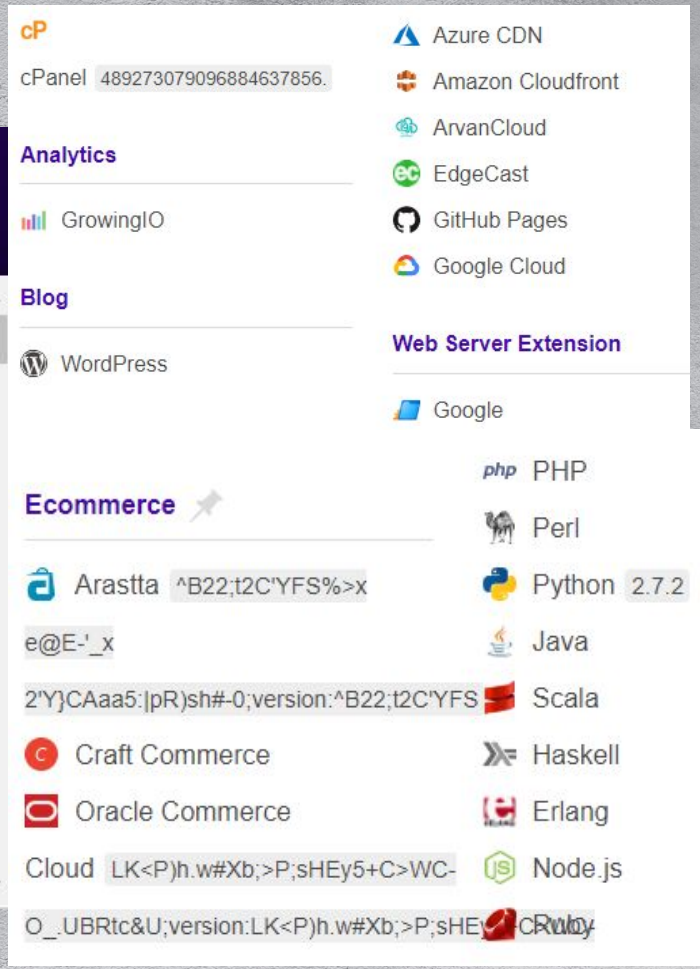
Generated tons of header-patterns and spewed those crafted server headers all over a bare-bone script on top of Python SimpleHTTPServer

```python
#!/usr/bin/env python
import SimpleHTTPServer

class MyHTTPRequestHandler(SimpleHTTPServer.SimpleHTTPRequestHandler):
    def end_headers(self):
        self.send_my_headers()

        SimpleHTTPServer.SimpleHTTPRequestHandler.end_headers(self)

    def send_my_headers(self):
        self.send_header("Server", "mod_wsgi/1;version:1")
        self.send_header("X-AH-Environment", "MceYyD39TQQHbGNrabJMRFElW9bomZzj4qvbUCZRiSRnv")
        self.send_header("X-Advertising-By", "adnegah.net")
        self.send_header("X-Powered-By", "E2 Aegea v234442575619739968710879319893811163575146427055435l6;version:234442575619739968710879
        self.send_header("Server", "Allegro-Software-RomPager/7.902.1702.99138125269915040S.773;version:7.902.1702.99138125269915040S.773"
        self.send_header("Via", "(CloudFront)")
        self.send_header("Server", "(Amazon)")
        self.send_header("Server", "ATS/6.92..59782.396.9657.431.80033.13727515;version:6.92..59782.396.9657.431.80033.13727515")
        self.send_header("X-Powered-By", "Salesforce.com ApexPages")
        self.send_header("Arastta", "^B22;t2C'YFS%>x e@E-'_x 2'Y}CAaa5:|pR)sh#-0;version:^B22;t2C'YFS")
        self.send_header("AR-PoweredBy", "Arvan Cloud (arvancloud.com)")
        self.send_header("X-Powered-By", "Chamilo 5945.76704646.936942.937459498.23831871328;version:5945.76704646.936942.937459498.238318
        self.send_header("Server", "CouchDB/62999.624.8576618;version:62999.624.8576618")
        self.send_header("X-Powered-By", "CppCMS/783789261.37826828970;version:783789261.37826828970")
        self.send_header("X-Powered-By", "Craft CMS")
        self.send_header("X-Powered-By", "Craft Commerce")
```

Results ?

Wappalyzer going crazy





Get ready for a major remodel fellas

Started coding a command-line application which,

1. pull the latest apps.json from the official repository
2. parses all objects from apps sub-node
3. processes list of RegExs making them payloads

   which then can be utilised via either embedding them into the front-end (website) or in back-end (server)

Introducing Wapparalyser

We went through a list of names though:

1. WAPDEAD
2. Wappalyzer-seizure
3. Weizure (transcended from No. 2)
4. Confulyser (WAP-Confuse)
5. WAP-DESTROY-INATOR (Phineas and Ferb)

Features of Wapparalyser

- ➔ Support for
  - ◆ Emulating services
    - Random
    - All
    - Certain tech-stack (e.g. MEAN, LAMP)
- ➔ In-built small fuzzer for Wappalyzer
    - Blind
    - metadata|js|scripts|html|headers|cookies

THE DEMO

Impact of Wapparalyser

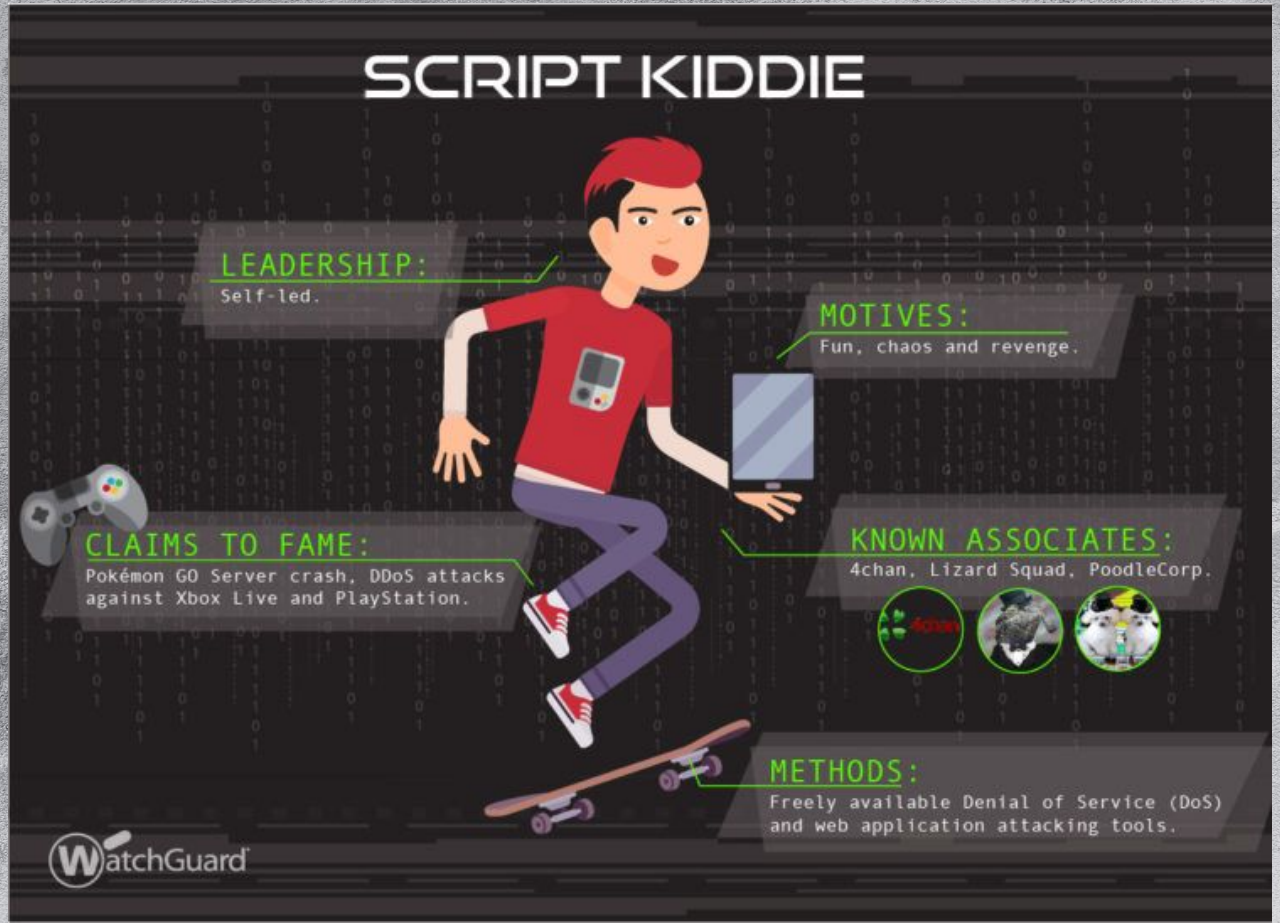
1. Breaks a company with $10M in estimated revenue annually, making their tech obsolete.
2. Shows current posture of a famous recon tool.
3. Describes the brittle nature of current methodologies.
4. Most probably, just another security tool which will get dusty over time.

Open Source == <3

Wapparalyser will be released under MIT license and can be found after this talk on my GitHub, 0x48piraj/wapparalyser

I'll probably whine about the bugs/features on Twitter, so, you will get to know, easily.

Messing with script kiddies



Messing with script kiddies

We all have coded exploits which go something like,

```
response = requests.get('http://target.com')

if response.status_code == 200:
    print('Success!\nRunning our 1337 exploit...')
    pwn(response)
elif response.status_code == 404:
    print('Target Not Found. Quitting.')
```

Messing with script kiddies

The typical what I called “script-kiddie cycle”

- We release our exploits.
- Script kiddie goes to repository, blog, the source!
- Does CTRL+C then CTRL+V into his Notepad.
- Saves the file, tries to run the script ...
- Cries in the corner if we intentionally broke some parts in our exploit.

Messing with script kiddies

1. What if we can break those exploits (?)
2. What if we can break security tools used for reconnaissance

Disclaimer: I don't support "Security by Obscurity" in any sense but if you can increase time and money of an attacker and shoo away script kiddies without much effort, then I don't see why not !?!

Their most favourite tool

RAPID7 metasploit®


```
xMMMMMMMMMMd                                  ,0MMMMMMMMMMMMK;
.WMMMMMMMMMc                                    'OMMMMMM0,
 lMMMMMMMMMMk.                                    .kMM0'
  dMMMMMMMMMMWd'                                    ..
   cWMMMMMMMMMMMNxc'.                            ##########
    .0MMMMMMMMMMMMMMMMWc                        #+#    #+#
      ;0MMMMMMMMMMMMMMMo.                      +:+
        .dNMMMMMMMMMMMMo                      +#++:++#+
           'oOWMMMMMMMMo                            +:+
               .,cdk00K;                  :+:     :+:

                                          :::::::+:
          Metasploit Under Construction

       =[ metasploit v5.0.0-dev-f355d10771                ]
+ -- --=[ 1726 exploits - 987 auxiliary - 300 post        ]
+ -- --=[ 507 payloads - 40 encoders - 10 nops            ]
+ -- --=[ ** This is Metasploit 5 development branch **   ]

msf5 exploit(linux/misc/drb_remote_codeexec) >
```

Messing with script kiddies

Do you know how often does Metasploit reference status codes?

```
grep -r -E 'res[p|ponse]?\.code' * | wc
```

Grep search for,

- res
- resp
- response

NOTE: The method is not reliable as it depends on how we name things.

Messing with script kiddies

For keeping this research “latest”, I cloned msf master repository today. (I’ve to admit, having a cloud server is handy)

```
dev@ubuntu:~/metasploit-framework$ grep -r -E 'res[p|ponse]?\.code' * | wc
   2416   20885  273936
modules/exploits/windows/http/coldfusion_fckeditor.rb:    if ( res and res.code == 200 and res.body =~ /OnUploadCompleted/ )
modules/exploits/windows/http/hp_pcm_snac_update_domain.rb:    if res and res.code == 200 and res.body =~ /domainName/
modules/exploits/windows/http/hp_pcm_snac_update_domain.rb:    if res and res.code == 200
modules/exploits/windows/http/hp_pcm_snac_update_domain.rb:    if res and res.code == 200 and res.body =~ /success:false/
modules/exploits/windows/http/php_apache_request_headers_bof.rb:    if res and res.code == 500
modules/exploits/windows/http/cogent_datahub_command.rb:    if res && res.code == 200 && res.body =~ /PermissionRecord/
modules/exploits/windows/http/zenworks_assetmgmt_uploadservlet.rb:    if (res.code == 500)
modules/exploits/windows/http/avaya_ccr_imageupload_exec.rb:    if res and res.code == 200 and res.body =~ /"Key":"RadUAG_success","Value":true/
modules/exploits/windows/http/avaya_ccr_imageupload_exec.rb:    if (!res or (res and res.code != 200))
modules/exploits/windows/http/serviio_checkstreamurl_cmd_exec.rb:    if res.code != 200 || res.body !~ %r{<errorCode>603</errorCode>}
modules/post/multi/gather/check_malware.rb:    case res.code
spec/lib/metasploit/framework/login_scanner/caidao_spec.rb:          res.code = 200
spec/lib/msf/core/exploit/http/wordpress/base_spec.rb:        res.code = wp_code
spec/lib/msf/core/exploit/http/wordpress/login_spec.rb:        res.code = 301
spec/lib/msf/core/exploit/http/wordpress/version_spec.rb:          res.code = 200
spec/lib/msf/core/exploit/http/wordpress/version_spec.rb:          res.code = wp_code
spec/lib/msf/core/exploit/http/wordpress/version_spec.rb:          res.code = wp_code
spec/lib/msf/core/exploit/http/wordpress/version_spec.rb:          res.code = wp_code
spec/lib/msf/core/exploit/http/jboss/base_spec.rb:          res.code = res_code
spec/lib/msf/core/exploit/http/jboss/base_spec.rb:          res.code = res_code
spec/lib/msf/core/exploit/http/jboss/deployment_file_repository_spec.rb:          res.code = res_code
spec/lib/msf/core/exploit/http/jboss/bean_shell_spec.rb:          res.code = res_code
spec/lib/msf/core/exploit/http/joomla/base_spec.rb:    res.code = 200
spec/lib/msf/core/exploit/http/joomla/version_spec.rb:    res.code = 200
spec/tools/md5_lookup_spec.rb:    res.code = 200
tools/password/md5_lookup.rb:      return result unless res && res.code == 200
tools/exploit/virustotal.rb:    case res.code
tools/modules/module_reference.rb: _if res.nil? || res.code == 404 || res.body =~ /<title>.*not found<\/title>/i
```

Messing with script kiddies

How can we break all the exploits ?

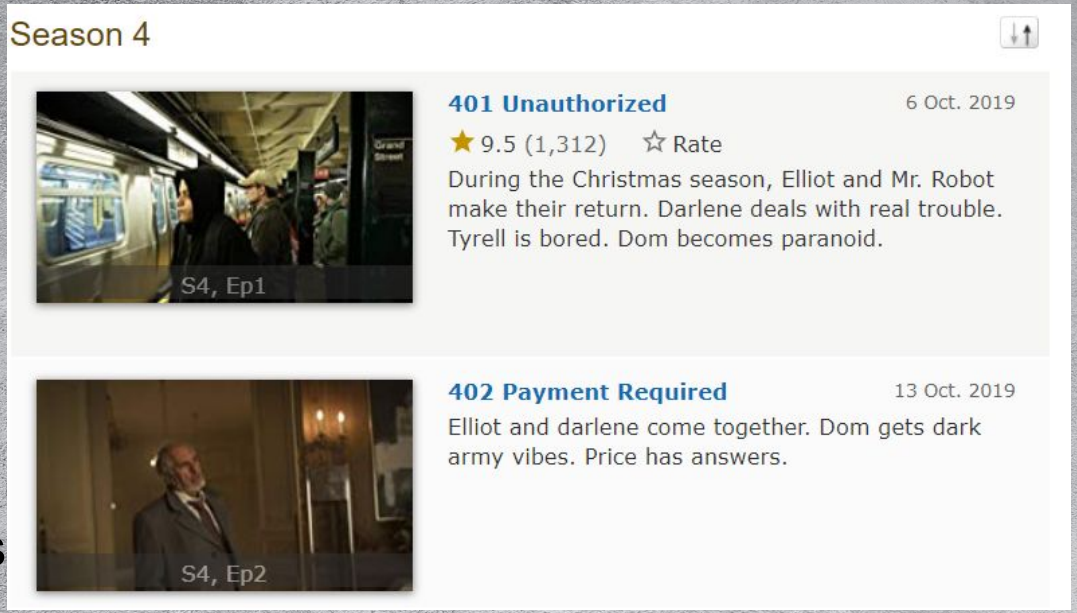
```
response = requests.get('http://target.com')

if response.status_code == 200:
    print('Success!\nRunning our 1337 exploit...')
    pwn(response)
elif response.status_code == 404:
    print('Target Not Found. Quitting.')
```

Let's suppose this is an exploit which our server is vulnerable of, if we send a 404 response code, this very exploit will **FAIL**

Messing with script kiddies

Same goes for,

- Scanners
- Web crawlers
- Web-App security suites



If we're able to change the res

codes, we will create **HAVOC** (flood

of false-positives)

Messing with script kiddies

This can be easily done by using HTTP Handler (reverse proxy) that takes an incoming request and sends it to another server, proxying the response back to the client.

Returns any response, with any HTTP status code.

What can be done?

Current methods for fingerprinting are very susceptible to attacks we discussed.

The most cheap reliable option is to do multiple tests with statistical analysis, combined with fuzzy logic techniques as we always have to make a security trade-off.

SECURITY vs RESOURCES/COST

What can be done?

New methods are also being developed, some utilizing machine learning.

- Deep Fingerprinting: Undermining Website Fingerprinting Defenses with Deep Learning (Attacking TOR with DL)
- Automated Website Fingerprinting through Deep Learning
- Deep Web Server Fingerprinting via analysing code-base & code-use (will publish the pre-print soon)

Gist of this research

- Too much RegEx matching is bad
- Heavy reliance on status code leads to the dark side
- New methods for fingerprinting are sorta cool
- And, yeah, trusting MAC addresses for identification is a bad idea …

Thanks

Piyush Raj | @0x48piraj
https://blog.0x48piraj.com