Hacking (and Securing) Web Applications

Kevin Bluer

Session Agenda

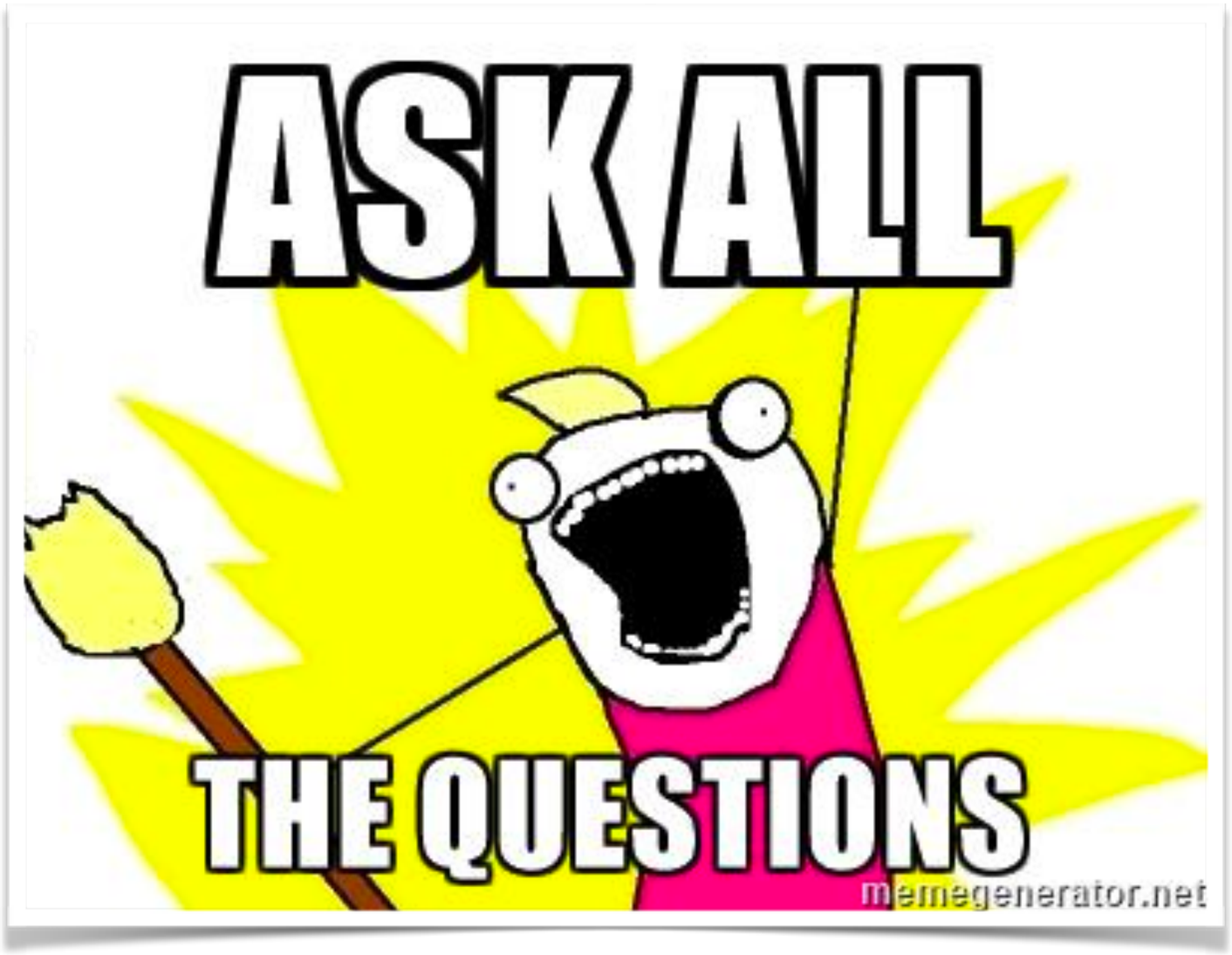
Session Agenda

- Session Goals
- Trends & Stats
- Background & Basics
- Tools of the Trade (aka Kali Linux)
- Phases x5
- Exploration of Common Attacks
- Future Practice
- Summary & Q&A

About Me

- CTO of Nest.vc
- 10+ years in web / mobile application development
- Active instructor (General Assembly, etc)
- Still “pen test” n00b

Interactivity FTW!



JS

Session Goals

Session Goals

- Come away a little bit scared (although hopefully not terrified) and inspired
- Appreciation of commonly used tools / techniques
- Empowered to know where to begin in terms of securing your (or other people's) applications

Session Scope / Approach

- It's a HUGE topic (web app, physical, network, mobile, social engineering, etc)
- Provide broad overview of the process specifically
- Dive specifically into (JS centric) web applications
- We'll also go through it in the classic "infosec" way
  - Explore the various stages
  - Look at some tools for each of them

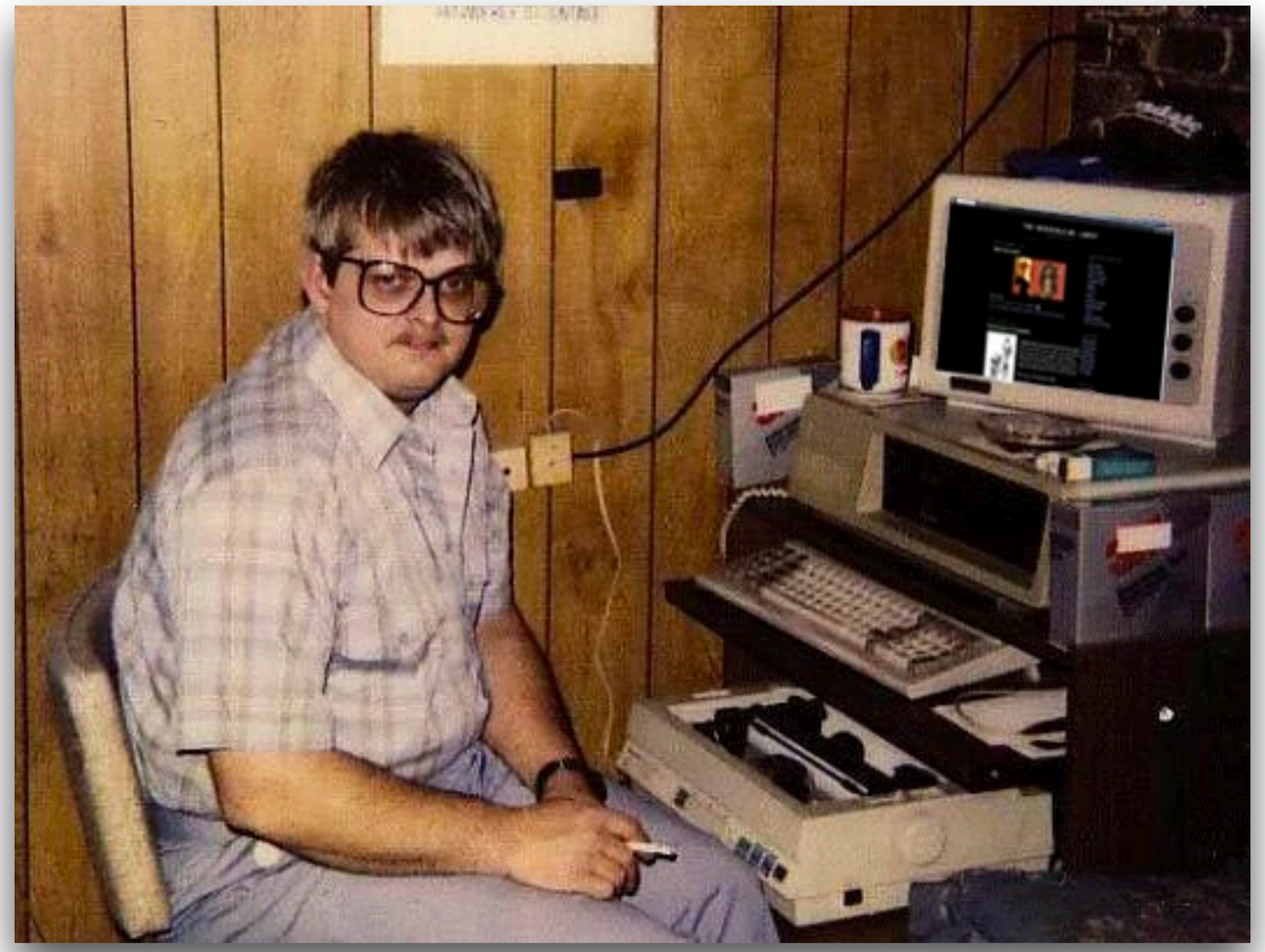
Show of Hands

- Kali Linux?
- OWASP?
- W3AF?
- DVWP?

Trends & Stats

Myth vs Reality

Myth vs Reality



Trends & Stats

- We're capturing more “**stuff**” than ever (4.4ZB in 2013 and an 44ZB by 2020) via more “**things**”
- Increasingly (rather obviously) this is made available via the web, **willingly or otherwise** :)
- Study by Verizon highlighted that 96% of hacks were **not "highly difficult"** (meaning misconfiguration, commonly known exploits, etc)
- Increasingly utilization cloud-based apps and infrastructure provide additional “attack vectors”

Background & Basics

Terminology

- “Penetration testing (also called **pen testing**) is the practice of testing a computer system, network or Web application to find vulnerabilities that an attacker could exploit.”
- “An **attack vector** is a path or means by which a hacker (or cracker) can gain access to a computer or network server in order to deliver a payload or malicious outcome.”
- **White Hat** vs **Black Hat** (vs Grey Hat)

Motivations of Hackers

- Monetary
- Political / Warfare
- Grudge
- Vandalism
- Fun or Curiosity (ethical hackers)

Types of Attacks

- Denial of Service
- **Theft** (database entries, source code, IP, etc)
- Infrastructural Damage
- Ransomware (grew 752% last year)

How?

- 81% Form of Hacking
- 69% Malware
- 10% Physical Attacks
- 7% Social Tactics
- 5% Privilege Misuse

Attack Vectors / Considerations

- A way to begin thinking about the services you / your target implements (on-premise and 3rd party)
- Examples
  - Web server
  - Database (known or unknown exploits)
  - Misconfiguration
  - Your application code
  - Your users (data, etc)

Rules of Engagement

While learning this stuff…

- Be careful + aware!
- Many of the tools and techniques could get you in serious trouble if used inappropriately
- Start with practice environments (or own systems)
- **Do not go begin testing (hacking) sites without prior permission!**

JS

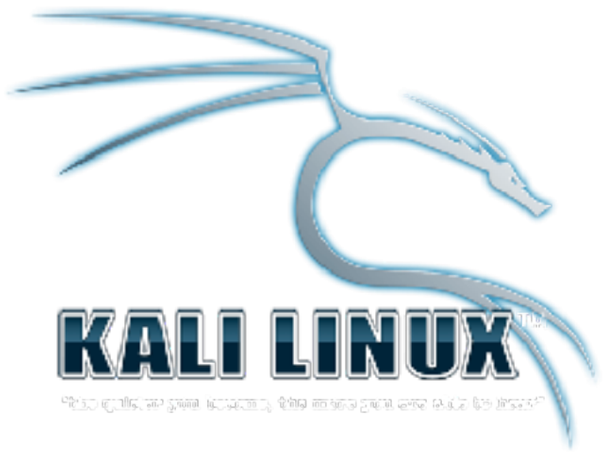
Just because you can…

- …doesn’t mean you should

Tool(s) of the Trade

Kali Linux

- https://www.kali.org/
- Debian-based Linux distribution
- Developed by Mati Aharoni and Devon Kearns of Offensive Security
- Completely loaded!

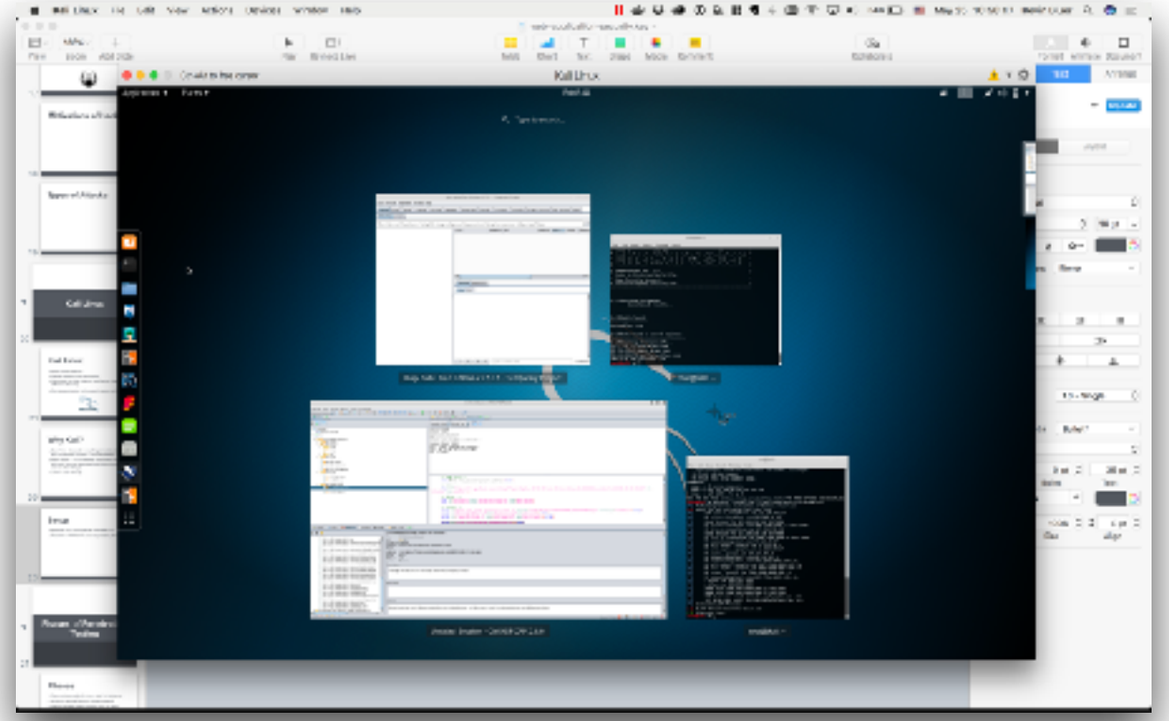


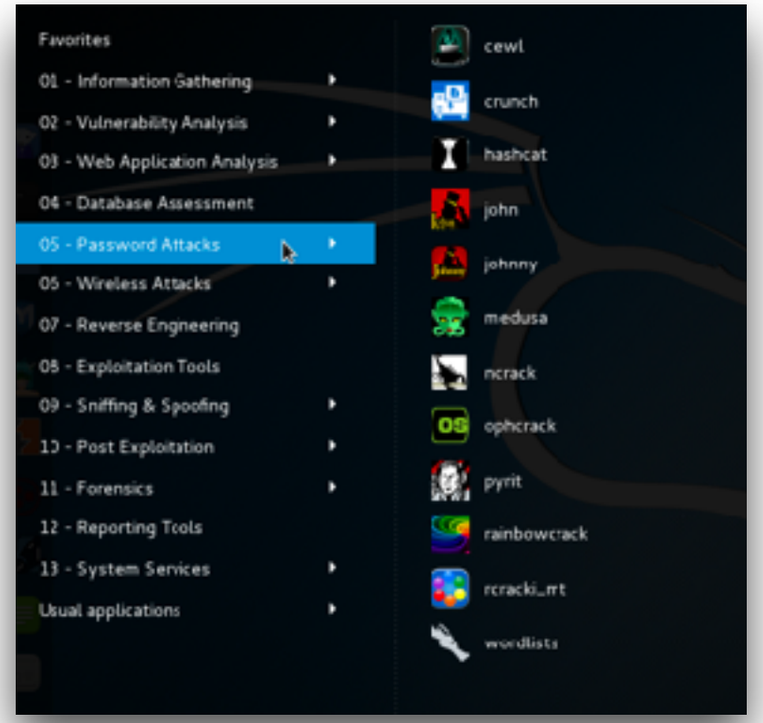
Why Kali?

- Time (saved) - Literally everything is installed, setup, configured, maintained, updated…
- Hassle (saved) - It’s throwaway and doesn't interfere with your primary desktop (which you invariably use for other things)
- Learning (increased) - By exploring all the tools and services that it ships with…

Setup

- Although you could use as your base OS…
- Parallels or VMWare is probably the way to go





Demo: Kali Linux

Phases of Pen Testing

Phases

1. **Reconnaissance** (preliminary data / intelligence)
2. **Scanning** (actual insights on the systems)
3. **Gaining Access** (taking control and / or data)
4. Maintaining Access (ensure continued access)
5. Covering Tracks (remove all tracks)

Note that there are a few variations on this (depending on your objectives, goals, etc)

Phases

1. Reconnaissance

Reconnaissance

“Reconnaissance is the act of gathering **preliminary data or intelligence** on your target. The data is gathered in order to better plan for your attack. Reconnaissance can be performed actively (meaning that you are directly touching the target) or passively (meaning that your recon is being performed through an intermediary).”, cybrary.it

Phase Goals

- Key staff and organizational structure
- Office locations
- Use of social media
- Sophistication of technology operations
- 3rd party systems, services, integrations, etc
- Preliminary understanding of technology stack, infrastructure, domains / subdomains, etc

JS

Job Boards

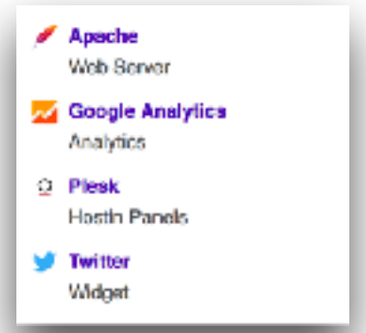
Job Boards, etc

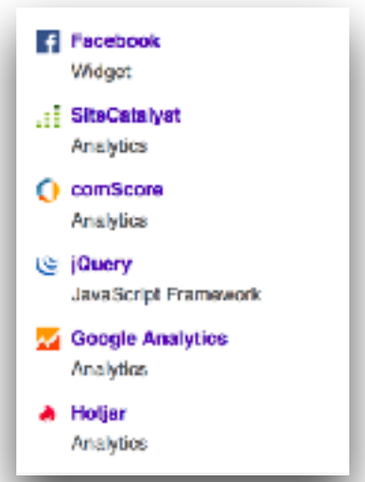
- Jobs boards often very clearly advertise the technologies, personal
  - JobsDB, Monster, Indeed, etc
  - LinkedIn - look at the company entry and types of skills / stacks / etc that employees list
  - Twitter / Google - site:twitter.com "job listing”

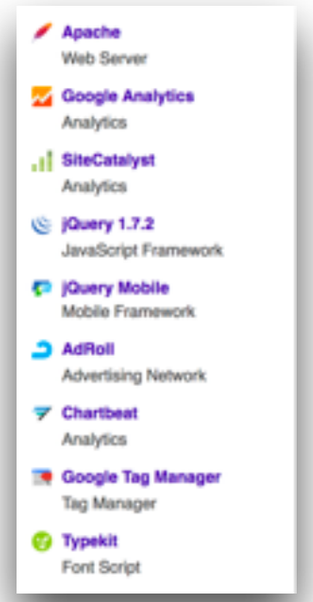
Wappalyzer

Wappalyzer

- Wappalyzer is a cross-platform utility that uncovers the technologies used on websites. It detects content management systems, ecommerce platforms, web frameworks, server software, analytics tools and many more.”







Demo: Wappalyzer

Wappalyzer

- Analyses the header information that is received / source code / etc
- Examples
  - https://wordpress.org (“WordPress”)
  - https://www.microsoft.com (“IIS”, “ASP.NET")
  - https://www.hsbc.com.hk/ (client-side only)

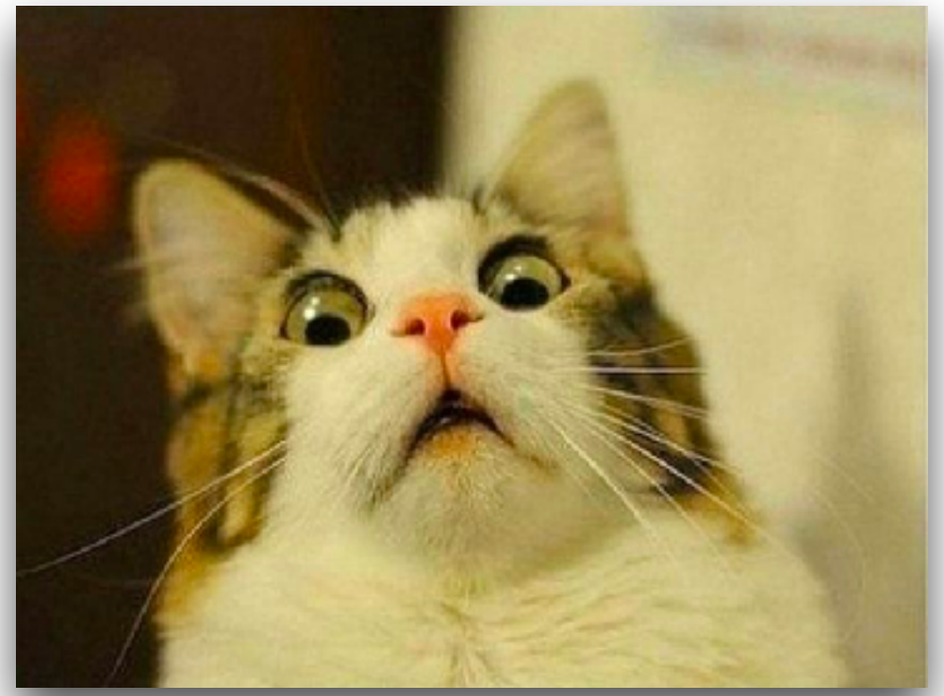
Google-fu

Google-fu

- Leverage more sophisticated Google searches: filetype, inurl, intitle, site, etc
- https://www.google.com.hk/search?safe=active&q=site%3Awestminstertravel.com+portal
- https://www.google.com.hk/#safe=active&q=site:jobsdb.com+filetype:pdf
- https://www.google.com.hk/search?q=site:hsbc.com.hk+intitle:admin

Google-fu

- From the above…
  - http://b2b.adholidays.com
  - http://ad.hkwtl.com/retail/login.asp



JS

Google Hacking DB

- Weaknesses / Exploits exposed directly via Google
- https://www.exploit-db.com/google-hacking-database
- https://www.exploit-db.com/google-hacking-database/?action=search&ghdb_search_cat_id=0&ghdb_search_text=wordpress

JS

The Harvester

The Harvester

- Automated tool for assessing the presence on social media
- Help mitigate social engineering attacks
- See screenshot for examples of harvesting everything related to your companies domains (e.g. Nest.vc / @nestideas for Twitter)
- > theharvester -d jobsdb.com -b google -l 10
- > theharvester -d nest.vc -b linkedin -l 10
- > theharvester -d westminstertravel.com -b …

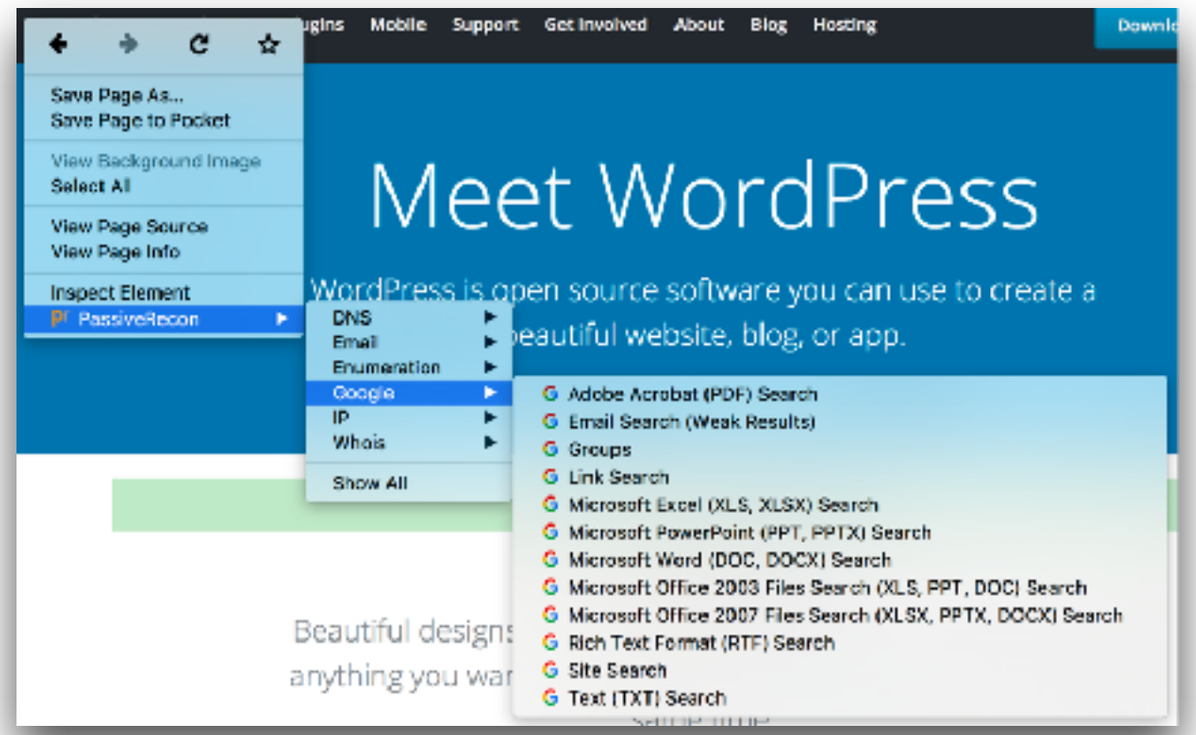
Demo: The Harvester

The Harvester

Passive Recon

Passive Recon

- Firefox Plugin (nice example of another type of tool)
- Nothing couldn't do from the command line or other tools but really handy to have all in one place



Demo: Passive Recon

Demo: Passive Recon

- IntoDNS: provides a ton of useful information for any given domain
- Netcraft: Site technology, Email security, Hosting history, Frameworks, Etc

Recon-ng

Recon-ng

- Recon-ng is a full-featured Web Reconnaissance framework
- Independent modules, database interaction, built in convenience functions, interactive help, and command completion
- Examples
- > show modules
- > use recon/domains-hosts/netcraft
- > set SOURCE oreilly.com (or metta.co)
- > run

2. Scanning

Scanning

“The phase of scanning requires the application of technical tools to gather further intelligence on your target, but in this case, the intel being sought is more commonly about the systems that they have in place.”, cybrary.it

Phase Goals

- Some overlap with “reconnaissance”…
- To go to the next level and begin understanding the actual infrastructure / platforms. E.g.
  - OSes, Web Servers, CMSs, CRMs, etc
  - Bespoke platforms (use of frameworks)
  - Major / minor versions (and patches)
  - SSL / TLS setup
  - **IP addresses, DNS, Open ports, WAFs, etc**
- Begin cross-referencing against vulnerability databases…

DNS Recon

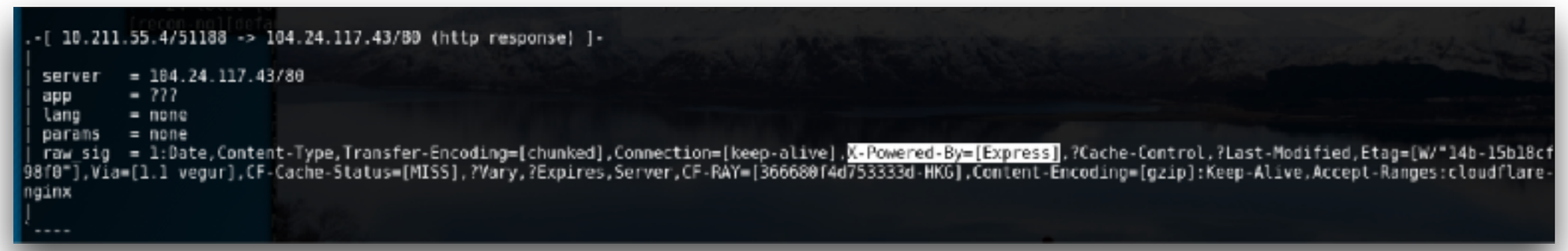
DNS Recon

- Domain Name System (DNS) provides a way to match names (e.g. google.com) to numbers (the IP address for the website)
- This would potentially find subdomains that haven't been indexed by Google, Netcraft, etc
  - > dnsrecon -d google.com -D /usr/share/wordlists/dnsmap.txt

p0f

p0f

- “PASSIVE FINGERPRINTING”
- Note that by virtue of being passive it will just manifest itself as "normal" traffic
- Examines traffic across the various OSI layers (and cross references a fingerprint database) to provide OS, Web Server, etc



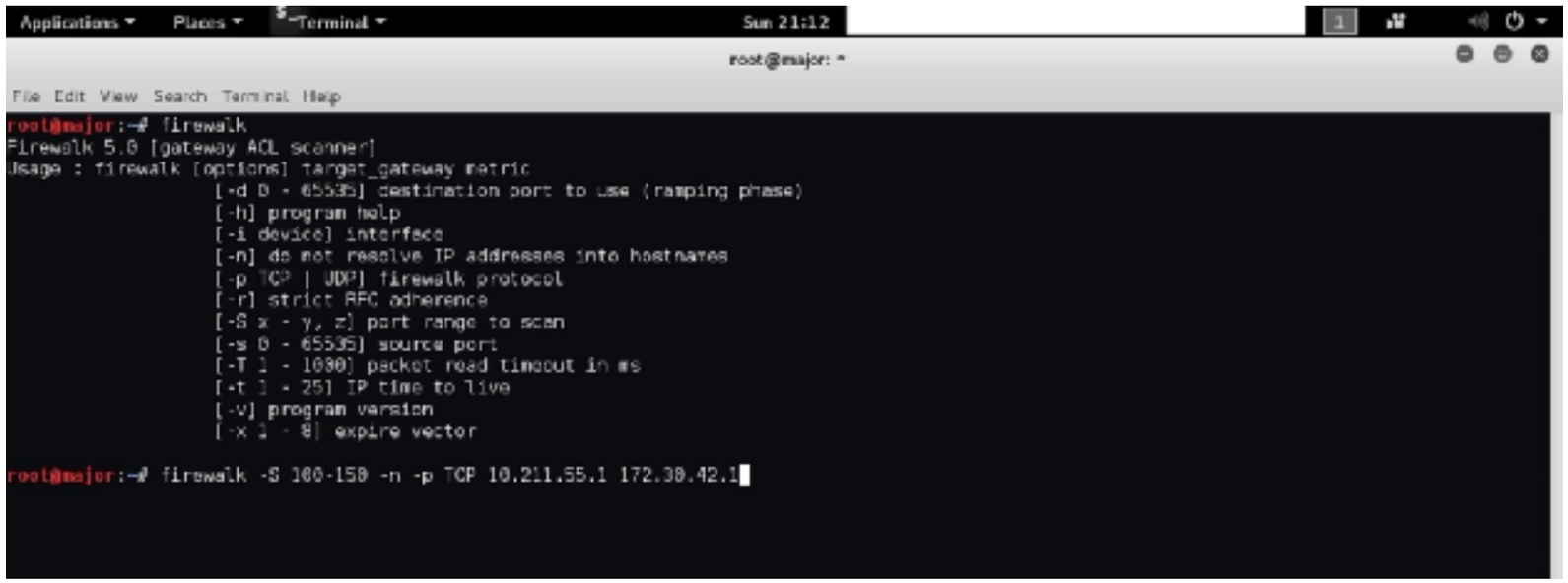
Additional “Scanning” Tools (glimpse)

Cookies Managers

- Cookie Managers - FF / Google Chrome Plugin
- Cookies can be viewed / edited
- "Monitor Cookies" allows you to view the adds, changes, deletion in real time (as you navigate through a website)

Firewalk

- Gives a you an idea of the firewall state / access control lists / etc
- Scans ports (and associated services behind them) that you may want to know about



NMap (Network Mapper)

- Port Scanner
- On the local system you can just run "netstat", obviously on a remote system this is not possible
- Once we ascertain what ports are open we can determine what services are running (e.g. 80 for HTTP, 25 for email, etc)
- Run "man nmap" to view the manual
- It can also help in determining the webserver (IIS, Nginx, Apache, etc)
- Can run "Lua" scripts by running them within the nmap scripting engine
- If you don't specific a port it defaults to approximately 1,000 as part of the scan (nmap -sT x.x.x.x)

Wireshark

- Packet capture analysis
- Shows literally ALL network traffic in / out of any given system that you run it on
- These kind of tools used to cost a lot of money (Wireshark used to be called Ethereal)
- Now free and open source :)
- Open it in Kali and do "ping" to generate some traffic
- Does a great job of splitting out the packets (based on the various OSI layers) and making that searchable ... explorable
- Shoes the raw packets at the bottom of the screen and reassembles them / note how it highlights the packets as you explore

3. Gaining Access

Gaining Access

- To what? It comes down to your motivations
- Obviously in our case we're trying to secure our applications. Alternatively you might be doing a review this on behalf of a 3rd company.
- Or…

Gaining Access

- “Gaining access requires taking control of one or more network devices in order to either extract data from the target, or to use that device to then launch attacks on other targets.”, cybrary.it

OWASP

OWASP

- https://www.owasp.org
- The Open Web Application Security Project (OWASP) is a 501(c)(3) worldwide not-for-profit charitable organization focused on improving the security of software. Our mission is to make software security visible,so that individuals and organizations are able to make informed decisions.

OWASP

- Offers services / features such as…
  - Development & Testing Guides
  - Zed Attack Proxy (ZAP)
  - WebGoat
  - OWASP Top 10
  - Etc

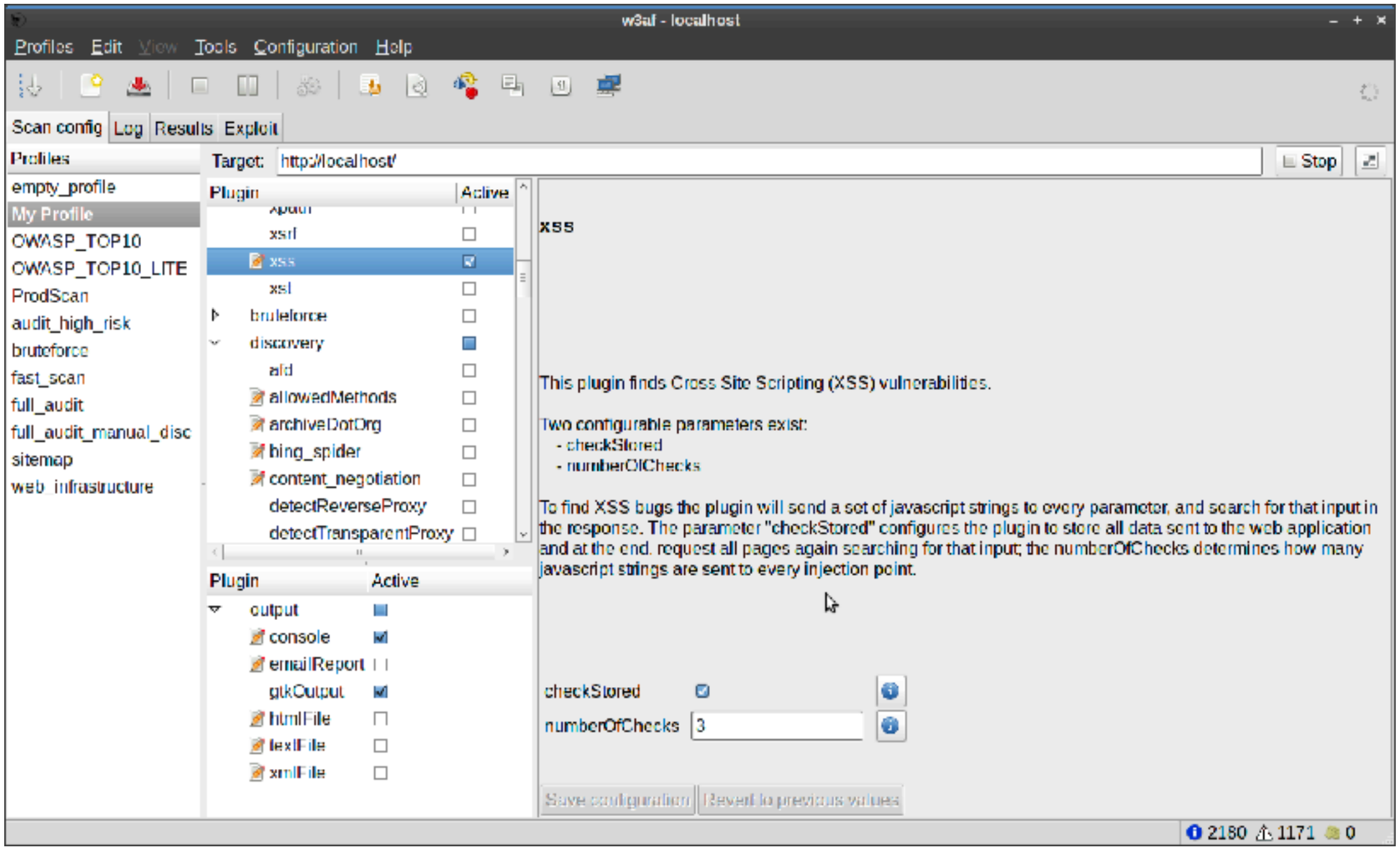
OWASP Top 10

OWASP Top 10 – 2013 (Previous)	OWASP Top 10 – 2017 (New)
A1 – Injection	A1 – Injection
A2 – Broken Authentication and Session Management	A2 – Broken Authentication and Session Management
A3 – Cross-Site Scripting (XSS)	A3 – Cross-Site Scripting (XSS)
A4 – Insecure Direct Object References - Merged with A7	A4 – Broken Access Control (Original category in 2003/2004)
A5 – Security Misconfiguration	A5 – Security Misconfiguration
A6 – Sensitive Data Exposure	A6 – Sensitive Data Exposure
A7 – Missing Function Level Access Control - Merged with A4	A7 – Insufficient Attack Protection (NEW)
A8 – Cross-Site Request Forgery (CSRF)	A8 – Cross-Site Request Forgery (CSRF)
A9 – Using Components with Known Vulnerabilities	A9 – Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards - Dropped	A10 – Underprotected APIs (NEW)

W3AF

W3AF

- http://w3af.org
- “W3AF is a Web Application Attack and Audit Framework which aims to identify and exploit all web application vulnerabilities.”
- Under Kali / Web Application Testing
- Literally a whole toolkit designed for finding weaknesses in your Web Application
- You can also create your own modules / attacks


w3af - localhost
Profiles Edit View Tools Configuration Help
Scan config Log Results Exploit
Profiles
empty_profile
My Profile
OWASP_TOP10
OWASP_TOP10_LITE
ProdScan
audit_high_risk
bruteforce
fast_scan
full_audit
full_audit_manual_disc
sitemap
web_infrastructure
Target: http://localhost/
Stop
Plugin
Active
xsrf
xss
xsl
bruteforce
discovery
afd
allowedMethods
archiveDotOrg
bing_spider
content_negotiation
detectReverseProxy
detectTransparentProxy
Plugin
Active
output
console
emailReport
gtkOutput
htmlFile
textFile
xmlFile
XSS
This plugin finds Cross Site Scripting (XSS) vulnerabilities.
Two configurable parameters exist:
- checkStored
- numberOfChecks
To find XSS bugs the plugin will send a set of javascript strings to every parameter, and search for that input in the response. The parameter "checkStored" configures the plugin to store all data sent to the web application and at the end, request all pages again searching for that input; the numberOfChecks determines how many javascript strings are sent to every injection point.
checkStored
numberOfChecks
3
Save configuration
Revert to previous values
2180
1171
0

Starting It Up

- Open Parallels | Ubuntu
- /home/parallels/w3af
- > ./w3af_gui
- Set the target: http://miami.bluer.com (a sample node.js app running on Vultr VPS)
  - > cd /var/www/hkjs-node-hello-universe
  - > node app
- “Hello World” via robots.txt check

Example: Crawling

- Crawling for undocumented / link routes (e.g. a hidden admin panel)
- Under Crawl | dir_file_bruter
- Takes word lists as parameters (which will look a large number of possible paths)
  - /myadmin/
  - /admin
  - /admin-panel

parallels@ubuntu: ~/w3af

```
GET http://miami.bluer.com/y3u40T8q.jsp returned HTTP code "404" (id=24,from_cache=0,grep=0)
GET http://miami.bluer.com/pILpi2BE.py returned HTTP code "404" (id=25,from_cache=0,grep=0)
GET http://miami.bluer.com/Sc5q6PSJ.pl returned HTTP code "404" (id=26,from_cache=0,grep=0)
GET http://miami.bluer.com/1138pc3e.htmls returned HTTP code "404" (id=27,from_cache=0,grep=0)
GET http://miami.bluer.com/t111V6dy.foobar returned HTTP code "404" (id=30,from_cache=0,grep=0)
GET http://miami.bluer.com/KmlYDlo9.do returned HTTP code "404" (id=32,from_cache=0,grep=0)
GET http://miami.bluer.com/0mCZcMUh.htm returned HTTP code "404" (id=34,from_cache=0,grep=0)
GET http://miami.bluer.com/Ktsjt2W4.php returned HTTP code "404" (id=28,from_cache=0,grep=0)
GET http://miami.bluer.com/q2EphVJW.rb returned HTTP code "404" (id=29,from_cache=0,grep=0)
GET http://miami.bluer.com/T1yXa0It.aspx returned HTTP code "404" (id=33,from_cache=0,grep=0)
GET http://miami.bluer.com/ZhoUlJEE.xhtml returned HTTP code "404" (id=31,from_cache=0,grep=0)
The 404 body result database has a length of 1.
"http://miami.bluer.com/" (id:19) is NOT a 404 [similarity_index < 0.75 with sample path in 404 DB].
GET http://miami.bluer.com/ returned HTTP code "200" (id=35,from_cache=1,grep=1)
dir_file_bruter plugin is testing: "Method: GET | http://miami.bluer.com/"
Called _discover_worker(dir_file_bruter,http://miami.bluer.com/)
Starting plugin: crawl.dir_file_bruter
dir_file_bruter is testing "http://miami.bluer.com/"
[dir_file_bruter] Crawling "http://miami.bluer.com/"
Updating socket timeout for miami.bluer.com from 6 to 3.3680617639 seconds
GET http://miami.bluer.com/support/ returned HTTP code "404" (id=36,from_cache=0,grep=1)
GET http://miami.bluer.com/in/ returned HTTP code "404" (id=37,from_cache=0,grep=1)
GET http://miami.bluer.com/people/ returned HTTP code "404" (id=38,from_cache=0,grep=1)
GET http://miami.bluer.com/browse/ returned HTTP code "404" (id=39,from_cache=0,grep=1)
GET http://miami.bluer.com/20/ returned HTTP code "404" (id=41,from_cache=0,grep=1)
GET http://miami.bluer.com/publications/ returned HTTP code "404" (id=40,from_cache=0,grep=1)
GET http://miami.bluer.com/documents/ returned HTTP code "404" (id=43,from_cache=0,grep=1)
GET http://miami.bluer.com/pics/ returned HTTP code "404" (id=44,from_cache=0,grep=1)
GET http://miami.bluer.com/other/ returned HTTP code "404" (id=42,from_cache=0,grep=1)
GET http://miami.bluer.com/books/ returned HTTP code "404" (id=45,from_cache=0,grep=1)
GET http://miami.bluer.com/index/ returned HTTP code "404" (id=46,from_cache=0,grep=1)
GET http://miami.bluer.com/30/ returned HTTP code "404" (id=47,from_cache=0,grep=1)
GET http://miami.bluer.com/content/ returned HTTP code "404" (id=48,from_cache=0,grep=1)
GET http://miami.bluer.com/strona_5/ returned HTTP code "404" (id=49,from_cache=0,grep=1)
GET http://miami.bluer.com/cgi-bin/ returned HTTP code "404" (id=50,from_cache=0,grep=1)
GET http://miami.bluer.com/sponsors/ returned HTTP code "404" (id=51,from_cache=0,grep=1)
GET http://miami.bluer.com/strona_21/ returned HTTP code "404" (id=52,from_cache=0,grep=1)
GET http://miami.bluer.com/art/ returned HTTP code "404" (id=53,from_cache=0,grep=1)
GET http://miami.bluer.com/it/ returned HTTP code "404" (id=54,from_cache=0,grep=1)
GET http://miami.bluer.com/service/ returned HTTP code "404" (id=55,from_cache=0,grep=1)
```

JS

Example: Crawling

- Output from a crawl against DVWA (which we'll see in a moment)
- Gives you a range of endpoints to attack / protect

DVWA (‘Sploit Time)

DVWA

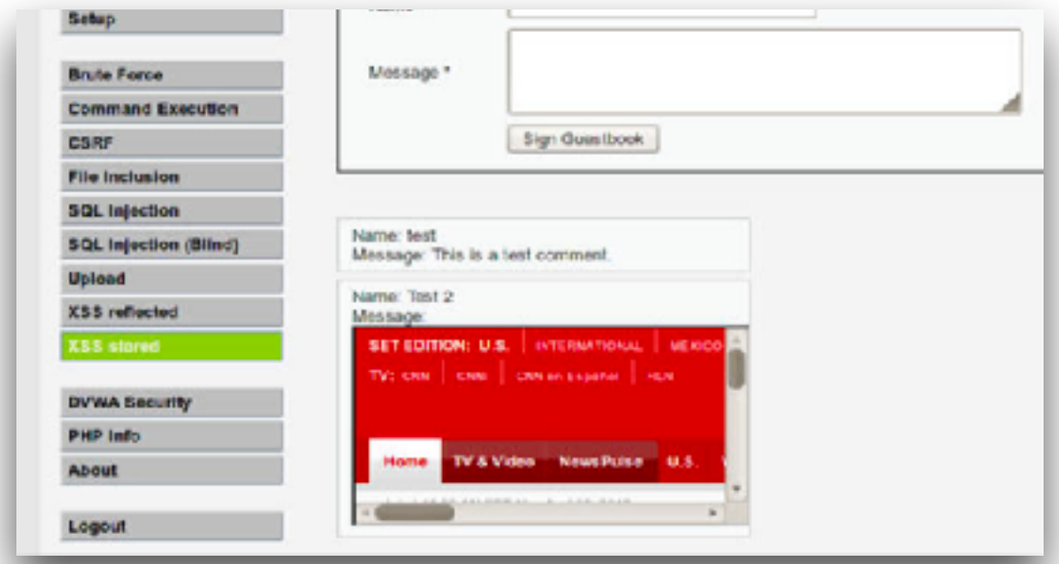
- http://www.dvwa.co.uk
- https://github.com/ethicalhack3r/DVWA
- DVWA (Damn Vulnerable Web Application)
- “Damn Vulnerable Web App (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goals are to be an aid for security professionals to test their skills and tools in a legal environment”

SQL Injection

- Execute arbitrary commands against the database
- > %' and 1=0 union select null, concat(first_name, 0x0a,last_name,0x0a,user,0x0a,password) from users #
- More details

XSS

- http://192.168.99.120:8888/vulnerabilities/xss_s/
- <iframe src="http://www.cnn.com"></iframe>
- Allow for a custom script to be injected / cookie to be hijacked (more)



CSRF

- If I can get someone to click on the pre-crafted link such as one that contains the following:

index.html

csrf_bypass_brute_force.py | index.html

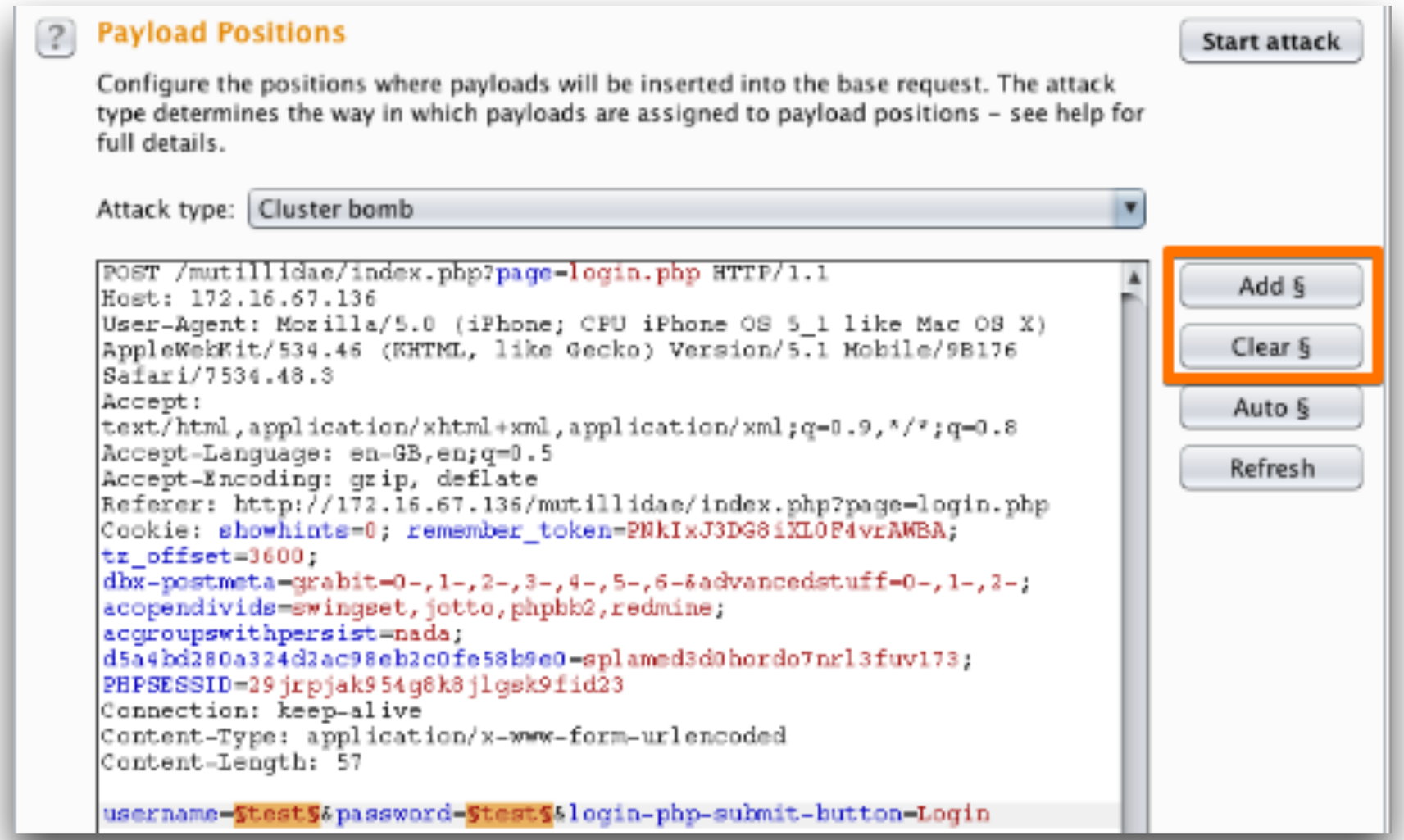
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Bad Guy Site</title>
</head>
<body>
  <h1>Hey there. Definitely nothing sketchy happening here.</h1>
  <img style="display:none" src="http://192.168.0.11/dvwa/vulnerabilities/csrf/?password_new=pwned&
  password_conf=pwned&Change=Change" alt="">
</body>
</html>
```

- The password will be changed to “pwned”

Brute Force (via Burp)

- Get Burp to act as the proxy for a the traffic
- Try to login to DVWA
- Brute force via the "Intruder" tab / note that you have to set the parameters that you care about in the post request / also the lists for username and passwords (combined it might be in the range of 3m for example). Positions and payloads respectively
- You can also do the same for Session IDs, etc
- Upon success it will return a HTTP 200 vs 302
- https://support.portswigger.net/customer/portal/articles/1964020-using-burp-to-brute-force-a-login-page

Brute Force (via Burp)



Practice / Hands On

Practice / Hands On

- Work with LEGAL practice tools :)
- DVWA - PHP / MySQL based with common vulnerabilities such as CSRF, XSS, etc
- OWASP WebGoat - Java on the back and Backbone, Underscore, jQuery on the front ... provides hints along the way :)
- Gruyere (cheese with holes in it) - Python based with lots of great precanned examples.

What's Next?

What's Next?

- Practice, training, etc
- Be careful / legal / etc
- Future Hong Kong JS sessions :)

Thank You / Q&A