

Mason Competitive Cyber

Web App Sec: Crash Course for Beginners



Playground

- There are a ton of sites and problems to get introduced
 - **@1pwnch's TCTF problems**
 - XSS Game
 - Game of Hacks
 - Damn Vulnerable Web App
 - OWASP (God-Tier web app sec wiki)



Recon and Scanning

- Recon and Scanning
 - Recon - Gathering information about a target
 - Scanning - Somewhat subjective
- How does it relate to real world work?
 - Any non-shit app requires more than 3 slides and random tools to hack
 - *study on your own as well*
 - There's no reason to run attacks that aren't applicable



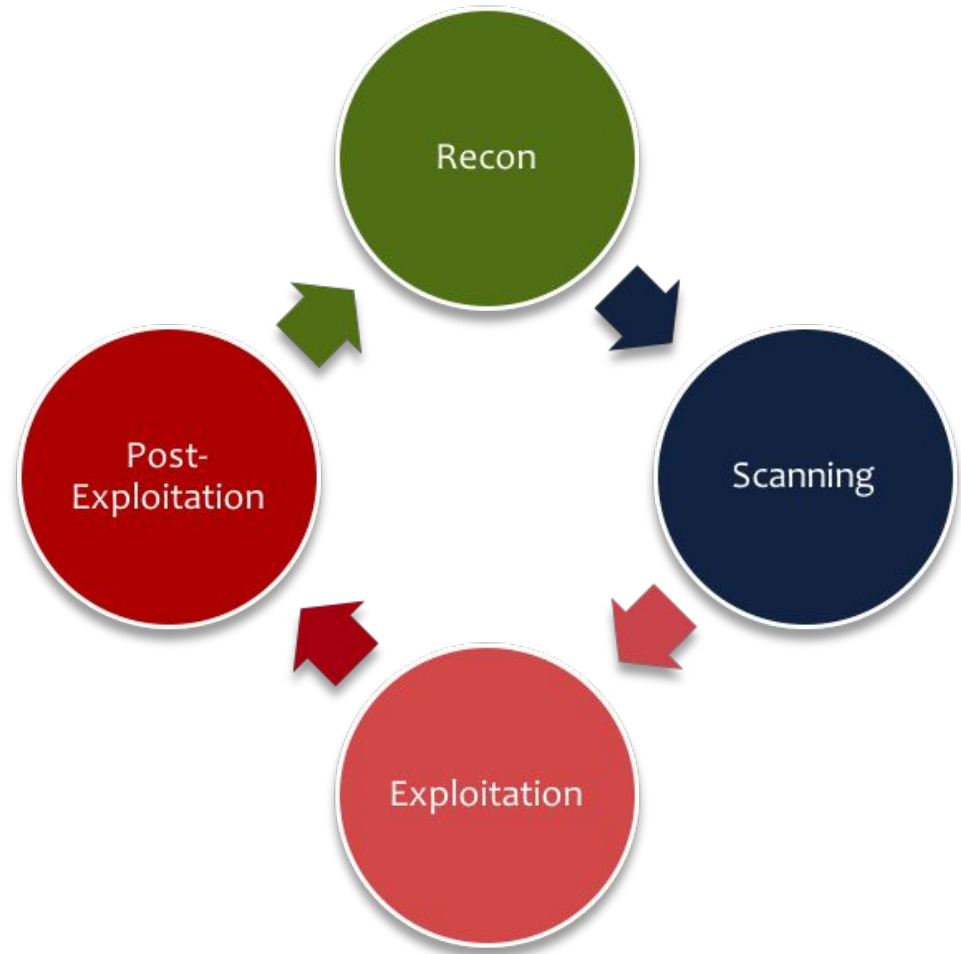
Different Levels of Impact

- What is your ultimate goal?
 - Are you on a bug bounty?
 - What's in scope? Is chaining authorized?
 - CTF?
 - What are they hinting at? What is the competition's context?
 - Pentest?
 - Impact they care about? Rules of engagement? (I guess?)
- This only follows a **subset** of the ways to recon and scan a website
- Templating bugs vs content injection vs domain takeover vs command injection
 - All varying levels of impact



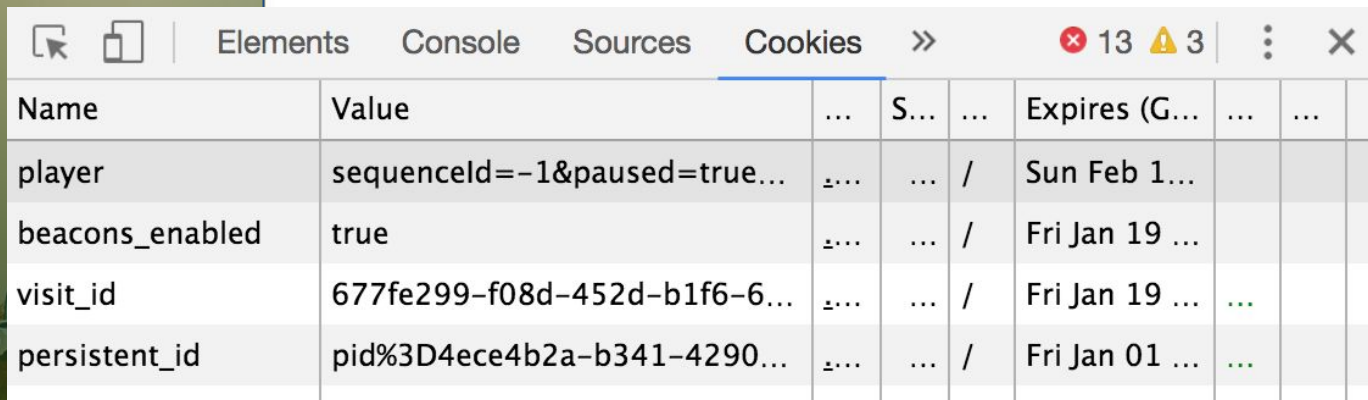
Understanding Cycle

- Where does web fit in the exploitation cycle?



(Chrome) Extensions To Use

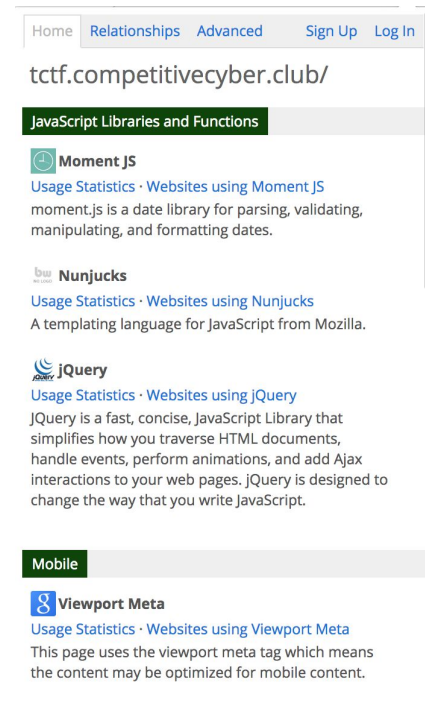
- BuiltWith - Discover technologies a site is built with
- EditThisCookie or CookieInspector - Manage cookies
- ModHeader - Modify headers you send to a server
- XSS Radar or comparable scanners (I don't personally use this)



Name	Value	...	S...	...	Expires (G...
player	sequenceId=-1&paused=true...	/	Sun Feb 1...			
beacons_enabled	true	/	Fri Jan 19 ...			
visit_id	677fe299-f08d-452d-b1f6-6...	/	Fri Jan 19		
persistent_id	pid%3D4ece4b2a-b341-4290...	/	Fri Jan 01		

CookieInspector

BuiltWith



Home Relationships Advanced Sign Up Log In

tctf.competitivecyber.club/

JavaScript Libraries and Functions

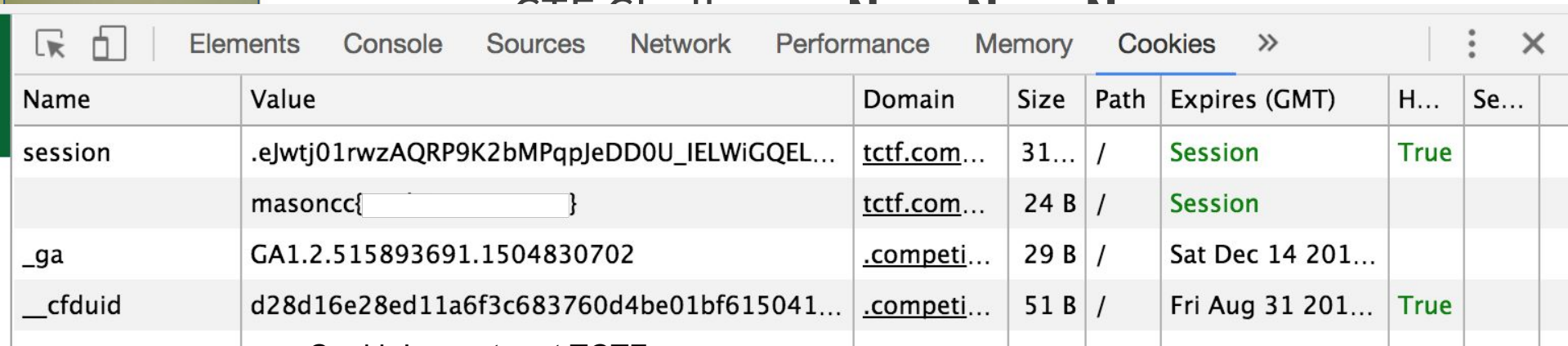
- Moment JS**
Usage Statistics · Websites using Moment JS
moment.js is a date library for parsing, validating, manipulating, and formatting dates.
- Nunjucks**
Usage Statistics · Websites using Nunjucks
A templating language for JavaScript from Mozilla.
- jQuery**
Usage Statistics · Websites using jQuery
jQuery is a fast, concise, JavaScript Library that simplifies how you traverse HTML documents, handle events, perform animations, and add Ajax interactions to your web pages. jQuery is designed to change the way that you write JavaScript.

Mobile

- Viewport Meta**
Usage Statistics · Websites using Viewport Meta
This page uses the viewport meta tag which means the content may be optimized for mobile content.

Using Said Extensions

- Builtwith is self-explanatory, install, click on it, it tells you technologies
- Any cookie reader if you don't have Chrome suffices
 - Run **document.cookie** in Console if that's not even an option to dump them



The screenshot shows the Chrome DevTools interface with the 'Cookies' tab selected. The table below lists the cookies found on the page.

Name	Value	Domain	Size	Path	Expires (GMT)	H...	Se...
session	.ejwtj01rwzAQRp9K2bMPqpJeDD0U_IELWiGQEL...	tctf.com...	31...	/	Session	True	
	masoncc{_____}	tctf.com...	24 B	/	Session		
_ga	GA1.2.515893691.1504830702	.competi...	29 B	/	Sat Dec 14 201...		
__cfduid	d28d16e28ed11a6f3c683760d4be01bf615041...	.competi...	51 B	/	Fri Aug 31 201...	True	

CookieInspector at TCTF

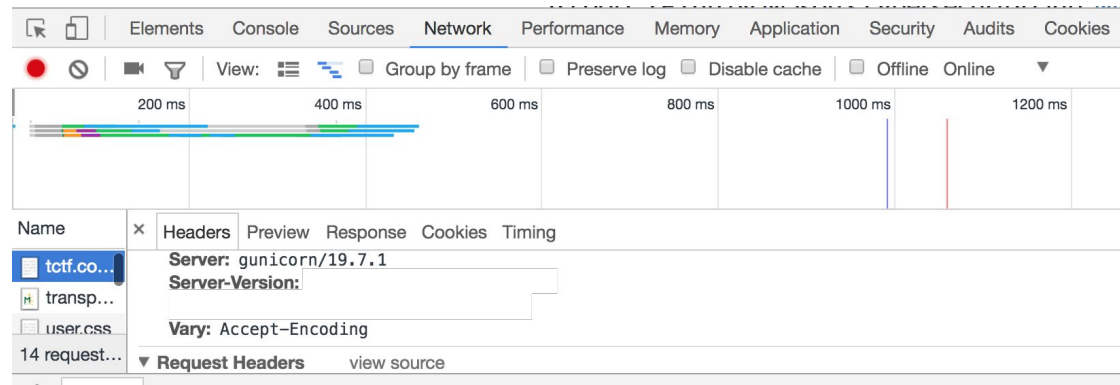
Using What You Have

- **Actually visit the site**
 - Get application version, it may not broadcast it in a technical format
 - Related challenge: A few...
- Google/research often and carefully
- **Use built in developer tab**, we'll be using it a few times here
 - Safari: Require you go to advanced preferences, enable **Develop**, then select it from the top nav menu
 - Chrome: Dots to top right -> More tools -> Developer Tools
 - Firefox: Similar to chrome, 3 tabs -> Developer -> Select Option like Web Console
 - **Each looks very similar once you get to it**



More on Network Tab

- **Related CTF Challenge: Heads Up**
- Click on interactions to view details about them, both their request and response
 - In Chrome, **Headers** and **Response**
- Good for determining higher level versions of what the sites running, such as language, web server, HTTP version, any sort of caching, etc
- Pay special attention to Headers in TCTF when you load it.



Dir Busting

- Appeared in a HackEd CTF challenge
- Brute forces files and directories in a website
- /uploads, /admin, etc among app-specific ones
- Uses a “wordlist”, a list of words to try, also used in our dnsrecon demo

This tctf problem got taken down :(

Using Windows? Run DirBuster
Using Linux (or MacOS)? Run **dirb**



Dirb Run Example

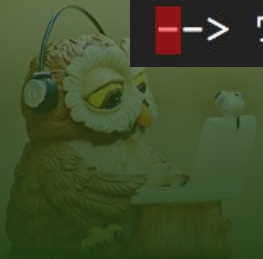
```
root@cloudshell:~$ dirb https://tctf.competitivecyber.club
```

```
-----  
DIRB v2.22  
By The Dark Raver  
-----
```

```
START_TIME: Fri Jan 19 01:25:00 2018  
URL_BASE: https://tctf.competitivecyber.club/  
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
```

```
-----  
  
GENERATED WORDS: 4612
```

```
---- Scanning URL: https://tctf.competitivecyber.club/ ----  
==> DIRECTORY: https://tctf.competitivecyber.club/[REDACTED]  
[-> Testing: https://tctf.competitivecyber.club/25
```



DNS Brute Forcing

- In bug bounties and pentests, I recommend **dnsdumpster.com** in addition to this technique
- Simply run **dnsrecon** to expose options

```
root@cloudshell:~$ dnsrecon
Version: 0.8.10
Usage: dnsrecon.py <options>

Options:
  -h, --help                Show this help message and exit.
  -d, --domain <domain>    Target domain.
  -r, --range <range>      IP range for reverse lookup brute force in formats (first-last) or in (range/bitmask).
  -s, --name-server <name> Domain server to use. If none is given, the SOA of the target will be used.
  -D, --dictionary <file> Dictionary file of subdomain and hostnames to use for brute force.
  -f, --filter <filter>    Filter out of brute force domain lookup, records that resolve to the wildcard defined
                           IP address when saving records.
  -t, --type <types>      Type of enumeration to perform:
                           std      SOA, NS, A, AAAA, MX and SRV if AXFR on the NS servers fail.
                           rvl      Reverse lookup of a given CIDR or IP range.
                           brt      Brute force domains and hosts using a given dictionary.
                           srv      SRV records.
                           axfr     Test all NS servers for a zone transfer.
                           goo       Perform Google search for subdomains and hosts.
                           snoop    Perform cache snooping against all NS servers for a given domain, testing
                           all with file containing the domains, file given with -D option.
                           tld       Remove the TLD of given domain and test against all TLDs registered in IANA.
                           zonewalk Perform a DNSSEC zone walk using NSEC records.
  -a                        Perform AXFR with standard enumeration.
  -s                        Perform a reverse lookup of IPv4 ranges in the SPF record with standard enumeration.
  -g                        Perform Google enumeration with standard enumeration.
  -w                        Perform deep whois record analysis and reverse lookup of IP ranges found through
                           Whois when doing a standard enumeration.
  -z                        Performs a DNSSEC zone walk with standard enumeration.
  --threads <number>       Number of threads to use in reverse lookups, forward lookups, brute force and SRV
                           record enumeration.
  --lifetime <number>      Time to wait for a server to response to a query.
  --db <file>              SQLite 3 file to save found records.
  --xml <file>             XML file to save found records.
  --iw                     Continue brute forcing a domain even if a wildcard records are discovered.
  -c, --csv <file>         Comma separated value file.
  -j, --json <file>        JSON file.
  -v                       Show attempts in the brute force modes.
```

DNS Brute Forcing Command

- **Related CTF Challenge: A Record to Remember**
- `dnsrecon -D /tmp/dnslist -d tctf.competitivecyber.club -t brt`
- Should produce any records in the wordlist
- Some wordlists are much larger, and as such take longer to run
 - This one is *quite* small

```
-v                               show attempts in the brute force modes.
root@cloudshell:~$ dnsrecon -D /tmp/dnslist -d tctf.competitivecyber.club -t brt
[*] Performing host and subdomain brute force against tctf.competitivecyber.club
[*]      A [REDACTED] 54.172.0.227
[*] 1 Records Found
root@cloudshell:~$
```

Key Terms

- Session: A user's browsing session tracked by the server, including their login status and information about their login
- Session ID: A sensitive value (stored in a cookie) that is used to identify a user's session in the browser
- Password hash: One-way encrypted (cannot be “decrypted”, but can be brute forced, usually easily) password of a user
- Session Hijack: Stealing someone's session
- Request: Data sent from browser to server
 - GET Request: Usually non-sensitive, sent in the URL
 - POST Request: Often sensitive, not sent in URL, often form data



Cross Site Scripting

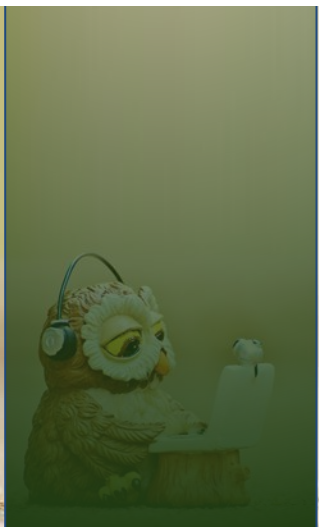
- Referred to as **XSS**
- Running your own Javascript on another user's browser
- Frequently used to impersonate users/"session hijack"
- Three kinds:
 - Stored
 - Stored on the server somewhere such as the database where it's retrieved at a later time
 - Much more dangerous, like if it's something like your First Name on a profile page
 - Reflected
 - Sent to the server and returned, such as in the URL
 - DOM-based
 - Leverages the "DOM", so basically existing Javascript or HTML



Code Example

```
<?php

// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Feedback for end user
    echo '<pre>Hello ' . $_GET[ 'name' ] . '</pre>';
}
```



SQL Injection

- Injecting your own code into a database statement, such as with a login
- Two major kinds:
 - Blind
 - You don't see output from what you've done
 - Normal
 - You do see your output

```
// Get input
$id = $_POST[ 'id' ];

// Check database
$query = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
```

SQLMap

- **You don't need to know SQL**
- Have to find vulnerable request, such as the first name or username or something
- In our VM and Kali by default
- **sqlmap --url**
http://yoururl.com/page?id=1 -p id --dbs
 - **Enumerates DBs assuming id is unsafe**
- Can use it to get an OS shell, etc
- **Was used at VT Summit like last week**



Command Injection

- Similar to SQL injection but commands
- Appeared in CyberFusion

```
// Get input
$target = $_REQUEST[ 'ip' ];

$cmd = shell_exec( 'ping ' . $target );
```



File Upload

- When people don't check their uploads and you can upload PHP
- Avoid using web shells like an idiot in competition
 - People don't have to upload their own if they can browse to /shell.php and be done with it
- Common in defense competitions to remove
- **b374k, c99, etc** are common web shells
 - RARELY will actually get caught by AV



File Include

- When a programmer relies on user input to fetch files
- If you visit a page or an image is loaded and it's something like **page.php?file=dog.jpg**

```
<?php  
  
// The page we wish to display  
$file = $_GET[ 'page' ];
```

In this example, we'd need to display `$file` with something like **`echo file_get_contents($file);`**



Outdated Components

- People almost never update their CMses
 - WordPress
 - Joomla
 - Drupal
 - There are exceptions, like presumably the White House
- Can scan with
- For smaller apps and other platforms, **research their version number**
 - WebCal 1.2.4 has shown up in 2 CTFs as well as CTF now
 - It has working exploit code
- **Be ready to chop up exploits to meet your needs**



Post Exploitation

- **You might not even be in a full OS**
 - VT Summit had a VM with busybox
 - TCTF uses slim Docker containers
 - Slim OSes exist

Common commands to exfil:

nc, openssl, perl, python, curl, wget, dig

I use curl and wget, a lot of people don't.

Worry about how you're going to get it over the wire, and how to format the data



Understanding Access

- **Where the hell are you?**
 - A container? A VM?
- **Who the hell are you?**
 - A dedicated web server user in most sane cases like *www-data* or *apache*
- **What the hell are you?**
 - Do you have shell? Is it bash or just shell? Are you injecting application code like PHP? If you have SQL injection, can you pivot to something like code execution? What are the differences and why will that make scripting a pain?



Commonly Stolen Artifacts

- Password files/hashes
- Anything to lend “persistence”
- User enumeration, server configurations
- User data

