Mason Competitive Cyber

Introduction to

Web Exploitation

Andy Smith (@andy)

News

MasonCC got 1st place in the
VMI Cyber Cup!

Upcoming CTFs

- UMDCTF: March 4, this Friday. #umdctf2022
- MACCDC Finals: March 17-19
- PatriotCTF: April 29 @ 5pm - May 1 @ 5pm
  https://competitivecyber.club/patriotctf/

Elections are coming up soon!

Agenda

- What is “Web”?
- Databases
- Types of Web Attacks
    - Directory Traversal
    - XSS
    - CSRF
    - SQLi
    - NoSQLi
    - Command Injection
    - SSRF
    - XXE
- Your turn!

What is “Web”?

- Web server
  - Apache HTTPd
  - Nginx (“Engine X”)
  - Lighttpd (“Lighty”)
  - Microsoft IIS
  - Others (Tomcat, Caddy, Jetty, OpenResty)
- Can run:
  - Static content: HTML, JS, CSS
    - Boring
  - Dynamic Content: PHP, Flask, Ruby on Rails, Struts, Node.js, Django, ASP.NET, etc.
    - Relies on a backend server like a CGI (e.g. PHP => php_fpm, Flask/Python => wsgi) or implemented in the backend server itself
    - Usually uses a database
    - *This is what we attack!*
    - Often, web exploitation relies on understanding the backend programming language

Databases

- Relational
  - Tables, rows, columns
  - Rigidly defined structure
  - E.g. MySQL / MariaDB, Postgres, Microsoft SQL Server (MSSQL), SQLite
- NoSQL
  - Tableless, much more loosely defined
  - MongoDB uses “documents” which are JSON objects
  - E.g. MongoDB, Cassandra
- Key-value
  - Versatile, but primarily for speeding up tasks & caching
  - E.g. Redis, Amazon DynamoDB, Memcached
- …

Web concepts you should know

- HTTP Verbs: GET, POST, HEAD, OPTIONS, etc.
- GET vs POST
- Cookies (Secure and HttpOnly flags)
- AJAX
- robots.txt
- Basic authentication
- Multiple websites can run on one machine
- HTTPS certificates
- Request and response (next slide)

Request and Response

Request

GET / HTTP/1.1
Host: mymasonportal.gmu.edu
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/98.0.4758.82 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9

^ all of these values are user-changeable

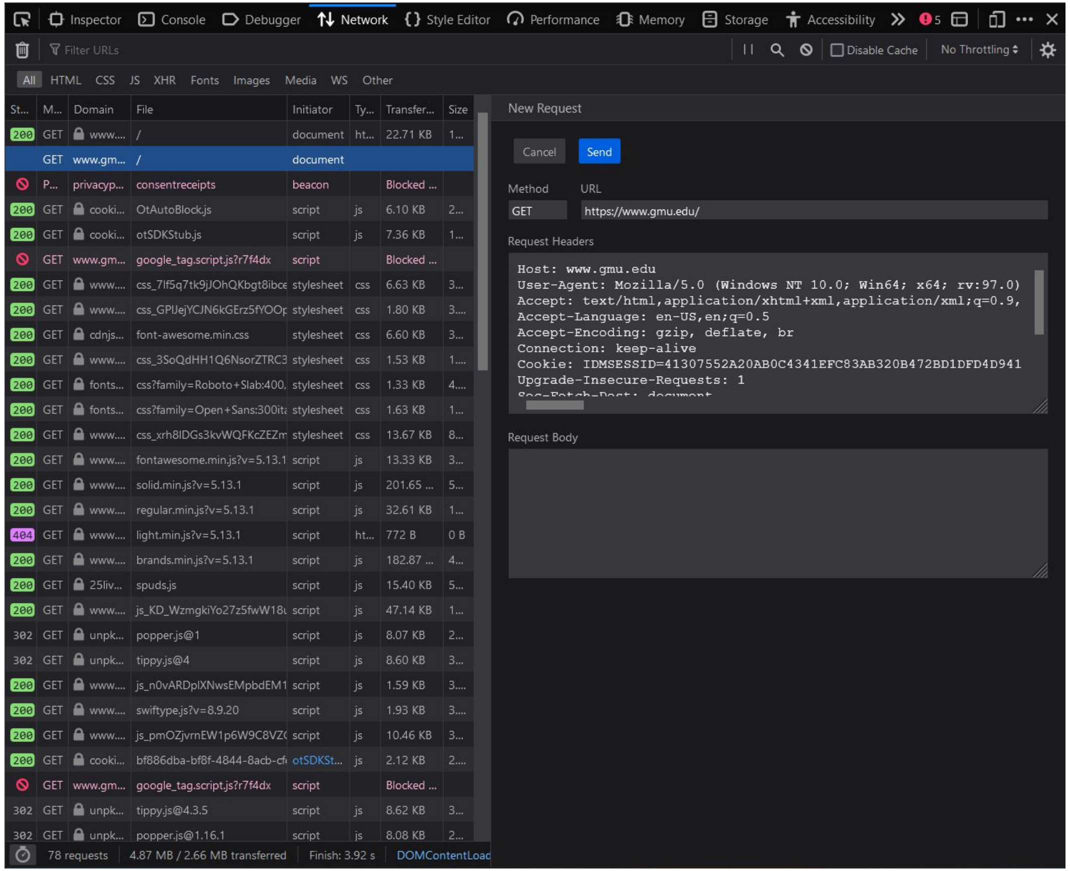
Response

HTTP/1.1 200
Content-Encoding: gzip
Content-Language: en-US-390021c
Content-Security-Policy: frame-ancestors 'self' https://*.gmu.edu
Content-Type: text/html;charset=UTF-8
Date: Mon, 14 Feb 2022 08:56:37 GMT
Expires: Sun, 14 Feb 2021 08:56:37 GMT
Last-Modified: Thu, 14 Feb 2002 08:56:37 GMT
Server: openresty/1.15.8.3
Set-Cookie: JSESSIONID=0C05B59655E346B9CE98984F41E03917; Path=/; Secure
X-Blackboard-appserver: ip-10-145-61-20.ec2.internal
X-Blackboard-product: Blackboard Learn &#8482; 3900.32.0-rel.31+a606f03
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1
Content-Length: 19202
Connection: keep-alive

How can we edit these requests?

Tools!

- Firefox Devtools
- Burp Suite
- OWASP ZAP
- curl



St...	M...	Domain	File	Initiator	Ty...	Transfer...	Size
200	GET	🔒 www....	/	document	ht...	22.71 KB	1...
	GET	www.gm...	/	document			
🚫	P...	privacyp...	consentreceipts	beacon		Blocked ...	
200	GET	🔒 cooki...	OtAutoBlock.js	script	js	6.10 KB	2...
200	GET	🔒 cooki...	otSDKStub.js	script	js	7.36 KB	1...
🚫	GET	www.gm...	google_tag.script.js?r7f4dx	script		Blocked ...	
200	GET	🔒 www....	css_7lf5q7tk9jJOhQKbgt8ibce	stylesheet	css	6.63 KB	3...
200	GET	🔒 www....	css_GPlJejYCJN6kGErz5fYOOp	stylesheet	css	1.80 KB	3....
200	GET	🔒 cdnjs...	font-awesome.min.css	stylesheet	css	6.60 KB	3...
200	GET	🔒 www....	css_3SoQdHH1Q6NsorZTRC3	stylesheet	css	1.53 KB	1....
200	GET	🔒 fonts...	css?family=Roboto+Slab:400,	stylesheet	css	1.33 KB	4....
200	GET	🔒 fonts...	css?family=Open+Sans:300ita	stylesheet	css	1.63 KB	1...
200	GET	🔒 www....	css_xrh8IDGs3kvWQFKcZEZm	stylesheet	css	13.67 KB	8...
200	GET	🔒 www....	fontawesome.min.js?v=5.13.1	script	js	13.33 KB	3...
200	GET	🔒 www....	solid.min.js?v=5.13.1	script	js	201.65 ...	5...
200	GET	🔒 www....	regular.min.js?v=5.13.1	script	js	32.61 KB	1...
404	GET	🔒 www....	light.min.js?v=5.13.1	script	ht...	772 B	0 B
200	GET	🔒 www....	brands.min.js?v=5.13.1	script	js	182.87 ...	4...
200	GET	🔒 25liv...	spuds.js	script	js	15.40 KB	5...
200	GET	🔒 www....	js_KD_WzmgkiYo27z5fwW18u	script	js	47.14 KB	1...
302	GET	🔒 unpk...	popper.js@1	script	js	8.07 KB	2...
302	GET	🔒 unpk...	tippy.js@4	script	js	8.60 KB	3...
200	GET	🔒 www....	js_n0vARDplXNwsEMpbdEM1	script	js	1.59 KB	3....
200	GET	🔒 www....	swiftype.js?v=8.9.20	script	js	1.93 KB	3....
200	GET	🔒 www....	js_pmOZjvrnEW1p6W9C8VZ(	script	js	10.46 KB	3...
200	GET	🔒 cooki...	bf886dba-bf8f-4844-8acb-cf	otSDKSt...	js	2.12 KB	2....
🚫	GET	www.gm...	google_tag.script.js?r7f4dx	script		Blocked ...	
302	GET	🔒 unpk...	tippy.js@4.3.5	script	js	8.62 KB	3...
302	GET	🔒 unpk...	popper.js@1.16.1	script	js	8.08 KB	2...

78 requests | 4.87 MB / 2.66 MB transferred | Finish: 3.92 s | DOMContentLoad

New Request

Cancel | Send

Method: GET

URL: https://www.gmu.edu/

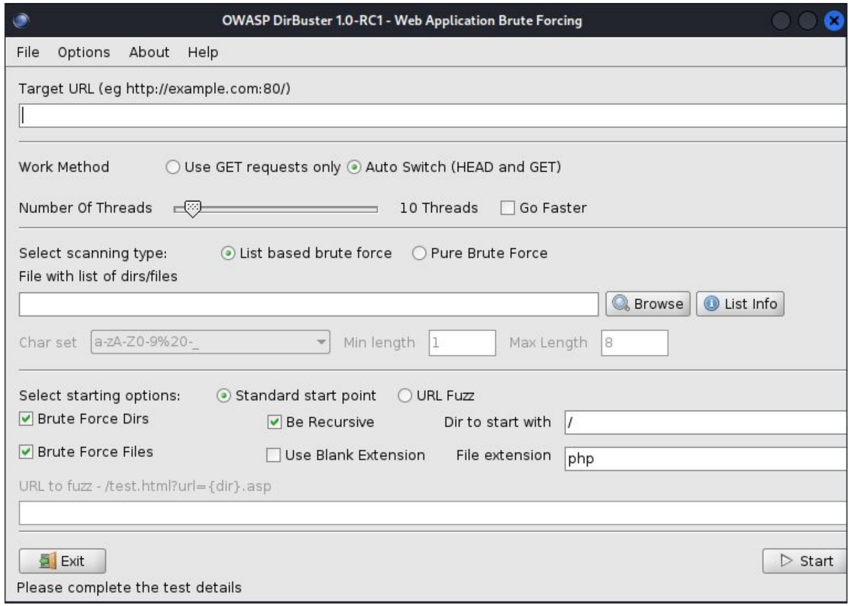
Request Headers

```
Host: www.gmu.edu
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:97.0)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Cookie: IDMSESSID=41307552A20AB0C4341EFC83AB320B472BD1DFD4D941
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
```

Request Body

Dirbusting

- Directory and domain brute-forcing
- Used to find URLs and subdomains
- VERY noisy
- Tools:
  - Dirbuster (out of date, but has GUI)
  - Dirb
  - Gobuster
  - ffuf
  - Wfuzz
  - Dirsearch
  - … there's a lot



Target URL (eg http://example.com:80/)

Work Method: Use GET requests only [✓ Auto Switch (HEAD and GET)]

Number Of Threads: 10 Threads ☐ Go Faster

Select scanning type: [✓ List based brute force] Pure Brute Force

File with list of dirs/files

Char set a-zA-Z0-9%20-_ Min length 1 Max Length 8

Select starting options: [✓ Standard start point] URL Fuzz

☑ Brute Force Dirs ☑ Be Recursive Dir to start with /

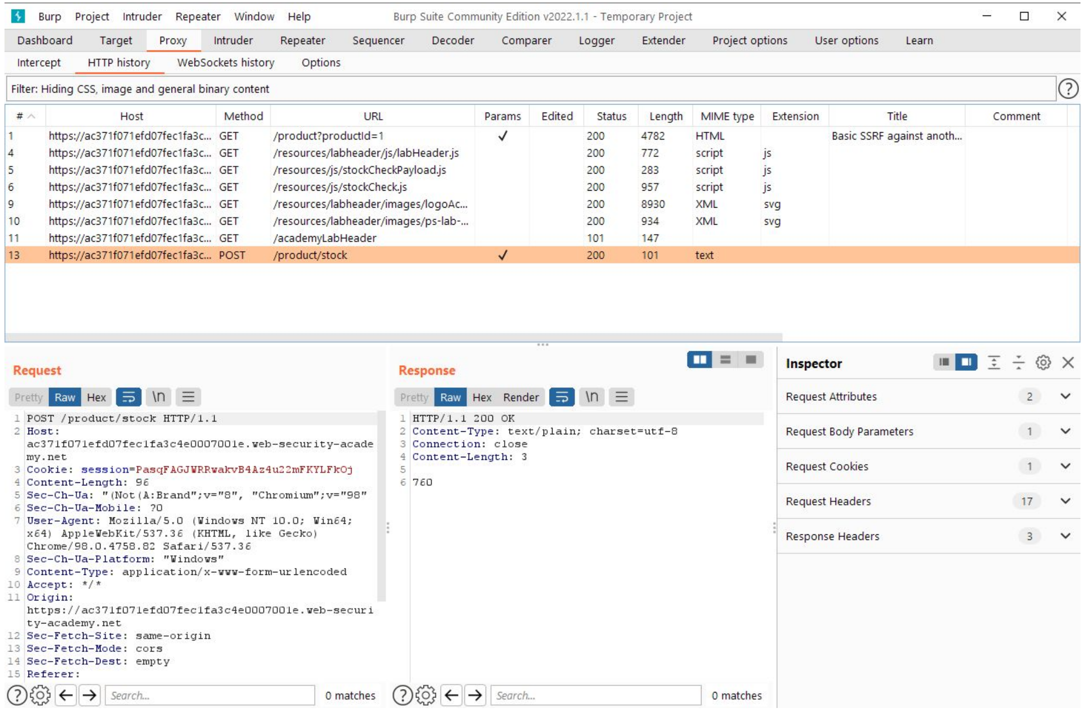
☑ Brute Force Files ☐ Use Blank Extension File extension php

URL to fuzz - /test.html?url={dir}.asp

Please complete the test details

Burp

- General web exploitation tool
- Features:
  - Spidering
  - HTTP request editing (repeater)
  - Brute forcing / fuzzing of HTTP fields (intruder)
- OSS alternative: OWASP ZAP
  - We'll use this later



Burp Suite Community Edition v2022.1.1 - Temporary Project

Filter: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extension	Title	Comment
1	https://ac371f071efd07fec1fa3c...	GET	/product?productId=1	✓		200	4782	HTML		Basic SSRF against anoth...	
4	https://ac371f071efd07fec1fa3c...	GET	/resources/labheader/js/labHeader.js			200	772	script	js		
5	https://ac371f071efd07fec1fa3c...	GET	/resources/js/stockCheckPayload.js			200	283	script	js		
6	https://ac371f071efd07fec1fa3c...	GET	/resources/js/stockCheck.js			200	957	script	js		
9	https://ac371f071efd07fec1fa3c...	GET	/resources/labheader/images/logoAc...			200	8930	XML	svg		
10	https://ac371f071efd07fec1fa3c...	GET	/resources/labheader/images/ps-lab-...			200	934	XML	svg		
11	https://ac371f071efd07fec1fa3c...	GET	/academyLabHeader			101	147				
13	https://ac371f071efd07fec1fa3c...	POST	/product/stock	✓		200	101	text			

Request

```
POST /product/stock HTTP/1.1
Host:
ac371f071efd07fec1fa3c4e0007001e.web-security-acade
my.net
Cookie: session=PasqFAGJWRRwakvB4Az4u22mFKYLFkOj
Content-Length: 96
Sec-Ch-Ua: "(Not(A:Brand";v="8", "Chromium";v="98"
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/98.0.4758.82 Safari/537.36
Sec-Ch-Ua-Platform: "Windows"
Content-Type: application/x-www-form-urlencoded
Accept: */*
Origin:
https://ac371f071efd07fec1fa3c4e0007001e.web-securi
ty-academy.net
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer:
```

Response

```
HTTP/1.1 200 OK
Content-Type: text/plain; charset=utf-8
Connection: close
Content-Length: 3

760
```

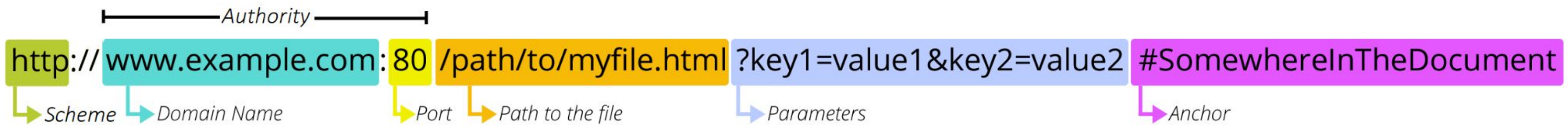
Inspector

Request Attributes	2
Request Body Parameters	1
Request Cookies	1
Request Headers	17
Response Headers	3

Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options Learn
Intercept HTTP history WebSockets history Options
Pretty Raw Hex \n
Pretty Raw Hex Render \n
Search...
0 matches

Directory Traversal

Breakdown of a URL


Authority
http://www.example.com:80/path/to/myfile.html?key1=value1&key2=value2#SomewhereInTheDocument
Scheme
Domain Name
Port
Path to the file
Parameters
Anchor

Let’s say our website has a page:

http://example.com/uploads.php?file=hello.txt

A naive site would do: **file_get_contents(“uploads/” . $_GET[“file”])**

What if we change the url parameter to **../../../../../etc/passwd** ?

It becomes: images/../../../../../etc/passwd $\Rightarrow$ **/etc/passwd**

Cross-Site Scripting (XSS)

Malicious JavaScript injected into a website
Attacker gets a web browser to run JS that it did not intend to
Result of pasting user content directly into the HTML

- **Reflected**
  - **A request to a website** itself contains the script
  - Not persistent
  - Usually a link sent in an email or posted on social media
  - E.g. /search.php?query=<script>alert(1)</script>
- **Stored**
  - **User-submitted content** stored on the website contains the script
  - Persistent
  - Can be the contents of a blog post, comment, social media post, etc.
  - E.g. Comment field on blog that supports HTML syntax

Prevention: sanitize user-generated inputs BEFORE saving to database (stored) or displaying on page (reflected)

Cross-Site Request Forgery (CSRF / XSRF)

Attacker causes victim to send an unintended request to another website

- E.g. POST to https://*bank.com*/transfer?from=alice&to=oscar&amount=1000
- Alice clicks on a link to *evil.com*
- If Alice is signed in to *bank.com*, then the *evil.com* page can automatically send a request to that URL and cause a transfer of funds!

Prevention:

- CSRF tokens:
  - A nonce is required in each POST request to be valid
  - Can only get a nonce from the legitimate *bank.com* site
- SameSite cookie:
  - Set the *bank.com* session cookie with SameSite=Strict and the cookie will not be sent unless the request comes from *bank.com*

SQL Injection (SQLi)

Execution of unintended SQL queries

- Occurs when user input isn't sanitized / parameterized
- "Breaking out" of a pre-existing SQL query
- Result of SQL being executed from another language (PHP, Python, etc.), queries are built from strings
- Plenty of premade payload lists for testing

SQL Primer:

- Statements: SELECT, INSERT, UPDATE, DELETE
- Conditions: start with WHERE
  - Booleans: AND, OR, NOT
  - Comparisons: =, <>, <, <=, >, >=
- Comments: --

Blind SQL injection:

- Page has a SQLi vulnerability but doesn't display SQL output
- Use errors, time delays, etc. to get information out

SQL Injection (SQLi) cont.

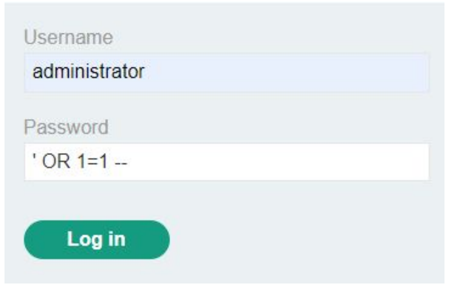
Any user input will go here

Login page example:

```
run(“SELECT * FROM users WHERE user = '$username' AND pass = '$password'”)
```

What if we enter:

- User: administrator
- Pass: ' OR 1=1 --



administrator

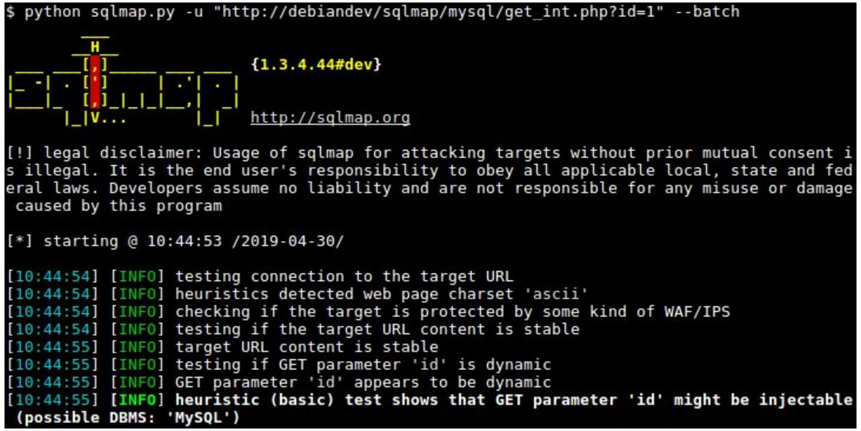
' OR 1=1 --

```
run(“SELECT * FROM users WHERE user = 'administrator' AND pass = '' OR 1=1 --'”)
```

Success! We’re logged in!

Sqlmap

- Automated SQL injection
- Supply it a parameter you want to test
- Great for blind SQL injection or where you’re not sure if something is vulnerable
- If successful, it can:
    - Enumerate users, databases, tables, etc.
    - Dump databases
    - Download/upload files
    - Execute arbitrary commands (SQL and shell commands)



```
$ python sqlmap.py -u "http://debiandev/sqlmap/mysql/get_int.php?id=1" --batch
        ___
       __H__
 ___ ___[,]_____ ___ ___  {1.3.4.44#dev}
|_ -| . [']     | .'| . |
|___|_  [,]_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent i
s illegal. It is the end user's responsibility to obey all applicable local, state and fed
eral laws. Developers assume no liability and are not responsible for any misuse or damage
 caused by this program

[*] starting @ 10:44:53 /2019-04-30/

[10:44:54] [INFO] testing connection to the target URL
[10:44:54] [INFO] heuristics detected web page charset 'ascii'
[10:44:54] [INFO] checking if the target is protected by some kind of WAF/IPS
[10:44:54] [INFO] testing if the target URL content is stable
[10:44:55] [INFO] target URL content is stable
[10:44:55] [INFO] testing if GET parameter 'id' is dynamic
[10:44:55] [INFO] GET parameter 'id' appears to be dynamic
[10:44:55] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable
 (possible DBMS: 'MySQL')
```

NoSQL Injection

SQL injection for NoSQL databases (e.g. MongoDB)

- Unlike SQL, queries aren't built from strings
- Specific for each DBMS
- Tricking the backend language (e.g. PHP) into thinking parameter is something other than a string

MongoDB example (PHP):

```
$collection->find(array(
'user' => $_GET['user'],
'password' => $_GET['password']
));
```

Like in SQLi, how do we specify conditions? https://docs.mongodb.com/manual/reference/operator/query/ne/

PHP to the rescue: send `user=admin&password[$ne]=foo` , translates the password parameter to an array (arrays in PHP can have keys like hashmaps or dictionaries in other languages)

Command Injection

Getting a web server to run user-specified system commands

- Usually a result of a server running some command-line tool
- E.g. Down detector service that pings a host from the command line:

```
<?php
$output = shell_exec(“ping -c 4 ” . $_GET[“ip”]);
echo "<pre>$output</pre>";
?>
```

We can change this to anything… what about 8.8.8.8; cat /etc/passwd ?

Prevention:

- Ok: Escaping shell characters before
- Better: Parameterizing arguments
- Best: Don’t run shell commands with any user input

Server-Side Request Forgery (SSRF)

Getting a web server to access an internal URL or a URL that the server is pre-authorized for

- E.g. Exploiting a web server to return a page from the corporate intranet
- Limitations: no custom headers, HTTP verb can't be changed
- Remember that directory traversal example? That also allows for SSRF.

Capital One data breach:

- Server had an SSRF vulnerability
- Attacker sent a request to the AWS EC2 metadata service (would look something like: http://169.254.169.254/latest/meta-data/iam/security-credentials/ISRM-WAF-Role)
- Used the returned access key to obtain data from S3

XML External Entity (XXE) Injection

Using custom XML entities to read local files (local file inclusion) or perform SSRF

- Less relevant now, most websites and APIs use JSON
- Example payload:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
  <!ELEMENT foo ANY >
  <!ENTITY xxe SYSTEM "file:///etc/passwd" >]>
<foo>&xxe;</foo>
```

- Prevention
  - Make sure your XML parsing library doesn’t allow XXE, or disable it
  - Don’t use XML

Your Turn!

Challenges under “Web” category on TCTF (https://tctf.competitivecyber.club)

- Easy start:
  - Hidden In Plain Sight
  - PETAIR
- Medium:
  - BigBusiness
  - Secure Server
  - Calculator-as-a-Service
- Hard(er):
  - Room Finder
  - Ephemeralcoin
  - Juggling

(hint: these last two have vulnerabilities based on the server software they’re running)

https://portswigger.net/web-security/all-labs (need to sign up, it’s free)

Proud Sponsors

CACI

EVER VIGILANT

VOLGENAU SCHOOL

MASON

ENGINEERING

WarCollar

industries