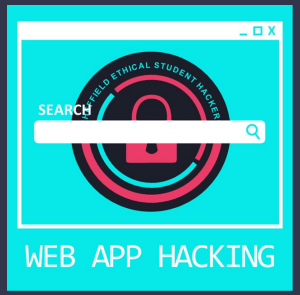
Difficulty: Novice

Ethical Student Hackers

Introduction to Web App Hacking



The Legal Bit

- The skills taught in these sessions allow identification and exploitation of security vulnerabilities in systems. We strive to give you a place to practice legally, and can point you to other places to practice. These skills should not be used on systems where you do not have explicit permission from the owner of the system. It is VERY easy to end up in breach of relevant laws, and we can accept no responsibility for anything you do with the skills learnt here.

- If we have reason to believe that you are utilising these skills against systems where you are not authorised you will be banned from our events, and if necessary the relevant authorities will be alerted.

- Remember, if you have any doubts as to if something is legal or authorised, just don't do it until you are able to confirm you are allowed to.

Code of Conduct

- Before proceeding past this point you must read and agree to our Code of Conduct - this is a requirement from the University for us to operate as a society.
- If you have any doubts or need anything clarified, please ask a member of the committee.
- Breaching the Code of Conduct = immediate ejection and further consequences.
- Code of Conduct can be found at https://shefesh.com/downloads/SESH%20Code%20of%20Conduct.pdf

Overview

1. Methodologies
2. Attacks
    a. SQLi
    b. XXS
    c. LFI
3. Tools
    a. GoBuster
    b. WFuzz
    c. Curl
    d. Burp Suite
4. Challenge time!

These slides are available at shefesh.com/sessions if you want to follow along!

Methodologies

What are we trying to achieve?

- Read sensitive files
- Find hidden pages
- Access what you're not supposed to
- **Ultimate goal - Code execution!**

What do we need to look for?

- User input
- Web app technologies
- Dependency vulnerabilities
- **Anything you can exploit!**

SQLi

SQL - Structured query language

Used to retrieve or modify data in databases

SELECT **INSERT INTO** **DELETE**

UNION **UPDATE**

SELECT [fields] FROM [table] (WHERE [condition]);

SELECT * FROM users WHERE admin = true;

SQLi

SQL Injection - Exploitation of SQL queries with unsanitized user input

In-band SQLi

- Attacker is able to use the same communication channel to both launch the attack and gather results

Inferential SQLi

- attacker is able to reconstruct the database structure by sending payloads, observing the web application's response and the resulting behavior of the database server

Out-of-band SQLi

- an attacker is unable to use the same channel to launch the attack and gather results

SQLi

Bypassing a login form

- A login query may look like this:
  - SELECT * FROM users WHERE username = ‘$username’ AND password = ‘$password’;
- Our attack payload looks like this:
  - In the username field we enter the following: ’ OR 1=1;--
  - The query now looks like this: SELECT * FROM users WHERE username = ‘’ OR 1=1;--AND password = ‘$password’;

SQLi

Data exfiltration

- A search query may look like this:
  - ”SELECT * FROM products WHERE name LIKE ‘%” + user_input + “%’;”
- Our attack payload looks like this:
  - In the search field we enter the following: %’ UNION SELECT * FROM users;--
  - The query now looks like this: SELECT * FROM products WHERE name LIKE ‘%%’ UNION SELECT * FROM users;--%’;

Protecting from SQLi

- Concatenation, input is executed as code

```
$query = ”SELECT * FROM products WHERE name = $user_input;”;
$result = mysql_query($conn, $query);
```

- Sanitised data using parameterization, input is executed as text

```
$db = connect_db();
$stmt = $db->prepare(”SELECT * FROM products WHERE name = ?;”);
$stmt->bind_param('s', $user_input);
$stmt->execute();
```

Example in PHP with MySQL

XSS

Cross Site Scripting (XSS) - Sending of malicious code to websites via unsanitized user input

- DOM - an element in the Document Object Model is changed by a feature on the page - e.g. a button
- Reflected - the payload is delivered in the URL and then rendered on the page - e.g. a search bar
- Stored - the payload is saved to a persistent storage location and later rendered - for example, a commenting system

Self retweeting XSS Attack in Tweetdeck

XSS

DOM XSS

Select your language:
<select><script>
document.write("<OPTION
value=1>"+decodeURIComponent(do
cument.location.href.substring(docu
ment.location.href.indexOf("default="
)+8))+"</OPTION>");
document.write("<OPTION
value=2>English</OPTION>");
</script></select>

Invoked with
http://www.some.site/page.html?def
ault=French

XSS Attack
http://www.some.site/page.html?def
ault=<script>alert(document.cookie)<
/script>

XSS

Reflected XSS

<% String eid =

request.getParameter("eid"); %>

Employee ID: <%= eid %>

Display employee id entered into

HTTP request

Usually used in phishing

Send via phishing

http://www.some.site/page.html?eid

=<script>alert(document.cookie)</sc

ript>

XSS

Stored XSS

$sql = "INSERT INTO MyGuests

(firstname, lastname, email)

VALUES ($_GET['firstname'],

$_GET['lastname'], $_GET['email'])";

Enter guest into database

<?php echo("<p>" . $email . "</p>"); ?>

Invoked with

http://www.some.site/add_guest?firstname=John&lastname=Doe&email=test@test.com

XSS Attack

http://www.some.site/add_guest?firstname=John&lastname=Doe&email=<script>alert(document.cookie)</script>

XSS

Preventing XSS

DOM based XSS - HTML encoding and JavaScript encode all untrusted input

https://cheatsheetseries.owasp.org/cheatsheets/DOM_based_XSS_Prevention_Cheat_Sheet.html#guideline

Reflected & Stored XSS - Deny all untrusted data where possible, HTML encode, attribute encode, JavaScript encode...Encode as much as possible!

https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html#xss-prevention-rules-summary

LFI

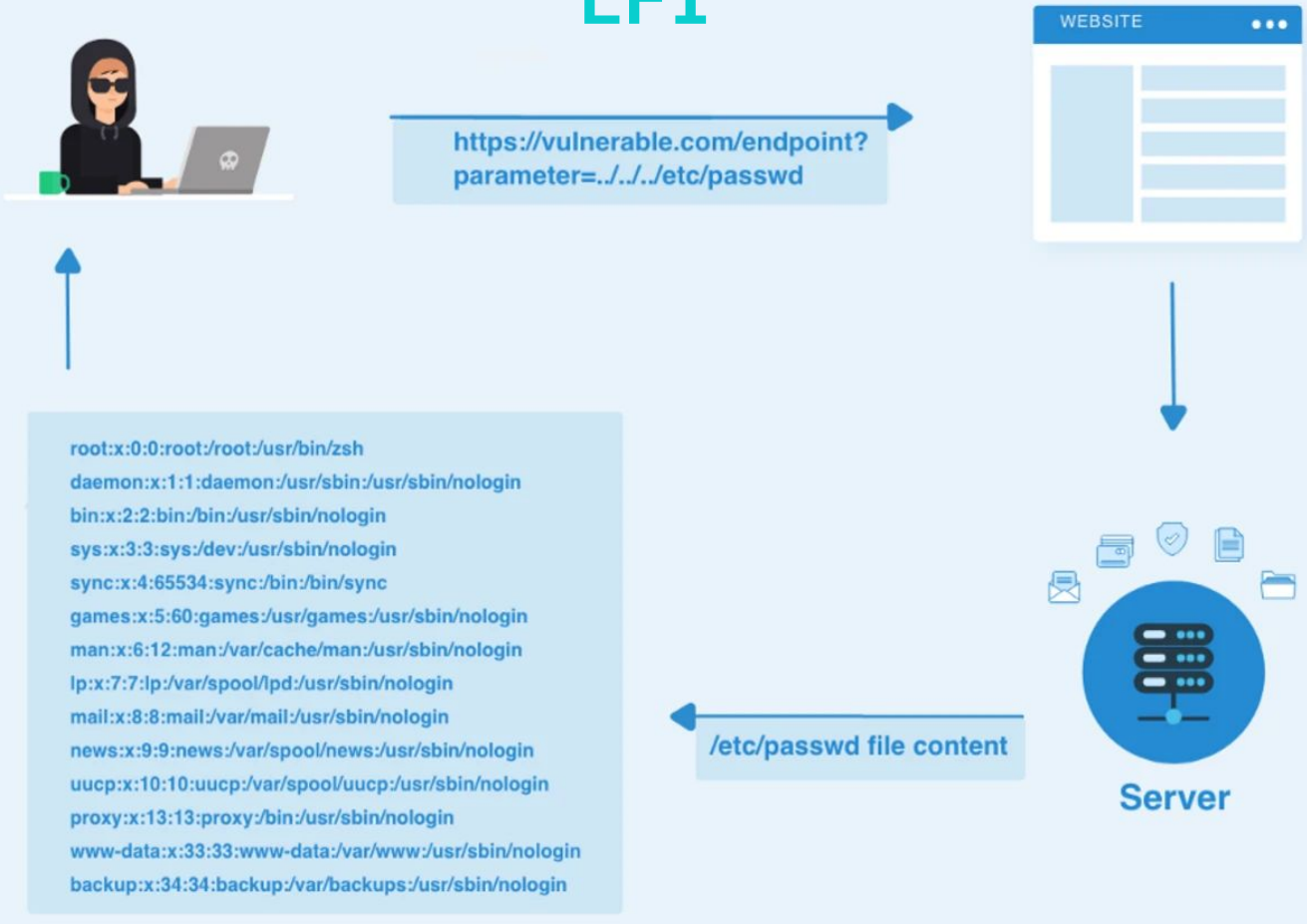
Local File Inclusion (LFI) - A trick to cause a webpage to expose or run local files on the web-server via the use of unsanitized user input

Usually PHP scripts

(Directory traversal if not PHP)

```php
<?php
  $file = $_GET['file'];
  if(isset($file)) {
    include("pages/$file");
  }
  else {
    include("index.php");
  }
?>
```

LFI



https://cobalt.io/blog/a-pentesters-guide-to-file-inclusion

LFI

Preventing LFI

- Avoid passing user-submitted input to any filesystem
- White list of files
- Use IDs instead of file paths (reject invalid IDs)
- Do not accept characters such as .. or / or %00 etc.
- PHP function to read from directory

GoBuster

Brute force URIs, DNS subdomains, virtual host names, open AWS S3 buckets

Why?

- Find hidden urls
- Find hidden subdomains
- Much quicker than doing it manually
- Use common wordlists

Alternative tools: DirBuster, FeroxBuster

GoBuster

- -P password
- -U username
- -c cookies
- -m mode (dir or dns)
- -r follow redirects
- -u target URL or domain
- -v verbose
- -w path to wordlist

gobuster -u http://192.168.0.155/ -w /usr/share/wordlists/dirb/common.txt -q -n -e

Search URL using wordlist common.txt, don't print banner, don't print status codes and use expand mode to print full URLs

Gobuster Examples

WFuzz

Brute force HTTP request fields

- Parameters
- Authentication
- Forms
- Headers

WFuzz

Fuzzing URL Parameters

wfuzz -z range,0-10 --hl 97

http://testphp.vulnweb.com/listproducts.php?cat=FUZZ

Test URL with values 0-10 for cat parameter.
Hide responses with 97 lines

Fuzzing POST Requests

wfuzz -z file,wordlist/others/common_pass.txt -d "uname=FUZZ&pass=FUZZ" --hc 302 http://testphp.vulnweb.com/userinfo.php

Test URL with wordlist data for uname and pass parameters. Hide 302 responses

Tools

Curl

Command line tool for HTTP requests

curl [PROTOCOL]://[URL]:[PORT]

curl [PROTOCOL]://[URL]:[PORT] -d [DATA]

https://shefesh.com/wiki/fundamental-skills/tools-1---curl.pdf

Burp Suite

Tool for inspecting HTTP traffic, proxy between browser and server

Default HTTP proxy - 127.0.0.1:8080

- Enable a proxy in your browser
- Requests show in Burp Suite
- Request held until forwarded or intercept is disabled

https://shefesh.com/wiki/fundamental-skills/web-3---burp-suite.pdf

http://44.192.5.204/DVWA/

sesh : seshdemo!

Join the Committee

EGM **11th October 2021**

Please contact ethicalhackers@sheffield.ac.uk if you are interested!

Positions available...

Publicity Officer

Manage social media accounts for the society and create advertising materials.

General Member

Focus on helping other committee members and contributing to creating content to educate our members.

Upcoming Sessions

What's up next?

www.shefesh.com/sessions

Automation in Cybersecurity + EGM: **11/10/21** 19:00 - 20:30 Arts Tower LT01

Yorkshire & Humber Regional Organised Crime Unit (Guest Talk) **18/10/21** 19:00 - 20:30 Location Hicks LT05

Operating System Security **25/10/21** 19:00 - 20:30 Location TBC

Reconnaissance **01/11/21** 19:00 - 20:30 Location TBC

Any Questions?

www.shefesh.com

Thanks for coming!