Exploitation

File Upload Vulns

- Simples type of vulnerabilities.
- Allow users to upload executable files such as php.

Upload a php shell or backdoor, ex: weevly

1. Generate backdoor > weevly generate [passord] [file name]
2. Upload generated file.
3. Connect to it > weevly [url to file] [password]
4. Find out how to use weevly > help

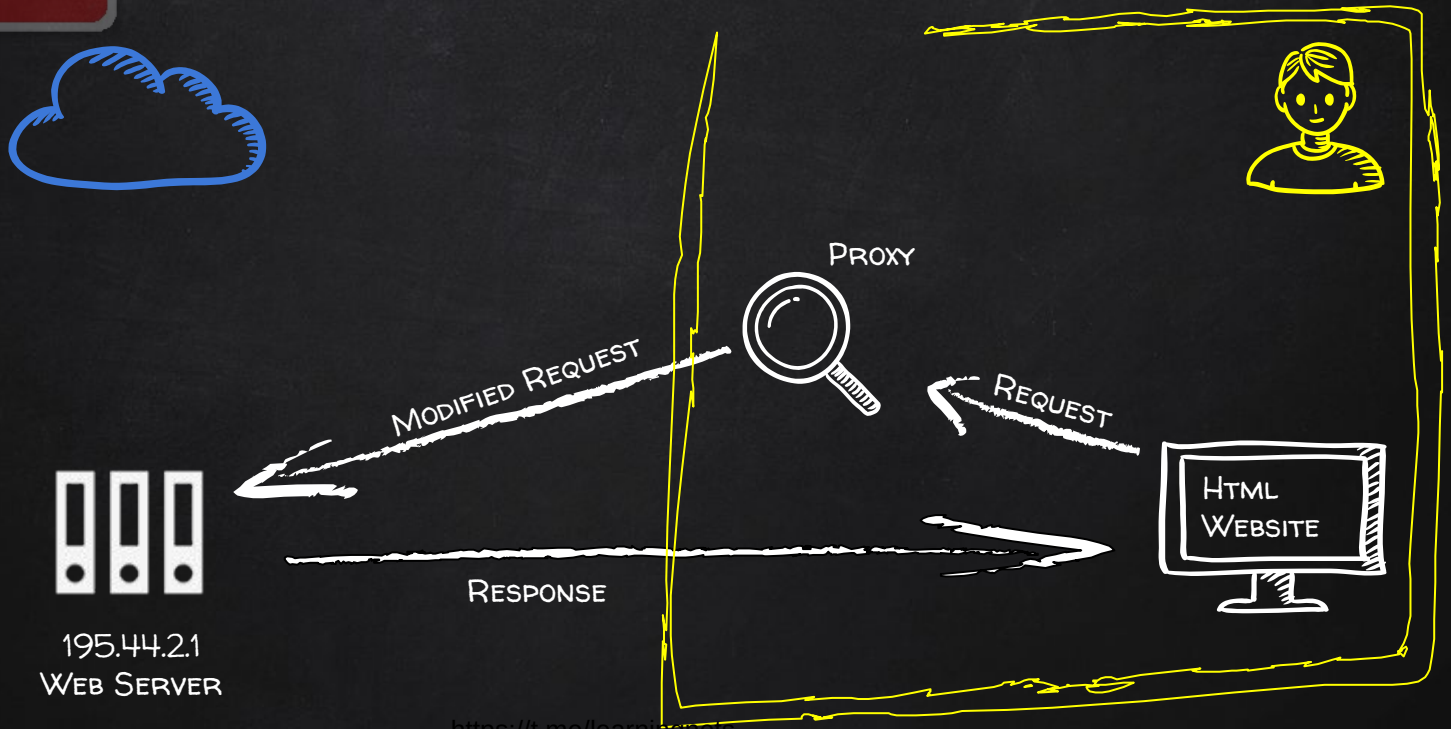
HTTP Requests

Basic information Flow

- User clicks on a link.
- HTML website generates a request (client side)
- Request is sent to the server.
- Server performs the request (Server Side)
- Sends response back.

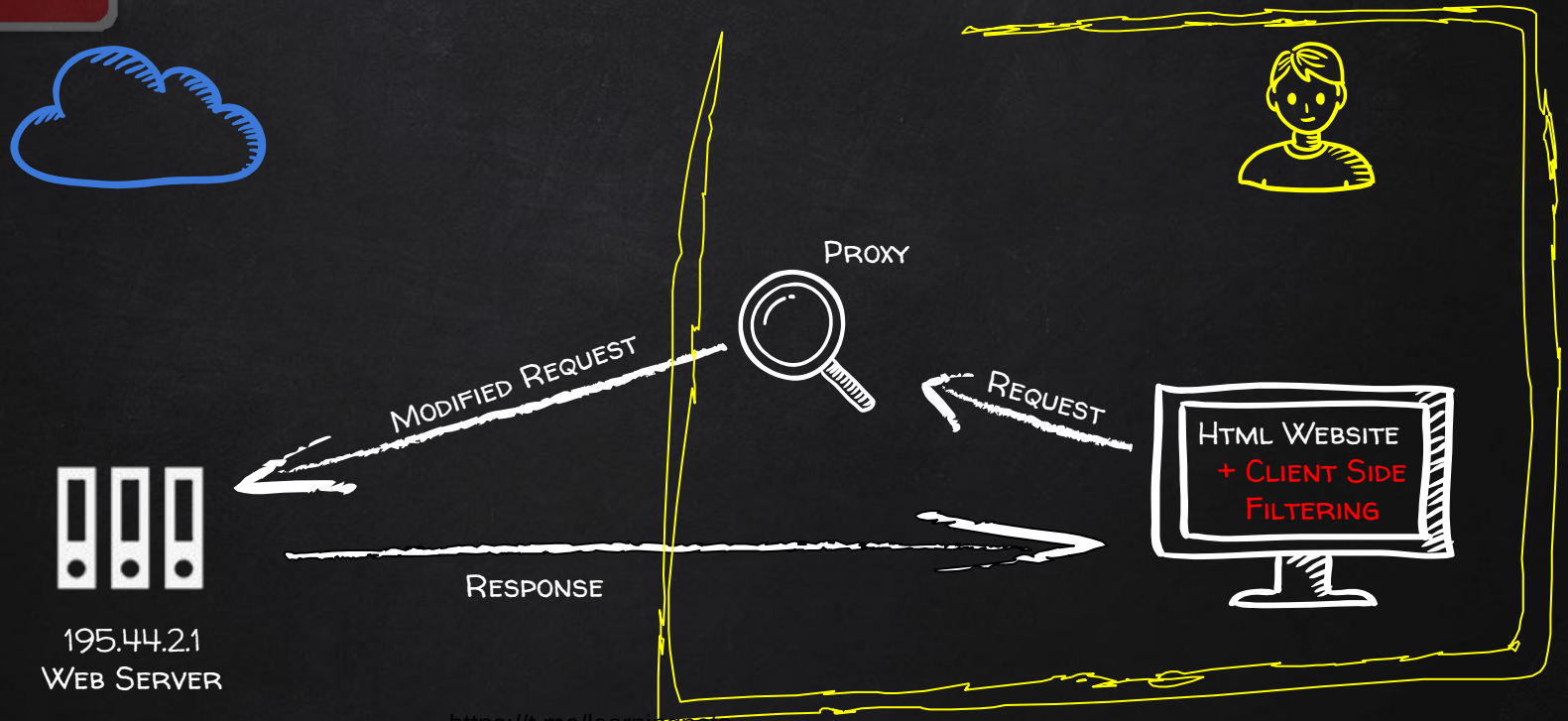
Intercepting Requests

Burp Proxy



Intercepting Requests

Burp Proxy



Mitigation

File Upload Vulns

1. Never allow users to upload executables (php, exe ...etc)
2. Check the file type AND the file extension.
3. Analyse the uploaded file itself, recreate it and rename it.

Exploitation

Code Execution Vulns

- Allows an attacker to execute OS commands.
- Windows or linux commands.
- Can be used to get a reverse shell.
- Or upload any file using wget command.
- Code execution commands attached in the resources.

Mitigation

Code Execution Vulns

1. Don't use dangerous functions.
2. Filter use input before execution.

Exploitation

Local File Inclusion

- Allows an attacker read ANY file on the same server.
- Access files outside www directory.

Exploitation

Shell from Local File Inclusion

- Try to inject code into readable files.
- Ex:
  - */proc/self/environ*
  - */var/log/auth.log*
  - */var/log/apache2/access.log*

Exploitation

Remote File Inclusion

- Similar to local file inclusion.
- But allows an attacker read ANY file from ANY server.
- Execute php files from other servers on the current server.
- Store php files on other servers as .txt

Mitigation

File Inclusion Vulns

1. Prevent remove file inclusion
   > Disable allow_url_fopen & allow_url_include.

2. Prevent local file inclusion
   > Use static file inclusion.

Exploitation - SQL Injection

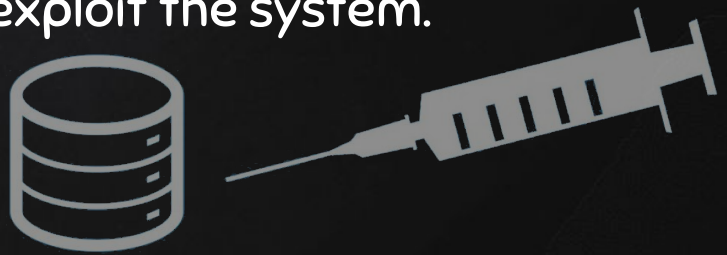
What SQL ?

- Most websites use a database to store data.
- Most data stored in it (usernames, passwords ..etc)
- Web application reads, updates and inserts data in the database.
- Interaction with DB done using SQL.

Exploitation - SQL Injection

Why are they so dangerous

1. They are everywhere.
2. Give access to the database → sensitive data.
3. Can be used to read local files outside www root.
4. Can be used to log in as admin and further exploit the system.
5. Can be used to upload files.



Exploitation - SQL Injection

Discovering SQLi

- Try to break the page.
- Using 'and', 'order by' or " ' ".
- Test text boxes and url parameters on the form http://target.com/page.php?something=something

Exploitation - SQL Injection

SQLmap

- Tool designed to exploit sql injections.
- Works with many db types, mysql, mssql ...etc.
- Can be used to perform everything we learned and more!

```
> sqlmap --help
> sqlmap -u [target url]
```

Preventing SQLi

- Filters can be bypassed.
- Use black list of commands? Still can be bypassed
- Use whitelist? Same issue

→ Use parameterized statements, separate data from sql code.

Exploitation - XSS Vulns

XSS - Cross Site Scripting vulns

- Allow an attacker to inject javascript code into the page.
- Code is executed when the page loads.
- Code is executed on the client machine not the server.

Three main types:

1. Persistent/Stored XSS
2. Reflected XSS
3. DOM based XSS

Exploitation - XSS Vulns

Discovering XSS

- Try to inject javasript code into the pages.
- Test text boxes and url parameters on the form http://target.com/page.php?something=something

Exploitation – XSS Vulns

Reflected XSS

- None persistent, not stored.
- Only work if the target visits a specially crafted URL
- EX
  http://target.com/page.php?something=<script>alert("XSS")</script>

Exploitation - XSS Vulns

Stored XSS

- Persistent, stored on the page or DB.
- The injected code is executed everytime the page is loaded.

Exploitation - XSS Vulns

Dom Based XSS

- Similar to reflected and stored XSS.
- Can be discovered and exploited similarly.
- Main difference is that it occurs entirely on the client side.
- Payload is never sent to the server.
  -> No logs, no filters, no server side protection

Exploitation - XSS Vulns

Exploiting XSS - Beef Framework

- Run any javascript code.
- Targets can be hooked to beef using javascript code.
- Browser Exploitation Framework allowing us to launch a number of attacks on a hooked target.

-> Inject Beef hook in vulnerable pages.
-> Execute commands from beef.

Preventing XSS Vulns

- Minimize the usage of user input on html.
- Escape any untrusted input before inserting it into the page.

Char	Result
&	→ &
<	→ <
>	→ >
"	→ "
'	→ '
/	→ /

→https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet

Brute Force & Dictionary Attacks

1. Brute Force Attacks

Cover all possible combinations.

2. Dictionary Attacks

Use a wordlist, try every password in the list only.

Creating a Wordlist

Crunch can be used to create a wordlist.

Syntax:

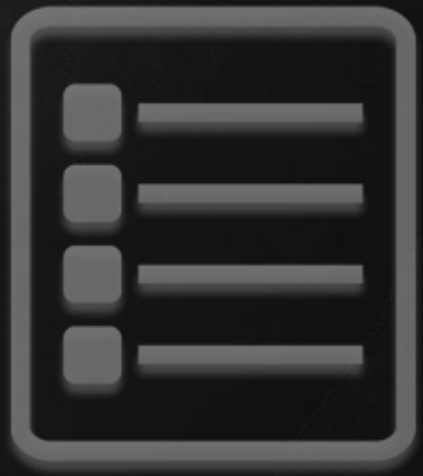
> crunch [min] [max] [characters] -t [pattern] -o [FileName]

Example:

> crunch 6 8 123abc$ -i wordlist -t a@@@@b

Generated passes:

aaaaab
aabbbb
aan$$b
.....



Hydra

Hydra is a bruteforce tool that can be used to bruteforce almost any authentication service.

Syntax:

```
> hydra [IP] -L [usernames] -P [passwords] [service]
```

Example:

```
> hydra 10.20.14.212 -l admin -P /root/wordlist.txt http-post-form
"/mutillidae/?page=login.php:username=^USER^&password=^PASS^&lo
gin-php-submit-button=Login:F=Not Logged In"
```