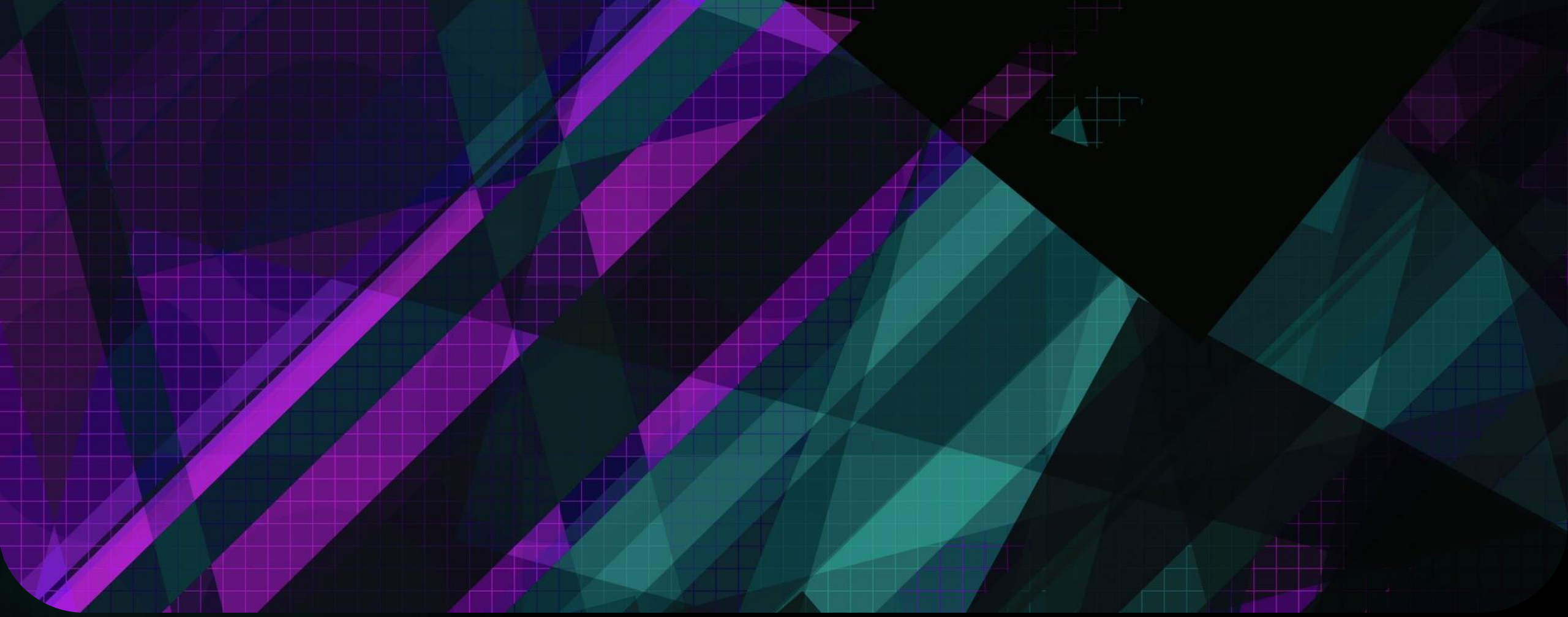


Introduction web vulnerability

WEB APPLICATION & API TESTING

Agenda

- Introduction
- OWASP
- Authentication Bypass
- IDOR
- File Inclusion
- SSFR
- Cross-site Scripting
- Command Injection
- SQL Injection
- Common API security problems

Web application

- Standard architecture webapp
- Intercept the traffic between a web browser/client and web server → (BurpSuite) to understand the communication
- Authentication vs Authorization

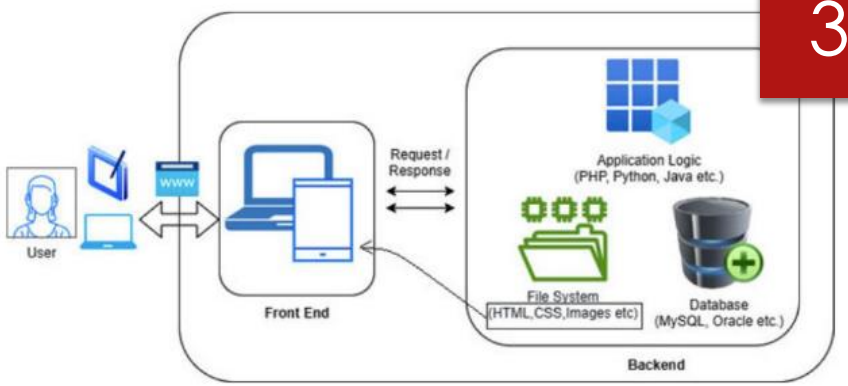


Figure 4.1: Standard web application architecture

Figure 4.2: Raw HTTP request and response

Different types of web application vulnerabilities

- Insecure digest
- Insecure Direct Object References (IDOR)
- Insufficient logging and monitoring
- Insufficient session expiration
- Insufficient transport layer protection
- Lightweight Directory Access Protocol (LDAP) injection
- Missing function level access control
- Operating System (OS) command injection
- Race condition
- Remote Code Execution (RCE)
- Remote File Inclusion (RFI)
- Security misconfiguration
- Sensitive data exposure
- Session ID leakage
- SQL injection
- Invalidated redirects and forwards
- Broken access control
- Broken authentication
- Carriage Return and Line Feed (CRLF) injection
- Cipher transformation insecure
- Components with known vulnerabilities
- Cross-Origin Resource Sharing (CORS) policy
- Credentials management
- Cross-Site Request Forgery (CSRF)
- Cross-Site Scripting (XSS)
- Directory indexing
- Directory traversal
- Error handling
- Failure to restrict URL access
- HTTP response splitting
- Insecure deserialization

OWASP

- https://owasp.org/API-Security/editions/2023/en/0x11-t10/
- https://owasp.org/Top10/
- Brocken Access Control
- Injection
- Insecure design
- Security Misconfiguration
- Identification and Authentification failures
- Server-Side Request Forgery

Injection – SQL Injection

- Appears when developpers don’t sanitize user input
- https://site.com/profile?id=123
- "123; DROP TABLE Users;" → SQL request run 2X
- Very high level :
  - Error-based SQLi
  - Blind SQLi

Exemple Error-based SQLi

- Supposons que l'URL du site Web permette aux utilisateurs de rechercher des articles par ID, comme suit : `https://example.com/articles?id=1`.
- Requête SQL côté serveur :

```
SELECT * FROM Articles WHERE ArticleID = 1;
```

- Un attaquant peut entrer une entrée malveillante comme `1' OR '1'='1`;--`. La requête SQL résultante sera :

```
SELECT * FROM Articles WHERE ArticleID = 1' OR '1'='1';--';
```

- Si l'application est vulnérable à l'Error-based SQL Injection, elle peut renvoyer une erreur de base de données, indiquant une vulnérabilité.

Exemple Blind SQL injection

- Supposons que le site Web ait une fonction de recherche qui renvoie une réponse différente en fonction de la vérité d'une condition, par exemple, si l'intrant est valide ou non. L'URL pourrait ressembler à ceci : `https://example.com/search?query=test`.
- SELECT * FROM Products WHERE Name = 'test';

- Un attaquant peut entrer une entrée malveillante comme `test' AND 1=1;--`. La requête SQL résultante sera :
- SELECT * FROM Products WHERE Name = 'test' AND 1=1;--';
- Si la page répond différemment (par exemple, affiche un certain message) lorsque la condition est vraie, l'attaquant peut déduire la vérité de la condition en fonction des réponses de la page, réalisant ainsi une Blind SQL Injection

Blind SQLi – Authentication bypass

- Little to no feedback
- Matching pair in the user table : username & Password

Blind SQLi

Boolean Based

- Response from our injection : true or false, yes/no , 1/0
- Confirm that injection payload was successful or not
- Possibility to enumarate a whole database structure and contents
- https://tryhackme.com/room/sqlinjectionlm

Boolean Based

- No visual indicator of your queries
- Based on time delay : SLEEP(x) alongside UNION statement.

- Outil : SQLMap

Remediation

As impactful as SQL Injection vulnerabilities are, developers do have a way to protect their web applications from them by following the below advice:

- **Prepared Statements (With Parameterized Queries)**
- **Input Validation**
- **Escaping User Input**
- **Enforcing least privilege**

Outils : SQLMap
Cheatsheet:

https://portswigger.net/web-security/sql-injection/cheat-sheet

Cross Site Scripting (XSS)

- Les attaques XSS injectent des scripts malveillants dans des sites web
- Quand : Application envoie des données non fiables au navigateur sans validation côté serveur.
- XSS: Serveur renvoi du code malveillant à un autre utilisateur.

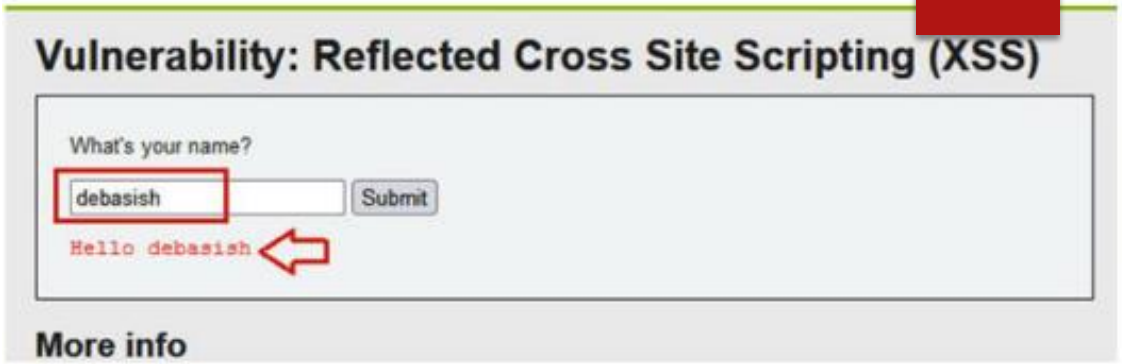


Figure 4.16: DVWA XSS

Figure 4.18: Burp Suite input sanitization check

Types of Cross-site scripting

- Reflected XSS
- Stored XSS
- DOM Based XSS
- Blind XSS

Reflected XSS

- https://tryhackme.com/room/xss

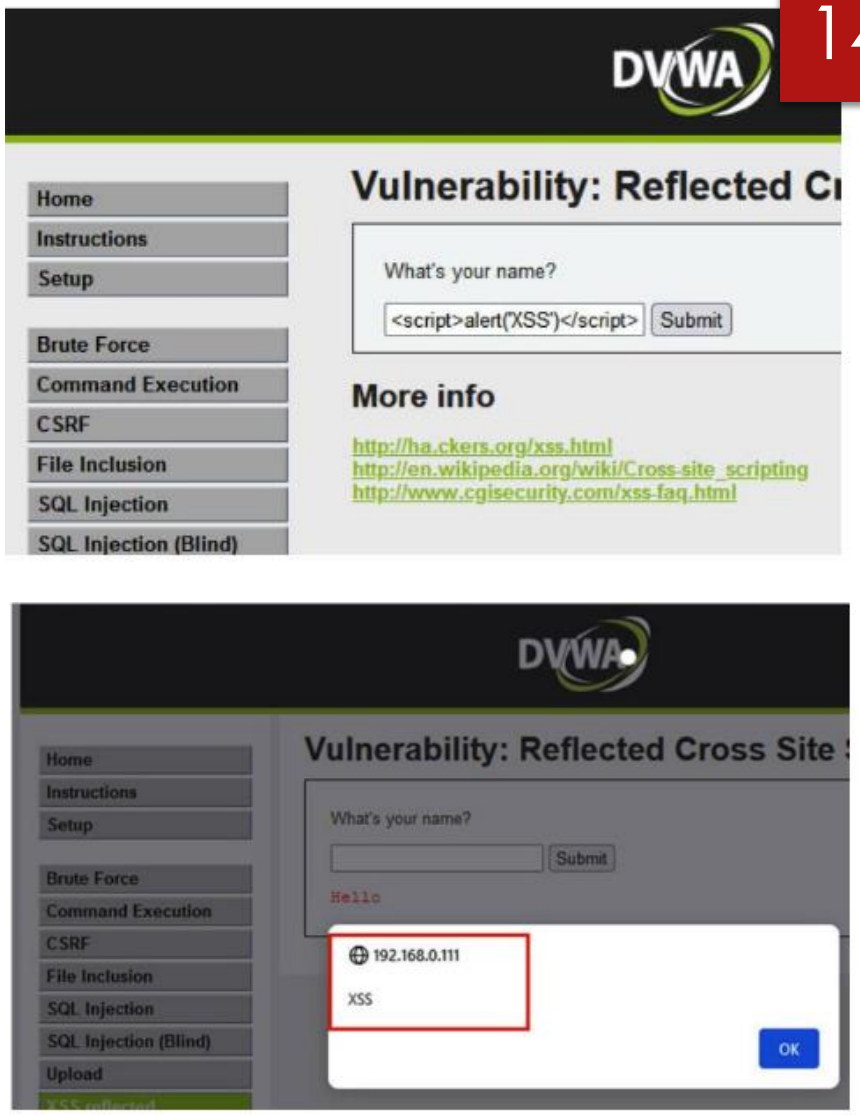
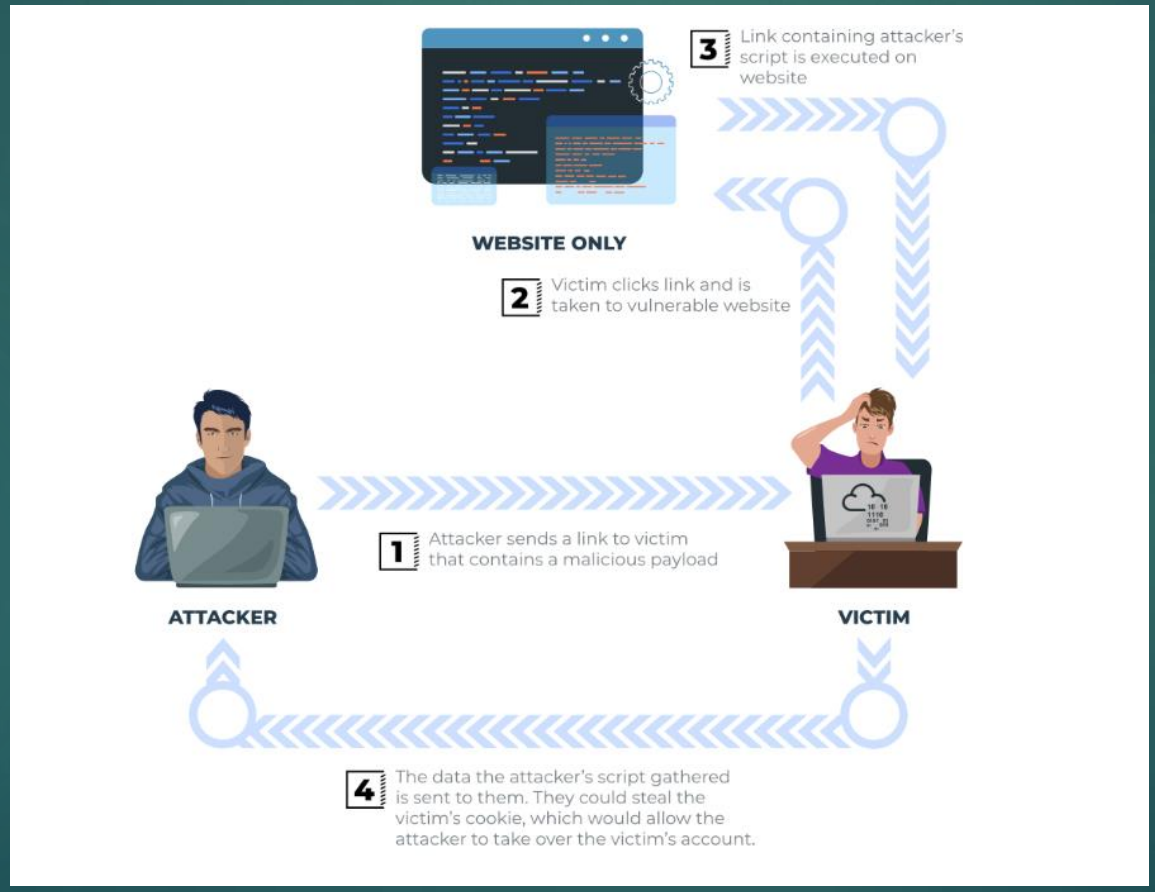


Figure 4.20: DVWA XSS

Reflected XSS



Stored XSS

- XSS Payload is stored on the web application (database for exemple)
- Payload runs when other user visit the site

DOM Based XSS

- DOM
- Exploitation du DOM
- Example Scenario
- Des liens manipulés pourraient être envoyés à des victimes potentielles, les redirigeant vers un autre site web ou volant du contenu de la page ou de la session de l'utilisateur.

Exploitation:

- Lire toutes les données que l'utilisateur peut accéder
- Injecter une fonctionnalité de cheval de Troie dans le site web
- Effectuer un défacement virtuel du site web
- Effectuer toute action que l'utilisateur peut effectuer
- Capturer les identifiants de connexion de l'utilisateur

DOM Based XSS

- Encodage des données en sortie
- Filtrage des données d'entrée
- Utilisation d'en-têtes de réponse appropriée :
- Définition du flag httponly pour les cookies
- Politique de sécurité du contenu

Server-Side Request Forgery (SSRF)

- Exploite des fonctionnalités sur le serveur pour lire ou modifier des ressources internes.
- Permet à l'attaquant de contraindre l'application côté serveur à effectuer des requêtes HTTP vers un domaine spécifique choisi par l'attaquant.

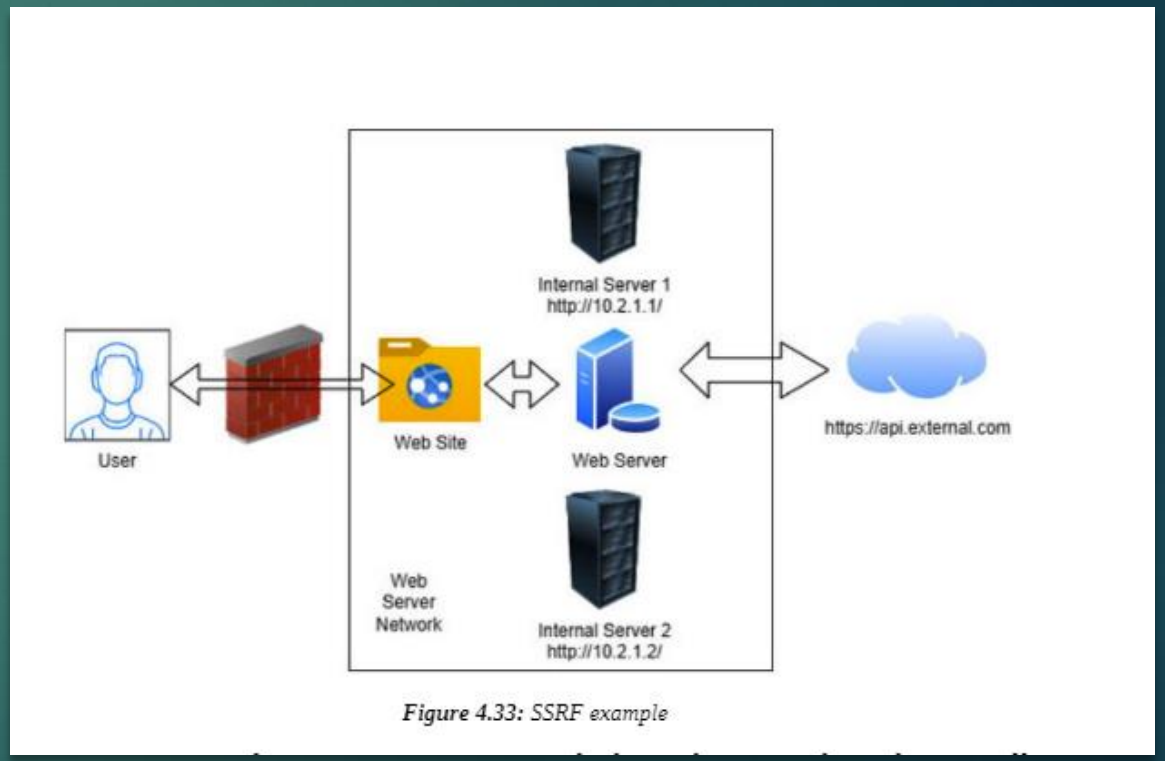


Figure 4.33: SSRF example

SSRF

Exploitations

- Accès à des zones non autorisé
- Accès aux données du client/organisation
- Capacité de s'étendre aux réseaux internes.

Mesures de protection possibles

- Principe de défense en profondeur
- Validation des entrées
- Validation des adresses IP et des noms de domaine

SSRF

Where to look

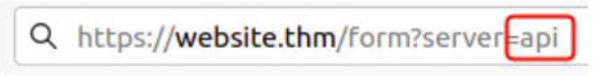
- Quand un URL entier est utilisé dans un paramètre dans la barre d'adresse
- URL partielle comme un nom d'hôte
- Chemin d'accès de l'URL.

When a full URL is used in a parameter in the address bar:

A hidden field in a form:

```
<form method="post" action="/form">
    <input type="hidden" name="server" value="http://server.website.thm/store">
    <div>Your Name:</div>
    <div><input name="client_name"></div>
    <div>Your Email:</div>
    <div><input name="client_email"></div>
    <div>Your Message:</div>
    <div><textarea name="client_message"></textarea></div>
</form>
```

A partial URL such as just the hostname:



Or perhaps only the path of the URL:

IDOR

Qu'est-ce qu'un IDOR

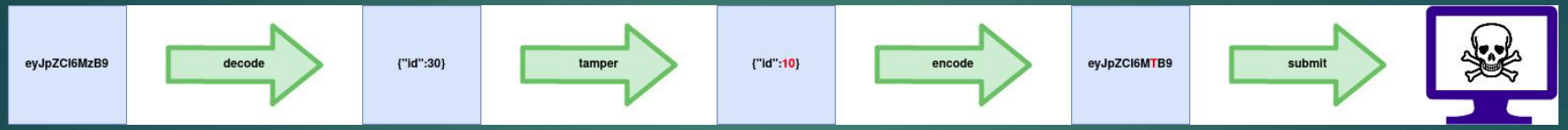
- Signifie «Insecure Direct Object Reference».
- Type de vulnérabilité de contrôle d'accès
- Survient lorsqu'un serveur web reçoit des données fournies par l'utilisateur pour récupérer des objets.
- Confiance excessive, données d'entrées ne sont pas validées pour confirmer que l'objet appartient à l'utilisateur le demandant

Exemple

http://online-service.thm/profile?user_id=1305 → http://online-service.thm/profile?user_id=1000

IDOR

Finding IDOR's in Encoded Ids



Finding IDOR's in Hashed Ids

Finding IDOR's in Unpredictable Ids

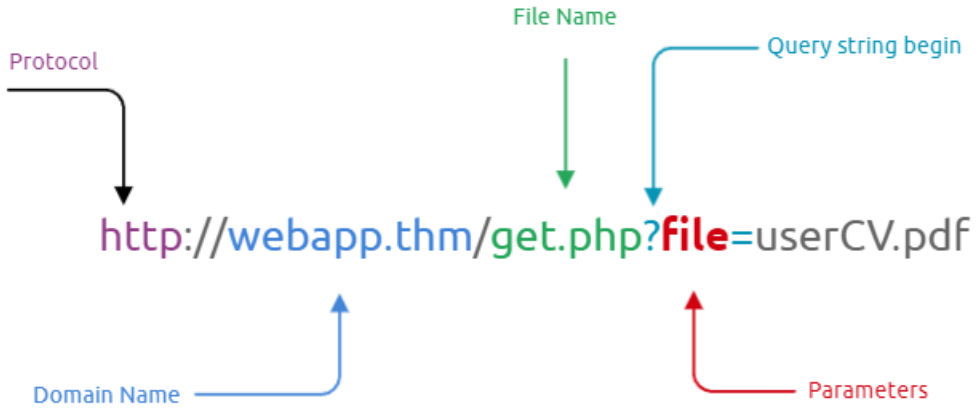
File Inclusion

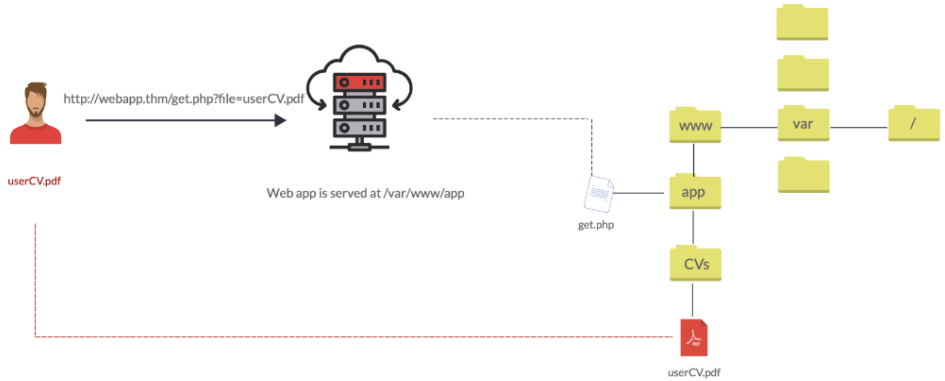
What is File inclusion?

- Pratique qui permet à une application de charger et utiliser des fichiers externes, tels que des scripts dans ses opérations

Risques?

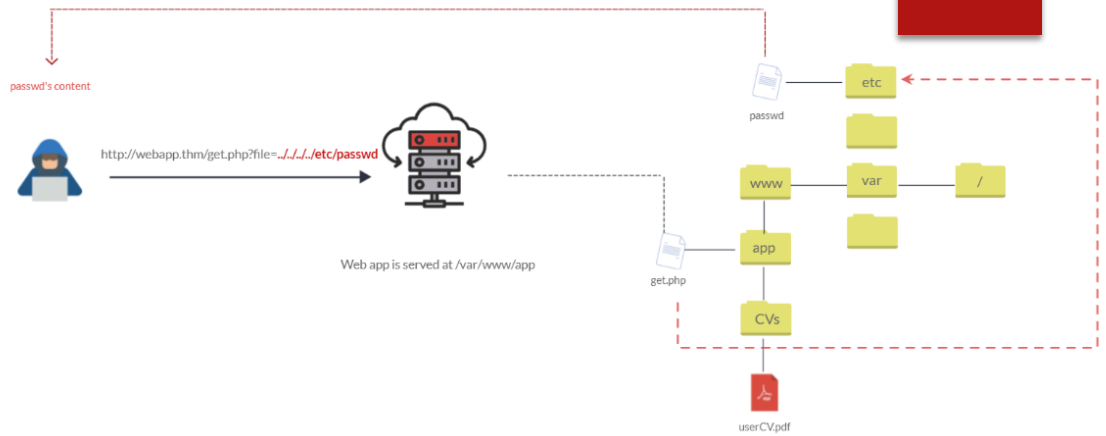
- Accès à des fichiers sensibles, exécuter du code malveillant ou compromettre la sécurité de l'application





File inclusion

Path traversal (directory transversal)



As a result, the web application sends back the file's content to the user.

Local file inclusion (LFI)

- Vulnérabilités LFI souvent liées au manque de sensibilisation à la sécurité chez les développeurs
- functions like include, require, etc., in PHP or similar functions in other languages.
- LFI vulnerabilities occur in various languages like ASP, JSP, Node.js, and in this case, PHP is used for demonstration.
- When the directory is specified in the include function, such as include("languages/" . $_GET['lang'])→http://webapp.thm/index.php?lang=../../../../etc/passwd.

- LFI exploits leverage similar concepts to path traversal, using the include function's flexibility to load external files into the current page context.

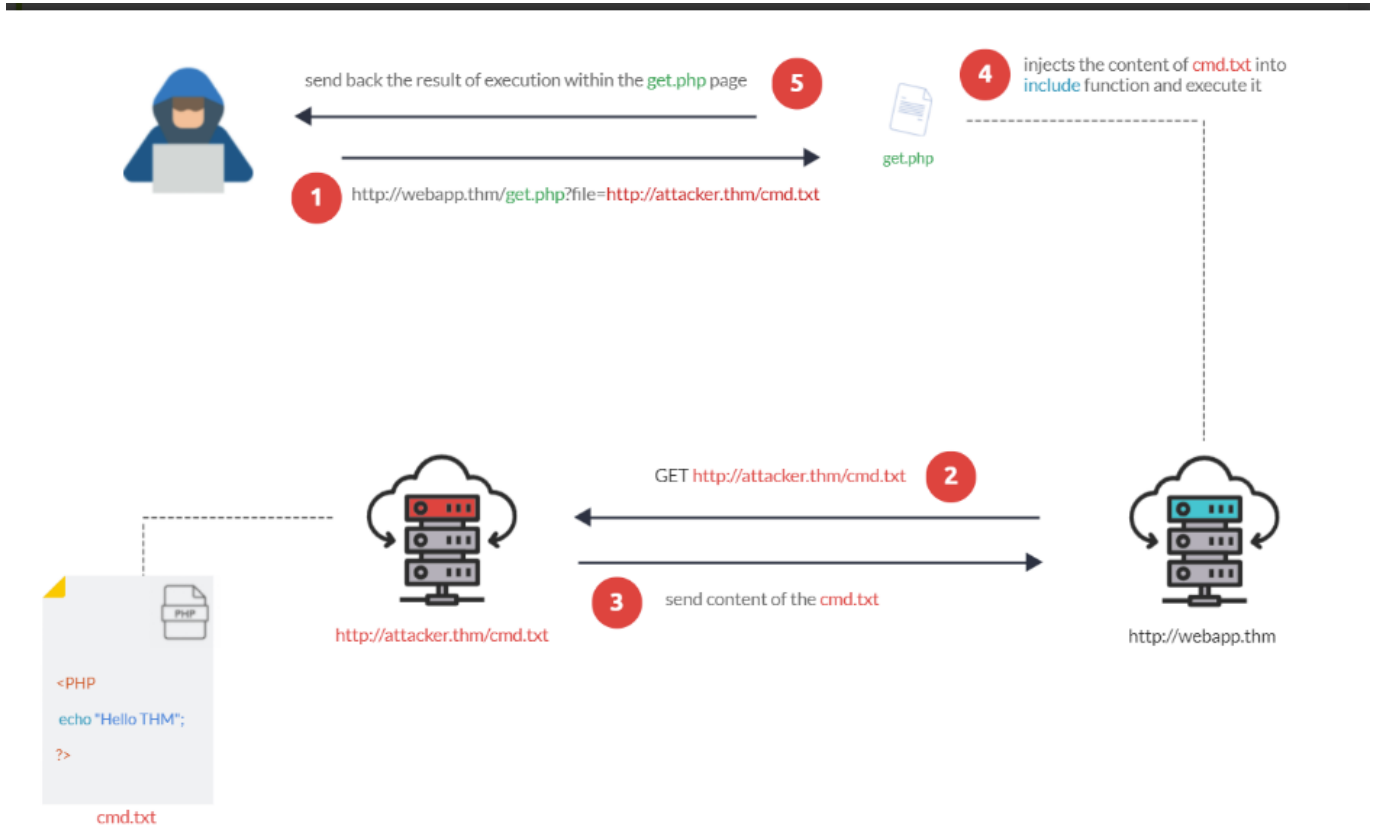
Remote File inclusion

- Remote File Inclusion (RFI) involves including remote files into a vulnerable application by exploiting input sanitization issues, similar to Local File Inclusion (LFI).
- RFI requires the 'allow_url_fopen' option to be enabled, allowing inclusion of remote files via URLs.
- RFI poses a higher risk than LFI as it can lead to Remote Command Execution (RCE) on the server.
- Consequences of successful RFI attacks include Sensitive Information Disclosure, Cross-site Scripting (XSS), and Denial of Service (DoS).
- Attackers host malicious files on their server and inject them into the vulnerable application server through the include function using HTTP requests.
- The content of the injected malicious file executes within the context of the vulnerable application server, allowing the attacker to exploit the system.

RFI - Steps

Remediation

- Latest version of system and services
- Turn off PHP errors
- Web Application Firewall
- Disable some PHP feature (allow url open, allow url include)
- Never trust user input → validation against file inclusion
- White listing for file name and locations as well as blacklisting



Command Injection

What is it ?

- Web vulnerability → exploit application's behavior to execute commands on the operating system
- Attacker can run commands using the privileges of the application
- Allows remote interaction with vulnerable systems

Understanding involves

- Recognizing its impact and risks on applications
- Learning to discover and test for this vulnerability
- Different operating system's payloads
- Implementing preventive measures to secure applications

Most common API security problems

- Return data in JSON or XML
- Rest API
- Soap API
- GraphQL API
- API-Centric Applications

SOAP API

- Request:

DELETE / HTTPS/1.1

Host: example.s3.amazonaws.com

<DeleteBucket xmlns="http://doc.s3.amazonaws.com/2006-03-01"> <Bucket>quotes</Bucket> <AWSAccessKeyId> AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId> <Timestamp>2006-03-01T12:00:00.183Z</Timestamp> <Signature>luyz3d3P0aTou39dzbqaEXAMPLE=</Signature> </DeleteBucket>

- Response:

<DeleteBucketResponse xmlns="http://s3.amazonaws.com/doc/2006-03-01"> <DeleteBucketResponse> <Code>204</Code> <Description>No Content</Description> </DeleteBucketResponse>

</DeleteBucketResponse>

GraphQL API

- Request:

query { shop { name primaryDomain { url host } } }

- Response:

{ "data": { "shop": { "name": "example", "primaryDomain": { "url": "https://example.myshopify.com", "host": "example.myshopify.com" } } }

}

API-Centric Applications

- Composent côté client qui requête et rends des données du server en utilisant des appells API.
- Bcp d'applications moblie sont construites de cette façon : But sauver du temps car pas besoin de réimplémenter les mêmes fonctionnalités.

Vulnérabilités et sécurité

- Broken Access Control and Info Leaks
- Technical Bugs

Broken Access Control and Info leaks

Broken Access Control

- Manque d'autorisation
- Défauts de conception
- Impacts potentiels

Information Leakage

- Messages d'erreur inappropriés
- Données sensibles
- Conséquences

Testing for Technical Bugs

- Server-Side Request Forgery(SSRF) :
  API peut accéder à des URL externes ou interne , permettant à un attaquant de déclencher des requêtes vers des ressources arbitraires, ou internes.
- Path Traversal
  Permet à un attaquant de naviguer au-delà du répertoire prévu
- Insecure File Inclusion
  Exécution de code malveillant en insérant des fichiers externes dans les requêtes
- Insecure Deserialization
  Modifier le flux de données ou d'exécuter du code malveilant lorsque désérialisation des données n'est pas sécurisé
- XML External Entity(XXE)
  Risque: Exploitation d'un fichier XML analyseur pour accéder à des ressources externes.
- Cross-Site Scripting(XSS)
  Impact: Failles XSS permettent d'injecter du code malveillant dans pages web consultées par d'autres utilisateurs

Bibliographie

- Tryhackme : Jr Penetration Tester https://tryhackme.com/
- Bug Bounty Bootcamp : Vickie Li
- **Penetration Testing for Jobseekers** https://univ.scholarvox.com/catalog/book/docid/88937955?searchterm=Penetration%20Testing%20for%20Jobseekers