

RAPPORT

Projet 6 : Pentest d'une application web interne

Slide 1 : Titre

Présentation du projet

- Nom du projet : pentest
- Réalisé par : FATIMA LY & AMINATA NIANG
- Date : 15/01/2026

1. Informations générales

- **Client** : Application interne (confidentiel)
- **Type de test** : Test d'intrusion applicatif (Web Pentest)
- **Méthodologie** : DVWA
- **Outils utilisés** : DVWA, Burp Suite
- **Date du test** : 24/12/2025

2. Résumé exécutif

L'objectif de ce test d'intrusion était d'évaluer le niveau de sécurité d'une application web interne déployée sans audit préalable.

Les tests ont mis en évidence **plusieurs vulnérabilités critiques**, notamment des failles d'injection SQL et des faiblesses de validation des entrées utilisateur.

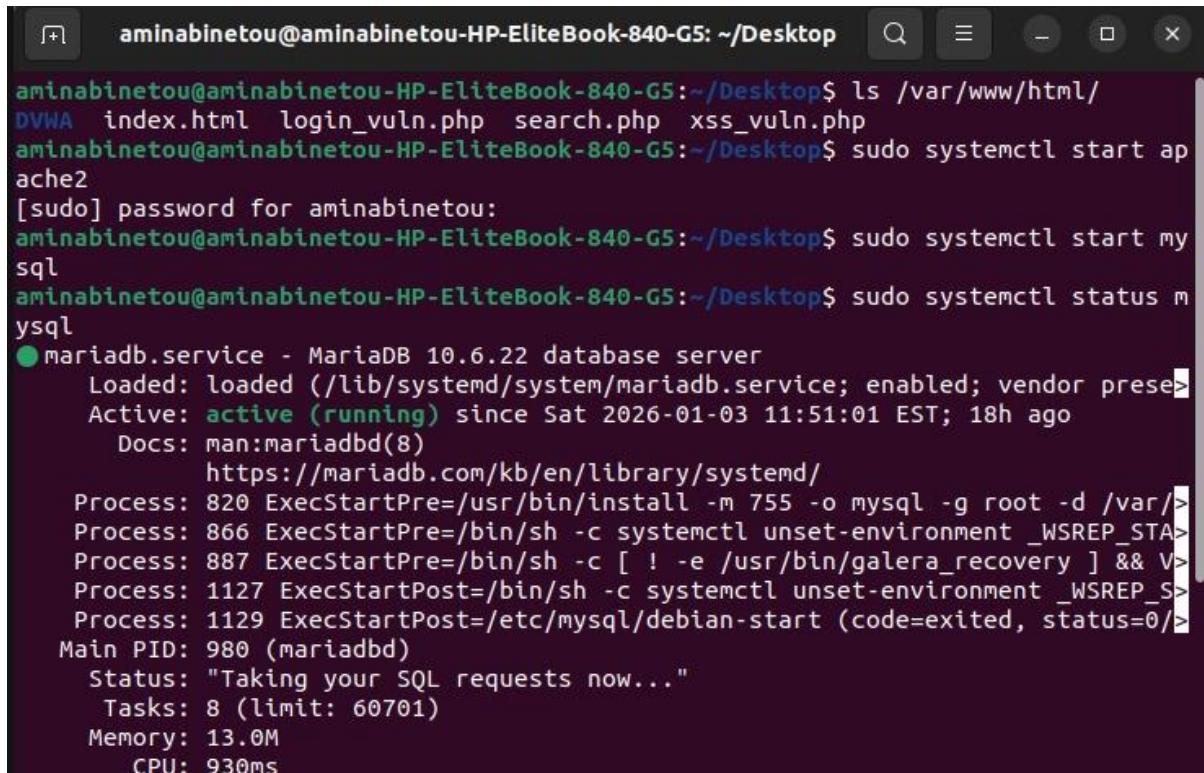
Certaines vulnérabilités permettent :

- L'accès non autorisé aux données
- L'exécution de code malveillant côté client
- Des actions à l'insu des utilisateurs authentifiés

⚠ corrections immédiates avant toute mise en production étendue.

Etape 1 : téléchargement de Apache2, MySQL,

Avant de commencer toute chose on doit d'abord mettre en place apache my sql et autres pour pouvoir commencer le travail normalement



```
aminabinetou@aminabinetou-HP-EliteBook-840-G5:~/Desktop$ ls /var/www/html/
DVWA index.html login_vuln.php search.php xss_vuln.php
aminabinetou@aminabinetou-HP-EliteBook-840-G5:~/Desktop$ sudo systemctl start apache2
[sudo] password for aminabinetou:
aminabinetou@aminabinetou-HP-EliteBook-840-G5:~/Desktop$ sudo systemctl start mysql
aminabinetou@aminabinetou-HP-EliteBook-840-G5:~/Desktop$ sudo systemctl status mysql
● mariadb.service - MariaDB 10.6.22 database server
   Loaded: loaded (/lib/systemd/system/mariadb.service; enabled; vendor prese>
   Active: active (running) since Sat 2026-01-03 11:51:01 EST; 18h ago
     Docs: man:mariadb(8)
           https://mariadb.com/kb/en/library/systemd/
  Process: 820 ExecStartPre=/usr/bin/install -m 755 -o mysql -g root -d /var/>
  Process: 866 ExecStartPre=/bin/sh -c systemctl unset-environment _WSREP_STA>
  Process: 887 ExecStartPre=/bin/sh -c [ ! -e /usr/bin/galera_recovery ] && V>
  Process: 1127 ExecStartPost=/bin/sh -c systemctl unset-environment _WSREP_S>
  Process: 1129 ExecStartPost=/etc/mysql/debian-start (code=exited, status=0/>
 Main PID: 980 (mariadb)
   Status: "Taking your SQL requests now..."
      Tasks: 8 (limit: 60701)
     Memory: 13.0M
        CPU: 930ms
```

La capture montre un environnement de test de sécurité web fonctionnel.

Les fichiers de l'application web vulnérable sont bien présents dans /var/www/html (DVWA, pages vulnérables à la SQL Injection et au XSS).

Le serveur **Apache** et la base de données **MariaDB (MySQL)** ont été démarrés avec succès et sont en cours d'exécution.

👉 L'application est donc opérationnelle et prête pour réaliser des **tests d'intrusion web** (SQLi, XSS, etc.) avec OWASP ZAP ou Burp Suite.

The screenshot shows the DVWA setup interface. On the left, there's a sidebar with buttons for 'Setup DVWA', 'Instructions', and 'About'. The main area is titled 'Database Setup' with a wrench icon. It says: 'Click on the 'Create / Reset Database' button below to create or reset your database. If you get an error make sure you have the correct user credentials in: /var/www/html/DVWA/config/config.inc.php'. Below that, it says: 'If the database already exists, it will be cleared and the data will be reset. You can also use this to reset the administrator credentials ("admin // password") at any stage.' A horizontal line separates this from the 'Setup Check' section. Under 'Setup Check', there are several sections: 'General' (Operating system: *nix), 'DVWA version' (Git reference: 47bf4292134f454d6d6639ba2be543931b861ff1, Author: Robin Wood), 'reCAPTCHA key: Missing', 'Apache' (Web Server SERVER_NAME: localhost, mod_rewrite: Not Enabled - mod_rewrite is required for the AP labs.), 'PHP' (PHP version: 8.1.2-1ubuntu2.22, PHP function display_errors: Disabled, PHP function display_startup_errors: Disabled, PHP function allow_url_include: Disabled - Feature deprecated in PHP 7.4, see lab for more information, PHP function allow_url_fopen: Enabled, PHP module gd: Missing - Only an issue if you want to play with captchas, PHP module mysql: Installed, PHP module pdo_mysql: Installed), 'Database' (Backend database: MySQL/MariaDB, Database username: dvwa, Database password: *****, Database database: dvwa, Database host: 127.0.0.1, Database port: 3306), and 'API' (This section is only important if you want to use the API module, Vendor files installed: Not Installed). At the bottom, it says: 'For information on how to install these, see the README.' and 'Status in red, indicate there will be an issue when trying to complete some modules.' It also notes: 'If you see disabled on either allow_url_fopen or allow_url_include, set the following in your php.ini file and restart Apache.'

Etape 2 : Ouverture de DVWA



Username

Password

[Damn Vulnerable Web Application \(DVWA\)](#)

On doit mettre l'identifiant et le mot de passe pour pouvoir accéder au page de garde directement

La page si dessous nous montre le résultat mais aussi elle est vulnérable





DVWA Security 

Security Level

Security level is currently: **low**.

You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of DVWA:

1. Low - This security level is completely vulnerable and **has no security measures at all**. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of **bad security practices**, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of **harder or alternative bad practices** to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be **secure against all vulnerabilities**. It is used to compare the vulnerable source code to the secure source code.
Prior to DVWA v1.9, this level was known as 'high'.

Additional Tools

- [View Broken Access Control Logs](#) - View access logs for the Broken Access Control vulnerability

Security level set to low

Username: admin
Security Level: Security Level: low
Locale: en
SQLi DB: mysql

Damn Vulnerable Web Application (DVWA)

Etape 3 : installation de Burt suite

Pour faire le téléchargement on doit d'abord faire cela sur le terminal de la machine virtuelle mais cela refuse de venir donc on a fait le téléchargement manuellement

```

s Terminal
aminabinetou@aminabinetou-HP-EliteBook-840-G5: ~/Downloads$ 
Jan 03 11:51:01 aminabinetou-HP-EliteBook-840-G5 mariadb[980]: 2026-01-03 11:51:01 [Note] 
Jan 03 11:51:01 aminabinetou-HP-EliteBook-840-G5 mariadb[980]: 2026-01-03 11:51:01 [Note] 
Lines 1-23
aminabinetou@aminabinetou-HP-EliteBook-840-G5:~/Desktop$ cd Downloads
bash: cd: Downloads: No such file or directory
aminabinetou@aminabinetou-HP-EliteBook-840-G5:~/Desktop$ ls
AutopsyCases      privatekey    to-be-copied.txt    vault-keycloak
docker           publickey     tp-2fa-elk
'Old Firefox Data' suricata.yaml   usb_image.dd
aminabinetou@aminabinetou-HP-EliteBook-840-G5:~/Desktop$ cd ~/Downloads
aminabinetou@aminabinetou-HP-EliteBook-840-G5:~/Downloads$ ls
'Atelier2-docker avance-1.pdf'
'Atelier2-docker avance.pdf'
autopsy-4.22.1-64bit.msi
burpsuite_community_linux_v2025_11_6.sh
gophish-v0.12 (1).1-linux-64bit
gophish-v0.12.1-linux-64bit
gophish-v0.12.1-linux-64bit.zip
Iso
'Projet académique_Cyber (Récupération automatique)(1).docx'
'Projet académique_Cyber (Récupération automatique).docx'
'Test XSS.html'
'vmware-workstation-linux-17.6.1-4876
aminabinetou@aminabinetou-HP-EliteBook-840-G5:~/Downloads$ 

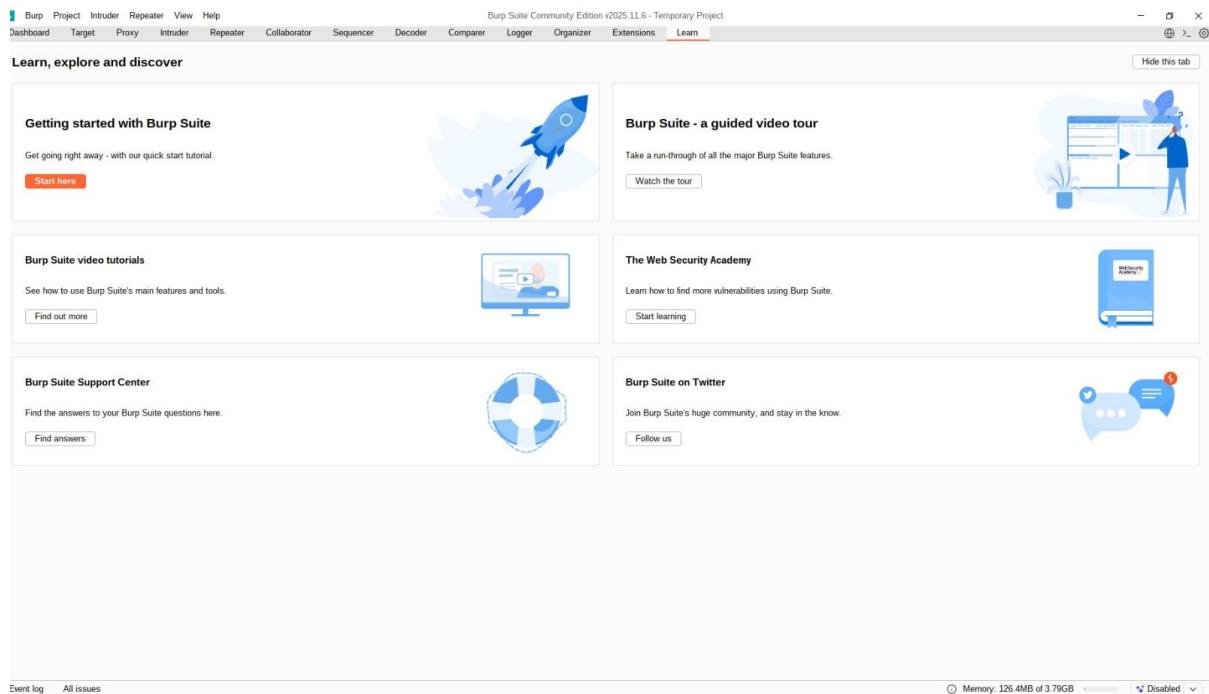
```

On a telecharger une image sur le site officiel on a pris le format lunix



A pres avoir fait les commandes correspondants y'a cette fenetre qui s'est ouvert automatiquement

1-Burt suite



Event log All issues

Memory: 126.4MB of 3.79GB Disabled

Cette image représente l'application Burp suite

On doit impérativement faire la configuration dans Firefox

Time	Type	Direction	Method	URL	Status code	Length
21:55 9 Jan ...	HTTP	→ Request	GET	http://detectportal.firefox.com/canonical.html		
21:44 9 Jan ...	HTTP	→ Request	GET	http://detectportal.firefox.com/canonical.html		
21:49 9 Jan ...	HTTP	→ Request	GET	http://detectportal.firefox.com/canonical.html		
20:25 3 Jan ...	HTTP	→ Request	GET	http://localhost/DVWA/		

La capture montre **Burp Suite (Proxy)** en fonctionnement avec l'**interception activée**.
 Burp capte les requêtes HTTP du navigateur, y compris l'accès à **DVWA (localhost)**.
 Les requêtes GET interceptées permettent d'analyser les **en-têtes HTTP**, **paramètres** et **cookies** avant leur envoi au serveur.

The screenshot shows the Burp Suite interface with the following details:

- Project Bar:** Burp Suite Community Edition v2025.11.6 - Temporary Project
- Selected Tab:** Intercept
- Request Inspector:**
 - Request: GET /DVWA/vulnerabilities/csrf/?password_current=password&password_new=tes123&password_change=Change&user_token=b44540394dc432f26
 - Response Headers: Content-Type: application/x-www-form-urlencoded; charset=UTF-8
 - Content: ... (partial view)
- Response Inspector:**
 - Request attributes: 2
 - Request query parameters: 5
 - Request cookies: 2
 - Request headers: 16
 - Response headers: 10

Cela confirme que **Burp Suite est correctement configuré** et prêt à être utilisé pour tester des vulnérabilités web (SQL Injection, XSS, CSRF, etc.).

Etape 4 : Identifier les failles (XSS, SQLi, CSRF, etc.).

Pour identifier les failles on commence par XSS

The screenshot shows the DVWA application with the following details:

- URL:** http://localhost/DVWA/vulnerabilities/xss_r/?name=<script>alert('XSS')</script>
- Page Title:** DVWA
- Content:** Vulnerability: Reflected Cross Site Scripting (XSS)
- Form:** What's your name? <input type="text" value="<script>alert('XSS')</script>"> Submit
- Message Box:** XSS (Reflected)
- More Information:**
 - https://owasp.org/www-community/attacks/xss/
 - https://owasp.org/www-community/attacks/http-evasion-cheatsheet
 - https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_%28Exploit%29
 - https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_%28Attacks%29
 - https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_%28Prevention%29
- Bottom Status:** Username: admin Security Level: Security Level: low Locale: en US MySQL DB: mysql

2- MYSQL

The screenshot shows the DVWA SQL Injection page. In the main content area, there is a table listing user information. The first row shows a user with ID 1, first name admin, and surname admin. The second row shows a user with ID 1, first name Gordon, and surname Brown. The third row shows a user with ID 1, first name Hack, and surname Me. The fourth row shows a user with ID 1, first name Pablo, and surname Picasso. The fifth row shows a user with ID 1, first name Bob, and surname Smith. Below the table, there is a section titled "More Information" with links to various resources about SQL injection.

Où fait le test et voici le résultat

3- CSRF

Ils ont les même résultat c'est pour cela on n'a pas prise de capture

4 -Analyse des requêtes HTTP de l'application DVWA via Burp Suite

The screenshot shows the Burp Suite interface with the "HTTP history" tab selected. The left pane displays a list of captured requests, and the right pane shows the details of a selected request. The selected request is a GET request to /DVWA/. The response content is a list of vulnerabilities, including Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), and CSP Bypass. The Burp Suite interface includes tabs for Request, Response, and Inspector, and various status indicators like TLS, IP, and Cookies.

La capture montre Burp Suite – onglet HTTP History interceptant le trafic vers DVWA sur localhost.

Les requêtes HTTP (GET) vers /DVWA/ et /DVWA/login.php sont enregistrées avec un

code 200, confirmant un accès réussi.

Burp affiche le **détail des requêtes et réponses**, incluant les **cookies de session (PHPSESSID)** et le contenu HTML listant les vulnérabilités DVWA (XSS, Weak Session IDs, CSP Bypass).

Cela confirme que **Burp Suite intercepte et analyse correctement les échanges HTTP**, permettant l'identification et l'exploitation de vulnérabilités web dans le cadre du pentest.

- Les

The screenshot shows the Burp Suite interface with the following details:

- HTTP history tab:** Shows a list of requests and responses. One request is selected, showing a GET to /DVWA/. The response body contains a list of vulnerabilities:
 - File Upload (Insecure CAPTCHA)
 - SQL Injection (Blind)
 - SQL Injection
 - Weak Session IDs
- Request pane:** Displays the raw request headers and body. The body includes parameters like 'v=143', 'Not A(Brand)', and 'v=24'.
- Response pane:** Displays the raw response body, which is the HTML code listing the vulnerabilities.
- Inspector pane:** Shows the request attributes, cookies, headers, and response headers for the selected request.

Se sont les mm résultats juste qu'on voit les résultats de mysql et xss

5- Classer les vulnérabilités selon CVSS.

Vulnérabilité identifiée	Type	Niveau CVSS	Justification
Injection SQL	SQLi	Critique (9.8)	Accès complet à la base de données
XSS stockée	XSS	Élevée (8.0)	Vol de session utilisateur
XSS réfléchie	XSS	Moyenne (6.1)	Nécessite une interaction de l'utilisateur
CSRF CSRF	Moyenne 6.8	Moyenne (6.8)	Exécution d'actions non autorisées

Voici une **section complète et professionnelle** que tu peux intégrer directement dans ton **rappor de pentest**.

Elle couvre **les solutions de correction** + une **formulation professionnelle**.

Recommandations et solutions de correction

À la suite du test d'intrusion réalisé sur l'application web interne, plusieurs vulnérabilités ont été identifiées. Les recommandations ci-dessous visent à corriger les failles détectées et à renforcer durablement la sécurité de l'application.

1. Injection SQL (SQL Injection)

Problème identifié :

Les entrées utilisateur ne sont pas correctement filtrées avant d'être utilisées dans des requêtes SQL.

Solutions de correction :

- Utiliser des **requêtes préparées (Prepared Statements)** ou des **ORM sécurisés**
- Éviter la concaténation directe des entrées utilisateur dans les requêtes SQL
- Mettre en place une **validation stricte côté serveur**
- Désactiver l'affichage des erreurs SQL en production
- Appliquer le principe du **moindre privilège** pour les comptes de base de données

2. Cross-Site Scripting (XSS)

Problème identifié :

Les entrées utilisateur sont renvoyées au navigateur sans encodage approprié.

Solutions de correction :

- Encoder toutes les sorties utilisateur (HTML entities)
- Valider et filtrer les données côté serveur
- Implémenter une **Content Security Policy (CSP)**
- Éviter l'utilisation de fonctions dangereuses comme innerHTML
- Séparer clairement les données et le code JavaScript

3. Cross-Site Request Forgery (CSRF)

Problème identifié :

Absence de mécanisme empêchant l'exécution de requêtes non autorisées.

Solutions de correction :

- Implémenter des **tokens CSRF uniques**
- Configurer les cookies avec les attributs SameSite, HttpOnly et Secure
- Restreindre les actions sensibles aux méthodes **POST**
- Vérifier l'en-tête Origin ou Referer

4. Mauvaise gestion des sessions

Problème identifié :

Sessions potentiellement prévisibles ou mal protégées.

Solutions de correction :

- Régénérer l'identifiant de session après authentification
- Définir une durée de session limitée
- Utiliser des cookies sécurisés (HttpOnly, Secure)
- Implémenter une déconnexion automatique après inactivité

5. Absence de headers de sécurité HTTP

Problème identifié :

Les en-têtes de sécurité ne sont pas correctement configurés.

Solutions de correction :

- Ajouter les headers suivants :
 - Content-Security-Policy
 - X-Frame-Options
 - X-Content-Type-Options
 - Strict-Transport-Security
- Forcer l'utilisation de HTTPS

Recommandations générales

- Intégrer la sécurité dans le cycle de développement (**DevSecOps**)
- Réaliser des audits de sécurité réguliers
- Former les développeurs aux bonnes pratiques OWASP
- Mettre en place une journalisation et une surveillance des événements
- Effectuer un nouveau test d'intrusion après correction

Conclusion des recommandations

La mise en œuvre des solutions proposées permettra de réduire significativement la surface d'attaque de l'application et d'améliorer son niveau de sécurité global. Une validation post-correction est indispensable afin de garantir l'efficacité des mesures appliquées.