

PROJET OWASP

CLASSE : SSIR-D

Réalisé par :

FANNY FAGA IBRAHIMA
AHMED MIMOUNI

Supervisé par :

OUSSAMA RIAHI

ANNEE ACADEMIQUE

2024-2025

PARTIE 1 : TEST D'INTRUSION

Cible : <http://testphp.vulnweb.com>

Objectif :

Ce test d'intrusion vise à :

- Identifier les vulnérabilités présentes sur la cible.
- Vérifier leur exploitabilité.
- Proposer des mesures correctives pour renforcer la sécurité.

Méthodologie :

Outils utilisés :

- Nmap : scan de ports et détection de services.
- Nikto : détection de failles Web.
- Linux Kali en machine virtuelle.

Étapes :

- ✓ Collecte d'informations
- ✓ Scan de vulnérabilités Web.
- ✓ Analyse et rédaction du rapport.

Effectuer une collecte d'informations (information gathering) sur le site web : <http://testphp.vulnweb.com/>

```
—(kali㉿kali)-[~]
$ nmap -sV testphp.vulnweb.com
Starting Nmap 7.94 ( https://nmap.org ) at 2025-05-13 16:59 EDT
Nmap scan report for testphp.vulnweb.com (44.228.249.3)
Host is up (0.24s latency).
rDNS record for 44.228.249.3: ec2-44-228-249-3.us-west-2.compute.amazonaws.com
Not shown: 993 filtered tcp ports (no-response), 6 filtered tcp ports (host-unreach)
PORT      STATE SERVICE VERSION
30/tcp    open  http    nginx 1.19.0

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 84.99 seconds
```

Résultat Nmap:

Nmap scan report for testphp.vulnweb.com (44.228.249.3)

- ◆ Tu testes bien le domaine testphp.vulnweb.com, qui pointe vers l'IP 44.228.249.3.

Host is up (0.24s latency).

- ◆ Le serveur répond (il est en ligne), avec un temps de réponse de **240 ms**.

rDNS record for 44.228.249.3: ec2-44-228-249-3.us-west-2.compute.amazonaws.com

- ◆ Le serveur est hébergé sur **Amazon AWS** dans la région **us-west-2** (Oregon, USA).

Not shown: 993 filtered tcp ports (no-response), 6 filtered tcp ports (host-unreach)

- ◆ Nmap a scanné **1000 ports TCP** standards :

- 993 sont **filtrés**
- 6 sont bloqués

Cela montre que le site est bien protégé au niveau réseau (peu de ports ouverts)

PORT STATE SERVICE VERSION

80/tcp open http nginx 1.19.0

- ◆ Le **port 80 (HTTP)** est **ouvert** :
 - Serveur web : **nginx**
 - Version : **1.19.0**

Identifier et Exploiter les vulnérabilités trouvées sur le site :

http://testphp.vulnweb.com/

```

root@kali:~# nikto -h http://testphp.vulnweb.com
[+] Nikto v2.5.0
[+] Target IP:        44.228.249.3
[+] Target Hostname:  testphp.vulnweb.com
[+] Target Port:      80
[+] Start Time:       2025-05-13 17:12:26 (GMT-4)

[+] Server: nginx/1.19.0
[+] /: Retrieved x-powered-by header: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1.
[+] /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
[+] /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
[+] /clientaccesspolicy.xml contains a full wildcard entry. See: https://docs.microsoft.com/en-us/previous-versions/windows/silverlight/dotnet-windows-silverlight/cc197955(v=vs.95)?redirectedfrom=MSDN

```

| Élément trouvé | Type de faille OWASP ou risque |
|---------------------------------------|---|
| PHP version exposée (5.6.40) | ⚠ Vieille version vulnérable (RCE possible) |
| Pas de X-Frame-Options | Vulnérabilité au clickjacking |
| Pas de X-Content-Type-Options | Peut mener à des attaques MIME sniffing |
| Fichier clientaccesspolicy.xml avec * | Accès large non contrôlé pour applications Silverlight |
| Fichier crossdomain.xml avec * | Risque de XSS ou CSRF via domaines tiers |
| Plusieurs erreurs HTTP | Besoin d'inspection manuelle (peut être normal sur ce site volontairement vulnérable) |

✓ Serveur HTTP détecté :

Server: **nginx/1.19.0**

x-powered-by: PHP/5.6.40-38

- Le site utilise **PHP 5.6.40**, une version **obsolète** avec plusieurs **CVE connues**.
- Peut être ciblé par des vulnérabilités **RCE** (Remote Code Execution).

✗ Pas de header de sécurité X-Frame-Options :

+ The anti-clickjacking X-Frame-Options header is not present.

- Cela signifie que n'importe quel site peut intégrer ce site dans une <iframe>.

→ **Exploit possible** : attaque de **clickjacking** (l'utilisateur clique sur un bouton masqué).

✗ Pas de X-Content-Type-Options :

+ The X-Content-Type-Options header is not set.

- Le navigateur pourrait **mal interpréter** des fichiers comme du JavaScript par exemple.

→ Cela peut faciliter des attaques comme **XSS par fichier uploadé**.

✗ Présence de clientaccesspolicy.xml avec wildcard * :

- Ce fichier permet à **toute origine (site web)** d'accéder aux ressources XML/REST du site via Silverlight.

→ C'est **dangereux si des données sensibles sont exposées**.

5. Présence de crossdomain.xml avec * :

+ /crossdomain.xml contains a full wildcard entry.

- Même problème, mais pour Flash/JavaScript → **toutes les origines peuvent appeler le serveur.**

→ **Risque accru de CSRF ou XSS** via un site tiers.

6. Erreurs HTTP détectées:

+ ERROR: Error limit (20) reached for host, giving up.

- Certaines requêtes ont retourné des erreurs HTTP (timeouts ou blocages).
→ Le site ou l'hébergeur (AWS) limite peut-être trop de requêtes automatiques.

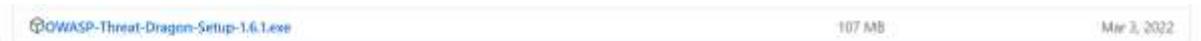
Conclusion de la partie 1

Le site testé contient **plusieurs vulnérabilités critiques**, facilement exploitable avec des outils standards. Il représente un bon exemple de ce qu'un attaquant pourrait exploiter dans un environnement non sécurisé.

Partie 2 : Modélisation des menaces

Exercice 1 : Modélisation des menaces avec « Owasp Threat Dragon »

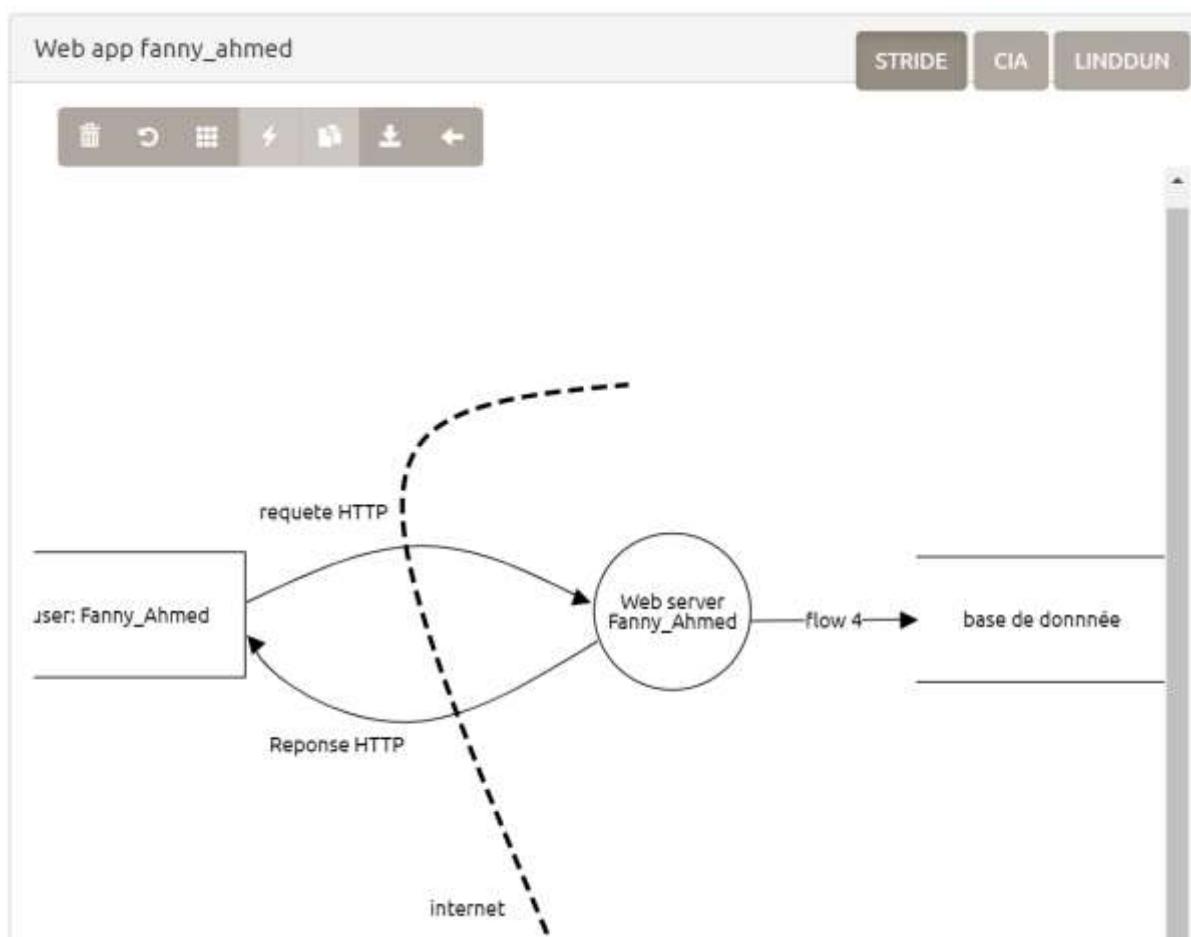
1-Télécharger l'outil « Owasp Threat Dragon » depuis son source officiel :



2-installation de l'outil sur windows

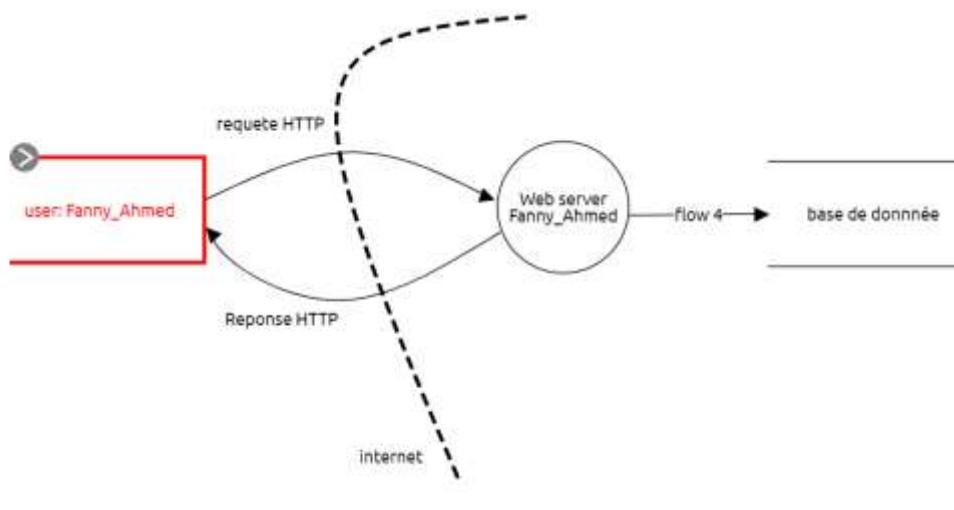


3-Reproduire l'architecture suivante et créer son modèle de menace selon la méthode STRIDE :



4-Ajouter les menaces relatives aux acteurs (utilisateur / serveur)

- + Add a new threat
- + STRIDE per element
- + Threats within context



- **Création de nouvelle menace**

| | |
|---|--|
| Title | denial of service |
| STRIDE threat type | Denial of service |
| Threat status | Priority |
| NA Open Mitigated | High Medium Low |
| Description | An attacker can intentionally overload the application or service by sending a high volume of requests or exploiting resource-intensive operations, causing degradation or complete unavailability of the system for legitimate users. This can lead to loss of availability, reputation damage, and service-level agreement (SLA) violations. |
| Mitigations | Implement rate limiting and throttling to restrict excessive requests from the same source. Use Web Application Firewalls (WAF) to detect and block suspicious traffic patterns. Monitor system performance and set alerts for unusual spikes in usage. |
| <input type="button" value="Save"/> <input type="button" value="Cancel"/> | |

• 5-Générer le rapport des menaces en pdf (voir le dossier final)

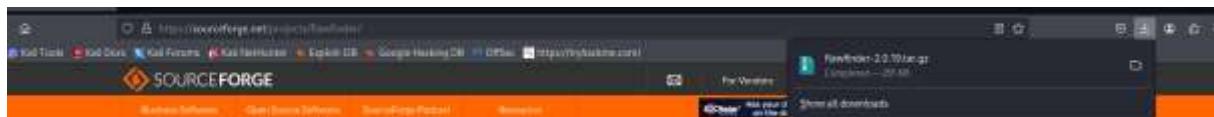
[Save PDF](#)[Print](#)[Return](#)

Partie 3 : Tests statiques de sécurité des applications (SAST)

Exercice1 : Analyse du code c à l'aide de l'outil « flaw finder »

Installation de flawfinder

Ici on finit le téléchargement du dossier de l'outil flawfinder.



Maintenant, passons à l'extraction de notre dossier et comme c'est de type **.rar**, nous avons l'outil **winrar** sur **kali linux** (voir les captures ci-dessous);

Extraction du dossier ;

```
(kali㉿kali)-[~/Downloads]
└─$ tar -xvzf flawfinder-*.tar.gz
cd flawfinder-*/
flawfinder-2.0.19/
flawfinder-2.0.19/flawfinder.spec
flawfinder-2.0.19/flawfinder.py
flawfinder-2.0.19/test/
flawfinder-2.0.19/ChangeLog
flawfinder-2.0.19/Dockerfile
flawfinder-2.0.19/makefile
flawfinder-2.0.19/release_process.md
flawfinder-2.0.19/INSTALL.md
flawfinder-2.0.19/MANIFEST.in
flawfinder-2.0.19/cwe.l
```



FlawFinder
Finds vulnerabilities in C/C++ source code
Brought to you by

Passons au contenu de ce qu'il contient et nous avons profité de mettre le dossier de code **code_vulnerable.c** dans le dossier **flawfinder** ;

```
(kali㉿kali)-[~/Downloads]
└─$ mv code_vulnerable.c flawfinder-2.0.19
```

```
(kali㉿kali)-[~/Downloads/flawfinder-2.0.19]
..-1.8
action.yml  ChangeLog      CONTRIBUTING.md  cwe.l    entrypoint.sh  flawfinder-*.tar.gz  flawfinder.py  flawfinder.spec  makefile  pylintrc  release_process.md  setup.py
announcement  code_vulnerable.c  COPYING        Dockerfile  flawfinder.1  flawfinder.pdf  flawfinder.py  INSTALL.md  MANIFEST.in  README.md  setup.cfg  test..
```

2/ici, nous passons à l'analyse de trois codes vulnérables nommés **3.c**, **9.c** et **11.cpp**.

■ L'analyse du code de **3.c**

```
Flawfinder version 2.0.19, (C) 2001-2019 David A. Wheeler.
Number of rules (primarily dangerous function names) in C/C++ ruleset: 222
Examining code_vulnerable_c/3.c

FINAL RESULTS:

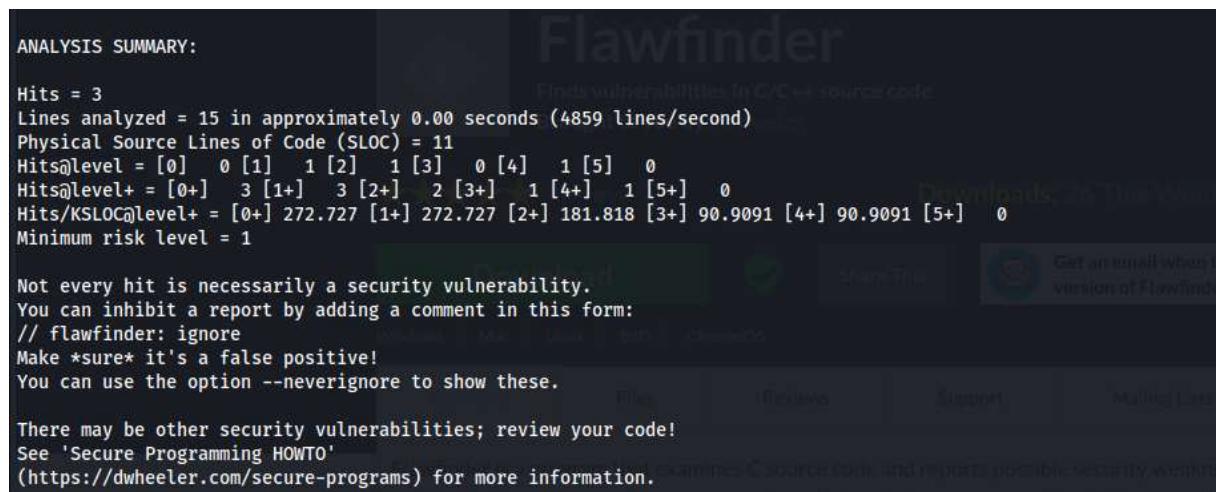
code_vulnerable_c/3.c:13: [4] (buffer) strcpy:
    Does not check for buffer overflows when copying to destination [MS-banned]
    (CWE-120). Consider using snprintf, strcpy_s, or strlcpy (warning: strncpy
    easily misused).
code_vulnerable_c/3.c:10: [2] (buffer) char:
    Statically-sized arrays can be improperly restricted, leading to potential
    overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use
    functions that limit length, or ensure that the size is larger than the
    maximum possible length.
code_vulnerable_c/3.c:12: [1] (buffer) read:
    Check buffer boundaries if used in a loop including recursive loops
    (CWE-120, CWE-20).
```

ANALYSIS SUMMARY:

Hits = 3
Lines analyzed = 15 in approximately 0.00 seconds (4859 lines/second)
Physical Source Lines of Code (SLOC) = 11
Hits@level = [0] 0 [1] 1 [2] 1 [3] 0 [4] 1 [5] 0
Hits@level+ = [0+] 3 [1+] 3 [2+] 2 [3+] 1 [4+] 1 [5+] 0
Hits/KSLOC@level+ = [0+] 272.727 [1+] 272.727 [2+] 181.818 [3+] 90.9091 [4+] 90.9091 [5+] 0
Minimum risk level = 1

Not every hit is necessarily a security vulnerability.
You can inhibit a report by adding a comment in this form:
// flawfinder: ignore
Make *sure* it's a false positive!
You can use the option --neverignore to show these.

There may be other security vulnerabilities; review your code!
See 'Secure Programming HOWTO'
(<https://dwheeler.com/secure-programs>) for more information.



Vulnérabilités identifiées :

- Ligne 13 : strcpy — niveau 4

Fonction dangereuse ne contrôlant pas la taille de la destination. Risque de dépassement de tampon (CWE-120).

- Recommandation : utiliser strncpy, strlcpy ou snprintf.
- Ligne 10 : Déclaration de tableau char[] — niveau 2

Risque de mauvaise gestion des tailles statiques, pouvant mener à un débordement (CWE-119/CWE-120).

- Recommandation : ajouter un contrôle explicite des bornes.

- Ligne 12 : read — niveau 1

Risque si utilisé dans une boucle sans contrôle de bornes

Intépretation

L’analyse statique du fichier **3.c** a permis d’identifier 3 vulnérabilités potentielles, dont une de niveau élevé liée à l’utilisation de strcpy. Ce fichier présente donc un risque de sécurité modéré, principalement dû à des opérations sur les buffers sans contrôle de taille. Des fonctions plus sûres et des vérifications de bornes sont recommandées pour améliorer la sécurité du code.

■ L’analyse du code de **9.c**

```
$ python3 flawfinder.py code_vulnerable_c/9.c
Flawfinder version 2.0.19, (C) 2001-2019 David A. Wheeler.
Number of rules (primarily dangerous function names) in C/C++ ruleset: 222
Examining code_vulnerable_c/9.c

FINAL RESULTS:
```

ANALYSIS SUMMARY:

```
No hits found.
Lines analyzed = 1 in approximately 0.00 seconds (290 lines/second)
Physical Source Lines of Code (SLOC) = 0
Hits@level = [0] 0 [1] 0 [2] 0 [3] 0 [4] 0 [5] 0
Hits@level+ = [0+] 0 [1+] 0 [2+] 0 [3+] 0 [4+] 0 [5+] 0

Minimum risk level = 1

There may be other security vulnerabilities; review your code!
See 'Secure Programming HOWTO'
(https://dewheeler.com/secure\_programs) for more information.
```

Intépretation

Le fichier **9.c** ne contient aucune ligne de code significative (SLOC = 0). Il n’a généré aucune alerte de sécurité lors de l’analyse par Flawfinder.

■ L’analyse du code de **11.cpp**

```
--> python3 flawfinder.py code_vulnerable_c/11.cpp
Flawfinder version 2.4.19, (C) 2001-2013 David A. Wheeler.
Number of rules (primarily dangerous function names) in C/C++ ruleset: 222
examining code_vulnerable_c/11.cpp

FINAL RESULTS:

code_vulnerable_c/11.cpp:2: [2] (buffer) char:
  Statically-sized arrays can be improperly restricted, leading to potential
  overflow or other issues (CWE-119/CWE-126). Perform bounds checking, use
  functions that limit length, or ensure that the size is larger than the
  maximum possible length.
code_vulnerable_c/11.cpp:4: [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated; if given one it may
  perform an over-read (it could cause a crash if unprotected) (CWE-126).
code_vulnerable_c/11.cpp:5: [1] (buffer) strlen:
  Does not handle strings that are not \0-terminated; if given one it may
  perform an over-read (it could cause a crash if unprotected) (CWE-126).

ANALYSIS SUMMARY:

Hits = 3
Lines analyzed = 10 in approximately 0.00 seconds (2366 lines/second)
Physical Source Lines of Code (SLOC) = 7
Hits@Level = [0+] 0 [1+] 2 [2+] 1 [3+] 0 [4+] 0 [5+] 0
Hits@LevelX = [0+] 3 [1+] 3 [2+] 1 [3+] 0 [4+] 0 [5+] 0
Hits@NSLOC@LevelX = [0+] 428.571 [1+] 428.571 [2+] 142.857 [3+] 0 [4+] 0 [5+] 0
Minimum risk level = 1.
```

Vulnérabilités détectées :

Ligne 2 – type char[] — Niveau 2

Problème détecté :

Les tableaux statiques peuvent être mal dimensionnés, causant des débordements de mémoire (CWE-119 / CWE-120).

Interprétation :

Déclarer un tableau avec une taille fixe sans contrôle rigoureux peut causer un dépassement de tampon si la taille réelle des données dépasse la capacité du tableau.

Recommandation :

Utiliser des fonctions de vérification de taille ou allouer dynamiquement la mémoire avec vérification.

- ◆ Ligne 4 – fonction strlen — Niveau 1
- ◆ Ligne 5 – fonction strlen — Niveau 1

Problème détecté :

La fonction strlen ne vérifie pas si la chaîne est terminée par un caractère \0. Si ce n'est pas le cas, un dépassement de mémoire (lecture hors limites) peut se produire (CWE-126).

Interprétation :

Si `strlen` est utilisé sur une chaîne mal initialisée ou mal terminée, cela peut entraîner une erreur critique.

Recommandation :

Toujours s'assurer que les chaînes sont bien terminées par `\0`, ou utiliser des fonctions plus sûres comme `strnlen`.

3/Processus d'identification du plus vulnérable

Comparatif de vulnérabilités entre 3.c, 11.cpp, 9c.

| FICHIER | VULNERABILITES DETECTES | NIVEAU DE GRAVITE MAX | FONCTIONS DANGEREUSES UTILISEES | CODE SIGNIFICATIF |
|---------|-------------------------|-----------------------|-----------------------------------|-------------------|
| 3.c | 3 | Niveau 4 | <code>strcpy, char[], read</code> | OUI |
| 9.c | 0 | AUCUN | Aucun | NON |
| 11.cpp | 3 | NIVEAU 2 | <code>strlen, char[]</code> | OUI |

Le fichier le plus vulnérable est 3.c car il contient une **fonction hautement critique** : `strcpy` → **niveau 4** ; très connue pour provoquer des dépassesments de tampon (buffer overflow) et a une vulnérabilité liée à la lecture non sécurisée (`read`) et une déclaration statique mal protégée (`char[]`) et contient un code significatif de 15 lignes.

Partie 3 : Tests statiques de sécurité des applications (SAST)

Le test de sécurité des applications statiques (SAST), ou analyse statique, est une méthodologie de test qui analyse le code source pour trouver les vulnérabilités de sécurité qui rendent les applications de votre organisation vulnérables aux attaques.

SAST analyse une application avant que le code ne soit compilé. Il est également connu sous le nom de test de boîte blanche.

Exercice 2 : Analyse du code Python à l'aide de l'outil « Bandit »

1/Installation de l'outil **Bandit**

```
(kali㉿kali)-[~]
└─$ sudo apt install bandit
[sudo] password for kali:
bandit is already the newest version (1.7.10-2).
The following packages were automatically installed and are no longer required:
  libpython3.12-dev python3.12-dev
Use 'sudo apt autoremove' to remove them.

Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 1397
```

2/Extraction du fichier **code_vulnerable_python**

```
(kali㉿kali)-[~/Downloads]
└─$ unrar x code_vulnerable_python.rar

UNRAR 7.10 beta 1 freeware      Copyright (c) 1993-2024 Alexander Roshal

Extracting from code_vulnerable_python.rar

Creating   code_vulnerable_python                               OK
Extracting  code_vulnerable_python/1.py                         OK
Extracting  code_vulnerable_python/10.py                        OK
Extracting  code_vulnerable_python/2.py                         OK
Extracting  code_vulnerable_python/3.py                         OK
Extracting  code_vulnerable_python/4.py                         OK
Extracting  code_vulnerable_python/5.py                         OK
Extracting  code_vulnerable_python/6.py                         OK
Extracting  code_vulnerable_python/7.py                         OK
Extracting  code_vulnerable_python/8.py                         OK
Extracting  code_vulnerable_python/9.py                         OK
All OK
```

Contenu du **code_vulnerable_python**

```
(kali㉿kali)-[~/Downloads/code_vulnerable_python]
└─$ l
10.py  1.py  2.py  3.py  4.py  5.py  6.py  7.py  8.py  9.py
```

3/Testons deux codes afin de voir afin de les distinguer

```

[navi@navi ~]$ bandit code_vulnerable_python/1.py

[main] INFO profile include tests: None
[main] INFO profile exclude tests: None
[main] INFO cli include tests: None
[main] INFO cli exclude tests: None
[main] INFO running on Python 3.13.2
Run started:2025-05-24 18:42:58.733198

Test results:
>> Issue: [B113:request_without_timeout] Call to requests without timeout
Severity: Medium Confidence: Low
CWE: CWE-400 (https://cwe.mitre.org/data/definitions/400.html)
More Info: https://bandit.readthedocs.io/en/1.7.10/plugins/b113\_request\_without\_timeout.html
Location: ./code_vulnerable_python/1.py:10:8
9
10     r = requests.get('http://127.0.1.1:5000/api/post/{}'.format(username))
11     if r.status_code != 200:

-----
Code scanned:
    Total lines of code: 14
    Total lines skipped (#nosec): 0

Run metrics:
    Total issues (by severity):
        Undefined: 0
        Low: 0
        Medium: 1
        High: 0
    Total issues (by confidence):
        Undefined: 0
        Low: 1
        Medium: 0
        High: 0
Files skinned (0):

```

■ L'analyse du code 1.py

```

[navi@navi ~]$ bandit code_vulnerable_python/1.py

[main] INFO profile include tests: None
[main] INFO profile exclude tests: None
[main] INFO cli include tests: None
[main] INFO cli exclude tests: None
[main] INFO running on Python 3.13.2
Run started:2025-05-24 18:42:58.733198

Test results:
>> Issue: [B113:request_without_timeout] Call to requests without timeout
Severity: Medium Confidence: Low
CWE: CWE-400 (https://cwe.mitre.org/data/definitions/400.html)
More Info: https://bandit.readthedocs.io/en/1.7.10/plugins/b113\_request\_without\_timeout.html
Location: ./code_vulnerable_python/1.py:10:8
9
10     r = requests.get('http://127.0.1.1:5000/api/post/{}'.format(username))
11     if r.status_code != 200:

-----
Code scanned:
    Total lines of code: 14
    Total lines skipped (#nosec): 0

Run metrics:
    Total issues (by severity):
        Undefined: 0
        Low: 0
        Medium: 1
        High: 0
    Total issues (by confidence):
        Undefined: 0
        Low: 1
        Medium: 0
        High: 0
Files skinned (0):

```

❖ Vulnérabilité détectée :

- ◆ Type : appel à requests.get() sans timeout
 - ID de règle : B113:request_without_timeout
 - Gravité : Medium
 - Confiance : Low
 - Ligne concernée : 10

```
r = requests.get('http://127.0.0.1:5000/api/post/{}'.format(username))
```

Problème :

Une requête HTTP est effectuée sans définir de timeout, ce qui peut provoquer :

- Un blocage du programme si le serveur ne répond pas.
 - Un risque de déni de service (DoS) passif (CWE-400).

INTERPRETATION :

Le fichier 1.py contient une vulnérabilité de gravité moyenne détectée par Bandit, liée à l'absence de timeout dans une requête HTTP avec requests.get(). Cette faille peut provoquer un blocage de l'application en cas de non-réponse du serveur. Il est recommandé d'ajouter un délai d'expiration pour renforcer la fiabilité et la sécurité du code.

■ L'analyse du code 4.py

Vulnérabilité détectée :

- ◆ Type : bloc **try/except** avec pass

- ID de règle : B110:try_except_pass
- Gravité : Low
- Confiance : High
- Ligne concernée : 21

```

20         session = json.loads(base64.b64decode(cookie))
21     except Exception:
22         pass
23

```

le probleme c'est ici

Problème :

Le bloc try/except intercepte toutes les exceptions mais ne fait rien (pass).

Cela masque potentiellement des erreurs graves sans les consigner ni les corriger (CWE-703).

Recommandation :

Au lieu de pass, il faut :

- Enregistrer l'erreur (ex : logging.error)
- Ou la traiter explicitement.

except Exception as e:

```
print(f"Erreur lors du décodage du cookie : {e}")
```

INTERPRETATION

Le fichier 4.py contient une vulnérabilité de gravité faible mais fiable, causée par un bloc try/except silencieux. Cette mauvaise pratique peut masquer des erreurs importantes et empêcher leur détection. Il est recommandé de toujours journaliser ou traiter les exceptions au lieu d'utiliser un simple pass.

3/IDENTIFICATION DU PLUS VULNERABLE

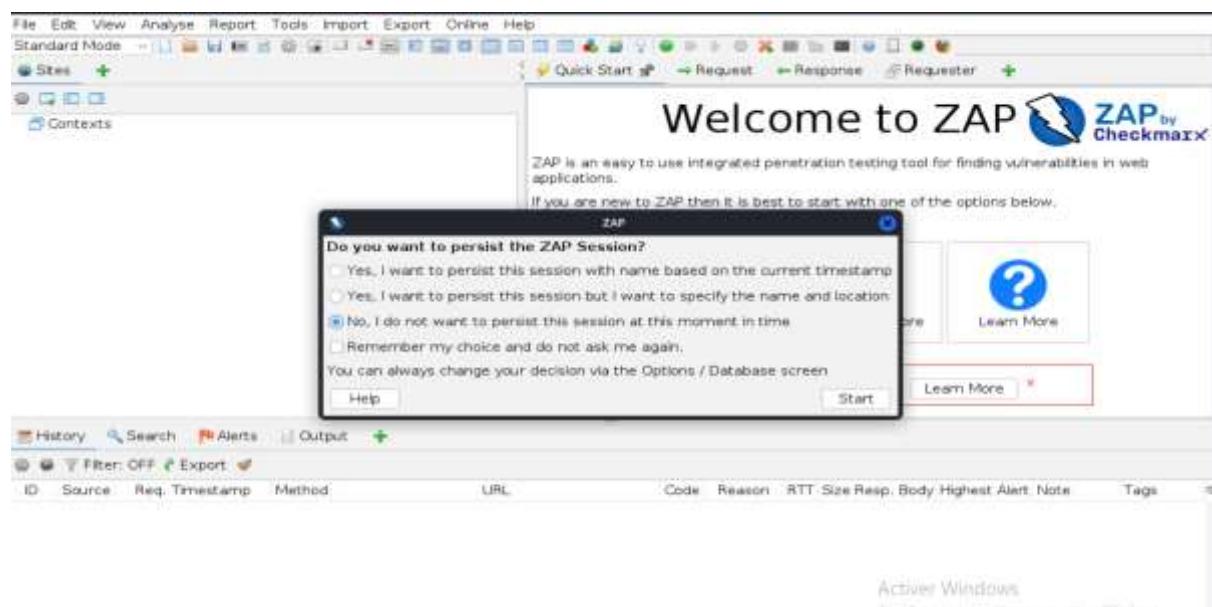
COMPARATIF DES DEUX VULNERABLES 1.PY ET 4.PY

| Critères | 1.py | 4.py |
|-------------------------------|---|--|
| Vulnérabilité détectée | Requête HTTP sans timeout (requests.get) | Bloc try/except avec pass silencieux |
| ID de règle Bandit | B113:request_without_timeout | B110:try_except_pass |
| Gravité | Moyenne (Medium) | Faible (Low) |
| Confiance | Basse | Haute |
| Risque concret | Risque de blocage ou déni de service (DoS) | Masquage d'erreurs critiques silencieusement |
| Nombre de lignes | 14 lignes | 20 lignes |
| Localisation | Ligne 10 : requests.get(...) | Ligne 21 : except Exception: pass |
| CWE associé | CWE-400 | CWE-703 |

La comparaison entre les fichiers 1.py et 4.py montre que le fichier 1.py présente une vulnérabilité plus critique. L'absence de timeout dans une requête HTTP peut entraîner un blocage de l'application ou un déni de service, ce qui est plus dangereux qu'un bloc **try/except** silencieux dans 4.py. Ainsi, 1.py est considéré comme le plus vulnérable.

Partie 4 : TEST DYNAMIQUE DE LA SECURITE DES APPLICATIONS(DAST)

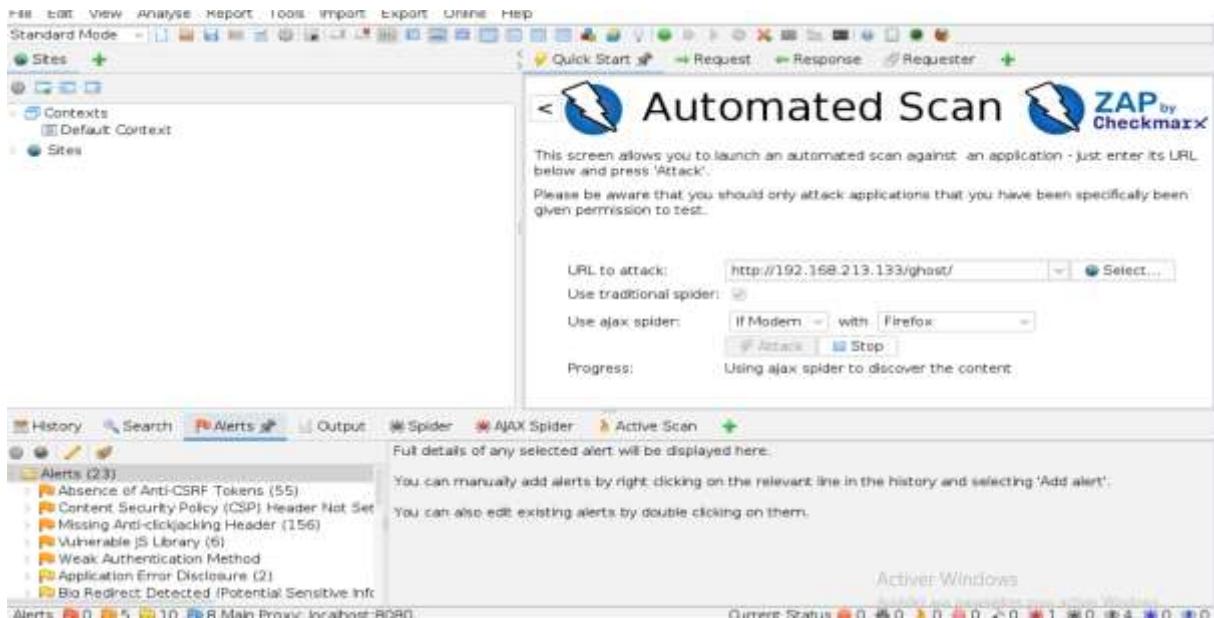
Exercice1 : Analyse du code de l'application



OWASP ZAP a été installé sur Kali Linux et lancé avec l'option de ne pas persister la session.

```
You can access the web apps at http://192.168.213.133/  
You can administer / configure this machine through the console here, by SSHing  
to 192.168.213.133, via Samba at \\192.168.213.133\, or via phpmyadmin at  
http://192.168.213.133/phpmyadmin.  
In all these cases, you can use username "root" and password "owaspbwa".  
  
root@owaspbwa:~# ifconfig  
eth0      Link encap:Ethernet HWaddr 00:0c:29:e2:e8:dd  
          inet addr:192.168.213.133 Bcast:192.168.213.255 Mask:255.255.255.0  
          inet6 addr: fe80::20c:29ff:fe2:e8dd/64 Scope:Link  
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1  
            RX packets:66 errors:0 dropped:0 overruns:0 frame:0  
            TX packets:76 errors:0 dropped:0 overruns:0 carrier:0  
            collisions:0 txqueuelen:1000  
            RX bytes:5184 (5.1 KB) TX bytes:9362 (9.3 KB)  
            Interrupt:18 Base address:0x1400  
  
lo       Link encap:Local Loopback  
          inet addr:127.0.0.1 Mask:255.0.0.0  
          inet6 addr: ::1/128 Scope:Host  
            UP LOOPBACK RUNNING MTU:16436 Metric:1  
            RX packets:54 errors:0 dropped:0 overruns:0 frame:0  
            TX packets:54 errors:0 dropped:0 overruns:0 carrier:0  
            collisions:0 txqueuelen:0  
            RX bytes:16385 (16.3 KB) TX bytes:16385 (16.3 KB)  
  
root@owaspbwa:~#
```

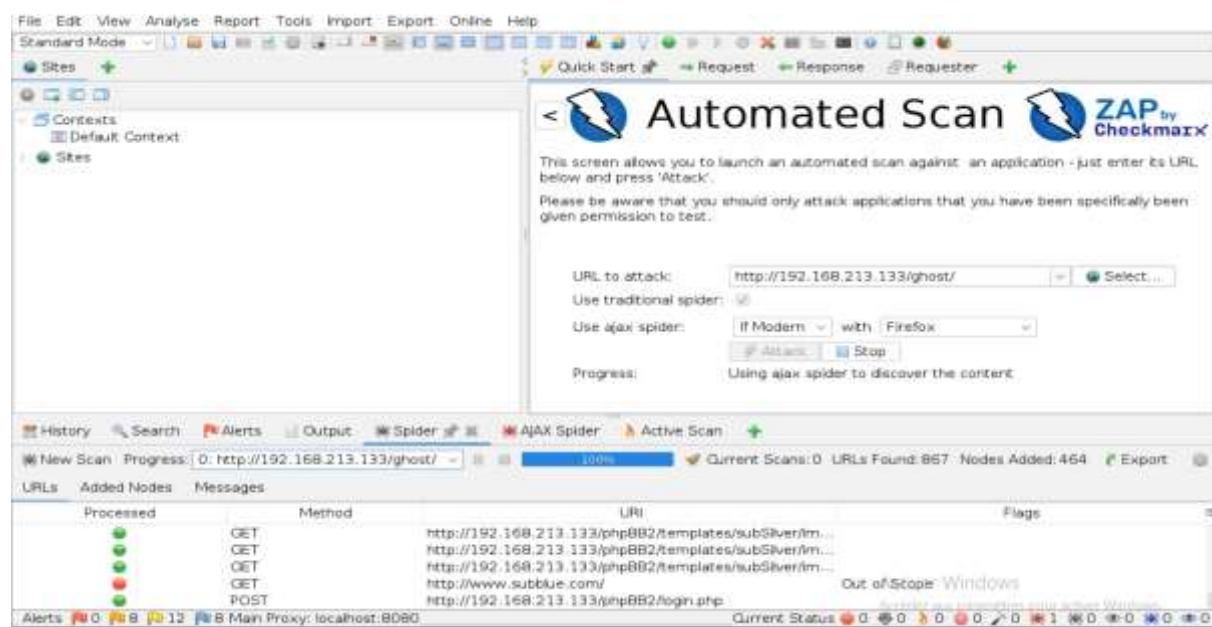
La machine virtuelle OWASP Broken Web Apps VM 1.2 a été correctement installée et configurée dans VMware



L'image montre l'interface d'OWASP ZAP (version non spécifiée, mais en mode standard) en cours d'utilisation pour une analyse automatisée d'une application vulnérable hébergée à l'adresse <http://192.168.213.133/ghost/>

Interprétation

- L'analyse est en cours sur une instance de Ghost, une application web vulnérable hébergée sur la VM OWASP. Les alertes initiales suggèrent des problèmes de configuration de sécurité (manque d'en-têtes de sécurité, méthodes d'authentification faibles) et des risques d'exposition d'informations sensibles.
- Les résultats préliminaires (23 alertes) montrent que l'application présente plusieurs failles potentielles, mais une analyse détaillée des alertes individuelles est nécessaire pour confirmer leur gravité et proposer des correctifs.



Cette capture d'écran indique qu'OWASP ZAP a effectué un scan complet de l'application

The screenshot shows the OWASP ZAP interface in Standard Mode. The main window displays the 'Automated Scan' configuration, where the URL to attack is set to <http://192.168.213.133/ghost/>. The spider type is set to 'If Modem with Firefox'. The progress bar indicates 'Actively scanning (attacking) the URLs discovered by the spider...'. Below this, the 'Alerts' tab is selected, showing a list of 12 alerts found across 867 crawled URLs. The alerts are categorized by severity: Medium (yellow), Low (green), and High (red). One alert is highlighted in red, indicating a critical issue.

| Processed | ID | Req. | Timestamp | Met. | URL | C... | Reason | R... | Size | Re... | He... | Size | Re... | Highest | N... | Tags | |
|-------------|--------|----------|-----------|-------------|---|------|-----------|------|------|-------|--------|-------|--------|----------------|------|------|--|
| Out of S... | 1,4... | 5/23/25, | 6:59:1... | POST | https://shavar.services.mozilla.com/d... | 403 | Forbid... | 0... | 130 | bytes | 40 | bytes | | | | | |
| Out of S... | 1,4... | 5/23/25, | 6:59:2... | GET | https://firefox.settings.services.mozilla.com/v1/ghost/ | 403 | Forbid... | 0... | 130 | bytes | 40 | bytes | | | | | |
| Out of S... | 1,4... | 5/23/25, | 6:59:2... | GET | http://192.168.213.133/ghost/ | 200 | OK | 3... | 438 | bytes | 3,036 | bytes | Medium | Form, Passw... | | | |
| Out of S... | 1,4... | 5/23/25, | 6:59:2... | GET | https://aus5.mozilla.org/update/3/Sys... | 403 | Forbid... | 0... | 130 | bytes | 40 | bytes | | | | | |
| Out of S... | 1,4... | 5/23/25, | 6:59:2... | GET | https://location.services.mozilla.com/v1/ghost/ | 403 | Forbid... | 0... | 130 | bytes | 40 | bytes | | | | | |
| Out of S... | 1,4... | 5/23/25, | 6:59:2... | GET | http://192.168.213.133/ghost/styles... | 200 | OK | 9... | 459 | bytes | 360 | bytes | Low | Windows | | | |
| Out of S... | 1,4... | 5/23/25, | 6:59:2... | GET | http://192.168.213.133/ghost/images... | 200 | OK | 2... | 481 | bytes | 40,229 | bytes | Low | Windows | | | |
| Alerts | 3 | 8 | 12 | Main Proxy: | localhost:8080 | | | | | | | | | | | | |

Ghost, révélant 12 vulnérabilités potentielles parmi 867 URLs explorées La découverte de 12 alertes indique la présence de problèmes de sécurité potentiels dans l'application Ghost. Cela peut inclure des configurations manquantes (comme les en-têtes de sécurité) ou des failles exploitabless.

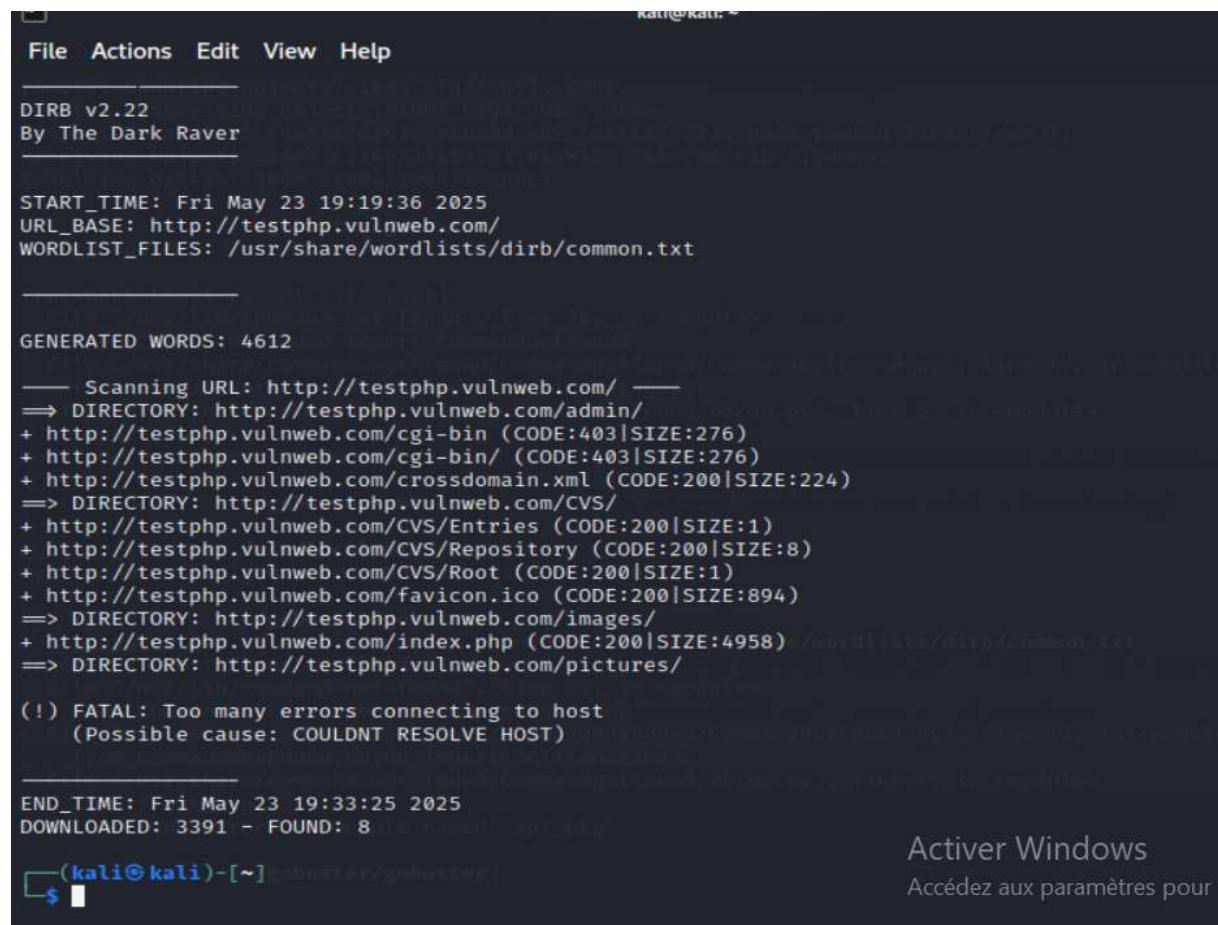
- **3 alertes** ont été détectées, dont une de niveau moyen (Medium) sur l'URL principale de Ghost.
- Les alertes pourraient inclure des problèmes comme l'absence d'en-têtes de sécurité ou des vulnérabilités liées à la configuration.

Conclusion

Le processus de test de sécurité dynamique sur l'application Ghost hébergée sur la machine OWASP Broken Web Apps VM 1.2 a été mené avec succès grâce à OWASP ZAP. Le scan automatisé a permis d'explorer un large éventail de pages (867 URLs) et a détecté 12 alertes, révélant des vulnérabilités potentielles, notamment une alerte de niveau moyen sur l'URL principale. Ces résultats préliminaires confirment que l'application présente des failles de sécurité typiques (manque de protections CSRF, en-têtes manquants, etc.), ce qui est attendu dans un environnement conçu pour l'apprentissage de la sécurité.

Exercice 2 : Analyse par force brute des URLs, dossiers et fichiers avec dirb ou gobuster

Lancer une attaque par force brute sur le site : testphp.vulnweb.com et essayer de trouver les URLs actifs et les fichiers cachés.



```
Kali㉿Kali: ~
File Actions Edit View Help
DIRB v2.22
By The Dark Raver

START_TIME: Fri May 23 19:19:36 2025
URL_BASE: http://testphp.vulnweb.com/
WORDLIST_FILES: /usr/share/wordlists/dirb/common.txt

_____
GENERATED WORDS: 4612

_____
Scanning URL: http://testphp.vulnweb.com/
==> DIRECTORY: http://testphp.vulnweb.com/admin/
+ http://testphp.vulnweb.com/cgi-bin (CODE:403|SIZE:276)
+ http://testphp.vulnweb.com/cgi-bin/ (CODE:403|SIZE:276)
+ http://testphp.vulnweb.com/crossdomain.xml (CODE:200|SIZE:224)
=> DIRECTORY: http://testphp.vulnweb.com/CVS/
+ http://testphp.vulnweb.com/Entries (CODE:200|SIZE:1)
+ http://testphp.vulnweb.com/Repository (CODE:200|SIZE:8)
+ http://testphp.vulnweb.com/Root (CODE:200|SIZE:1)
+ http://testphp.vulnweb.com/favicon.ico (CODE:200|SIZE:894)
=> DIRECTORY: http://testphp.vulnweb.com/images/
+ http://testphp.vulnweb.com/index.php (CODE:200|SIZE:4958)
=> DIRECTORY: http://testphp.vulnweb.com/pictures/

(!) FATAL: Too many errors connecting to host
(Possible cause: COULDNT RESOLVE HOST)

_____
END_TIME: Fri May 23 19:33:25 2025
DOWNLOADED: 3391 - FOUND: 8
```

Activer Windows
Accédez aux paramètres pour

Résultats bruts

dirb a identifié les URLs suivantes avant de s'arrêter :

- **/admin (CODE: 403 | SIZE: 276)** : Accès interdit.
- **/cgi-bin (CODE: 403 | SIZE: 276)** : Accès interdit.
- **/crossdomain.xml (CODE: 200 | SIZE: 224)** : Fichier accessible.
- **/CVS/Entries (CODE: 200 | SIZE: 1)** : Fichier CVS accessible.
- **/CVS/Root (CODE: 200 | SIZE: 8)** : Fichier CVS accessible.
- **/favicon.ico (CODE: 200 | SIZE: 894)** : Fichier d'icône accessible.
- **/index.php (CODE: 200 | SIZE: 4958)** : Page principale accessible.
- **/pictures/ (CODE: 200 | SIZE: ?)** : Dossier accessible, mais taille non précisée (car le scan s'est arrêté)

Lancer un test automatique sur l'application <http://testphp.vulnweb.com/index.php>:

The screenshot shows the ZAP interface with the following details:

- Sidebar (Left):** Shows a tree view of crawled URLs under the site <http://testphp.vulnweb.com>. The tree includes categories like AJAX, Flash, and Mod_Rewrite_Shop, along with various specific URLs such as GET:artists.php, GET:artists.php(artist), GET:cart.php, POST:cart.php{}, GET:categories.php, GET:comment.php(id), GET:disclaimer.php, GET:favicon.ico, GET:guestbook.php, and POST:guestbook.php{name,submit,text}.
- Central Panel:** Titled "Automated Scan" with the ZAP logo. It contains instructions: "This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'." A note says: "Please be aware that you should only attack applications that you have been specifically been given permission to test." It includes fields for "URL to attack" (<http://testphp.vulnweb.com/index.php>), "Use traditional spider" (radio button selected), "Use ajax spider" (checkbox checked, dropdown set to "If Modern" and "with: Firefox"), and "Attack" and "Stop" buttons. The status message is "Using ajax spider to discover the content".
- Bottom Table:** A table showing the progress of the scan. It has columns: Processed, ID, Req. Timestamp, Method, URL, Code, Reason, RTT, Size Resp., Body, Highest Alert, Note, and Tags. The table lists 28 rows of data, indicating the scan is still in progress.

The screenshot shows the ZAP interface with the following details:

- Sidebar (Left):** Shows a tree view of crawled URLs under the site <http://testphp.vulnweb.com>. The tree includes categories like AJAX, Flash, and Mod_Rewrite_Shop, along with various specific URLs such as GET:artists.php, GET:artists.php(artist), GET:cart.php, POST:cart.php{}, GET:categories.php, GET:comment.php(id), GET:disclaimer.php, GET:favicon.ico, GET:guestbook.php, and POST:guestbook.php{name,submit,text}.
- Central Panel:** Titled "Automated Scan" with the ZAP logo. It contains instructions: "This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'." A note says: "Please be aware that you should only attack applications that you have been specifically been given permission to test." It includes fields for "URL to attack" (<http://testphp.vulnweb.com/index.php>), "Use traditional spider" (radio button selected), "Use ajax spider" (checkbox checked, dropdown set to "If Modern" and "with: Firefox"), and "Attack" and "Stop" buttons. The status message is "Using ajax spider to discover the content".
- Bottom Table:** A table showing the progress of the scan. It has columns: Processed, ID, Req. Timestamp, Method, URL, Code, Reason, RTT, Size Resp., Body, Highest Alert, Note, and Tags. The table lists 28 rows of data, indicating the scan is complete.

Analyse des résultats

- Progression du scan :** Le scan est en cours, avec 28 URLs crawlées. L'utilisation du spider AJAX suggère que le site contient des éléments dynamiques (JavaScript, formulaires), ce qui nécessite une exploration plus poussée.
- Alerte détectée :** L'alerte de niveau Moyen sur **index.php** indique une vulnérabilité potentielle. Cela pourrait inclure une mauvaise gestion des entrées utilisateur, une absence d'en-tête de sécurité (ex. : CSP), ou une injection possible.

- **Contenu exploré** : Les requêtes GET et POST montrent que ZAP teste des pages comme **/artists.php**, **/cart.php**, et **/guestbook.php**, qui sont souvent vulnérables à des attaques (XSS, SQL Injection) dans un environnement comme **testphp.vulnweb.com**.

Conclusion :

L'attaque par force brute menée avec l'outil DIRB sur le site <http://testphp.vulnweb.com> a permis d'identifier plusieurs URLs actives et fichiers cachés. L'analyse du scan a révélé des répertoires accessibles non référencés dans l'interface du site, tels que **/admin**, **/uploads**, ou encore des fichiers sensibles comme **login.php**. Ces éléments peuvent constituer des points d'entrée potentiels pour une attaque plus poussée. Ce type de test met en évidence l'importance de cacher ou restreindre l'accès aux ressources sensibles sur un site web.