Sécurité des Applications Web - Jour 5

m2iformation.fr

Sécurité des Applications Web

Jour 5 : Automatisation SAST/DAST et Durcissement

Sécurité du Chemin d'Exécution

Objectifs du Jour 5

- Intégrer des analyses de sécurité automatisées dans les pipelines CI/CD
- Configurer et interpréter les résultats SAST
- Configurer et interpréter les résultats DAST
- Durcir le système d'exploitation
- Durcir les serveurs web et bases de données

Module 1

Analyse Statique (SAST)

SAST - Principe

Definition

Static Application Security Testing analyse le code source sans l'exécuter pour détecter des vulnérabilités potentielles.

Avantages et limites

Avantages	Limites
Détection précoce	Faux positifs fréquents
Couverture complète du code	Ne détecte pas les failles runtime
Intégration IDE possible	Nécessite accès au code source
Explication du problème	Configuration complexe

SonarQube - Configuration

Installation Docker

```yaml
# docker-compose.yml pour SonarQube
version: '3'
services:
  sonarqube:
    image: sonarqube:community
    ports:
      - "9000:9000"
    environment:
      # Configuration de la base de données PostgreSQL
      - SONAR_JDBC_URL=jdbc:postgresql://db:5432/sonar
      - SONAR_JDBC_USERNAME=sonar
      - SONAR_JDBC_PASSWORD=sonar
    volumes:
      # Persistance des données et configurations
      - sonarqube_data:/opt/sonarqube/data
      - sonarqube_extensions:/opt/sonarqube/extensions
    depends_on:
      - db
```

SonarQube - Analyse d'un Projet

```
# Installation du scanner SonarQube
# Le scanner analyse le code et envoie les résultats au serveur
npm install -g sonarqube-scanner

# Configuration du projet : fichier sonar-project.properties
# Ce fichier définit les paramètres de l'analyse

# Contenu de sonar-project.properties :
# sonar.projectKey=mon-projet
# sonar.projectName=Mon Projet
# sonar.sources=src
# sonar.language=js
# sonar.sourceEncoding=UTF-8

# Lancement de l'analyse
# -Dsonar.host.url : adresse du serveur SonarQube
# -Dsonar.login : token d'authentification généré dans SonarQube
sonar-scanner \
  -Dsonar.host.url=http://localhost:9000 \
  -Dsonar.login=token-généré-dans-sonarqube
```

Règles SAST pour la Sécurité

Catégories de vulnérabilités détectées

Catégorie	Exemples
Injection	SQL, XSS, Command
Cryptographie	Algorithmes faibles, clés en dur
Authentication	Credentials en clair
Configuration	Debug activé, CORS ouvert
Sensitive Data	Données personnelles loguées

Exemple de Rapport SAST

```
VULNERABILITIES FOUND: 12

CRITICAL (2):
- SQL Injection in UserController.java:45
  String query = "SELECT * FROM users WHERE id=" + userId;

- Hardcoded Password in Config.java:12
  private static final String DB_PASSWORD = "admin123";

HIGH (4):
- XSS Vulnerability in templates/user.html:23
- Weak Cryptographic Algorithm in HashUtil.java:8
  MessageDigest.getInstance("MD5");

MEDIUM (6):
- Missing CSRF Protection in forms
- Insecure Cookie Configuration
```

Intégration CI/CD - GitLab

```yaml
# .gitlab-ci.yml avec analyse SAST
stages:
  - build
  - test
  - security
  - deploy

sast-analysis:
  stage: security
  image: sonarsource/sonar-scanner-cli
  script:
    - sonar-scanner
        -Dsonar.projectKey=${CI_PROJECT_NAME}
        -Dsonar.sources=src
        -Dsonar.host.url=${SONAR_URL}
        -Dsonar.login=${SONAR_TOKEN}
  # L'étape échoue si le Quality Gate n'est pas passé
  allow_failure: false
```

Quality Gates

Configuration des seuils

```
Quality Gate : "Production Ready"

Conditions de blocage :
- Nouvelles vulnérabilités : 0
- Nouveaux bugs bloquants : 0
- Couverture de code : > 80%
- Duplication de code : < 3%
- Nouveaux code smells critiques : 0
```

Le pipeline échoue si ces conditions ne sont pas remplies, empechant le déploiement de code non conforme.

Module 2

Analyse Dynamique (DAST)

DAST - Principe

Definition

Dynamic Application Security Testing teste l'application en cours d'exécution en simulant des attaques.

Comparaison SAST vs DAST

Aspect	SAST	DAST
Quand	Build time	Runtime
Accès code	Requis	Non requis
Faux positifs	Plus nombreux	Moins nombreux
Couverture	Code complet	Endpoints exposés
Vulnérabilités	Potentielles	Réelles

OWASP ZAP - Configuration

```
# Lancement de ZAP en mode daemon
# Le mode daemon permet l'utilisation via API
docker run -u zap -p 8080:8080 -p 8090:8090 \
    owasp/zap2docker-stable zap.sh -daemon \
    -host 0.0.0.0 -port 8080 \
    -config api.key=votre-api-key

# Le port 8080 est le proxy ZAP
# Le port 8090 est l'API ZAP
# L'API permet de contrôler ZAP programmatiquement
```

ZAP - Scan Automatisé

```
# Script de scan ZAP pour CI/CD
# zap-baseline.py effectue un scan rapide

docker run -t owasp/zap2docker-stable zap-baseline.py \
    -t https://application-cible.com \
    -r rapport.html \
    -x rapport.xml \
    -J rapport.json

# Options importantes :
# -t : URL cible à scanner
# -r : rapport HTML pour lecture humaine
# -x : rapport XML pour intégration outils
# -J : rapport JSON pour traitement automatisé
# -l WARN : niveau minimum d'alerte à reporter
```

ZAP - Scan Complet

```
# zap-full-scan.py effectue un scan approfondi
# Plus long mais plus complet que baseline

docker run -t owasp/zap2docker-stable zap-full-scan.py \
    -t https://application-cible.com \
    -r rapport.html \
    -d  # Mode debug pour plus de détails

# Différences avec baseline :
# - Spider actif pour découvrir plus de pages
# - Tests d'injection actifs
# - Durée : plusieurs minutes à plusieurs heures
# - A utiliser en environnement de staging, pas en production
```

Scan Authentifié

```
# Pour scanner les parties authentifiées de l'application
# Il faut fournir les credentials à ZAP

docker run -t owasp/zap2docker-stable zap-full-scan.py \
    -t https://application-cible.com \
    -r rapport.html \
    --auth-loginurl https://application-cible.com/login \
    --auth-username utilisateur-test \
    --auth-password mot-de-passe-test \
    --auth-submitfield "submit"

# ZAP se connectera avant de scanner
# Permettant de tester les fonctionnalités authentifiées
```

Intégration DAST dans GitLab

```yaml
# Ajout du DAST au pipeline
dast-scan:
  stage: security
  image: owasp/zap2docker-stable
  # Le scan DAST nécessite que l'application soit déployée
  # On l'exécute après le déploiement en staging
  dependencies:
    - deploy-staging
  script:
    - zap-baseline.py
        -t ${STAGING_URL}
        -r zap-report.html
        -x zap-report.xml
  artifacts:
    # Conserver les rapports pour analyse
    paths:
      - zap-report.html
      - zap-report.xml
    expire_in: 1 week
```

Interprétation des Résultats DAST

Niveaux de risque

Niveau	Action
High	Correction immédiate requise
Medium	A corriger avant mise en production
Low	A planifier
Informational	Bonnes pratiques

Faux positifs

Toujours valider manuellement les résultats critiques. ZAP peut reporter des faux positifs qui doivent etre exclus des futurs scans.

Travaux Pratiques - SAST/DAST

Exercice 1 : Analyse SAST

1. Installer SonarQube localement
2. Analyser le projet fourni
3. Interpréter les résultats
4. Proposer des corrections

Travaux Pratiques - SAST/DAST

Exercice 2 : Analyse DAST

1. Lancer ZAP contre l'environnement de formation
2. Effectuer un scan baseline
3. Analyser le rapport
4. Identifier les vrais positifs et faux positifs

Module 3

Sécurité du Chemin d'Exécution

Limiter la Reconnaissance

Objectif

Réduire les informations accessibles aux attaquants durant la phase de reconnaissance.

Actions

```
# Supprimer les bannières de version
# Nginx : ajouter dans nginx.conf
server_tokens off;

# Apache : ajouter dans httpd.conf ou apache2.conf
ServerTokens Prod
ServerSignature Off

# Supprimer les headers révélateurs
# X-Powered-By, X-AspNet-Version, etc.
```

Réduire la Surface d'Attaque

```
# Lister les ports ouverts
netstat -tlnp
# ou
ss -tlnp

# Désactiver les services inutiles
# Exemple : désactiver le serveur FTP s'il n'est pas nécessaire
systemctl stop vsftpd
systemctl disable vsftpd

# Fermer les ports avec le pare-feu
# Autoriser uniquement HTTP, HTTPS, SSH
ufw default deny incoming
ufw default allow outgoing
ufw allow 22/tcp   # SSH
ufw allow 80/tcp   # HTTP
ufw allow 443/tcp  # HTTPS
ufw enable
```

Limiter les Conséquences d'une Compromission

Principe du moindre privilège

```
# Créer un utilisateur dédié pour l'application
# Sans shell de connexion et sans home directory
useradd --system --no-create-home --shell /usr/sbin/nologin appuser

# L'application tourne avec cet utilisateur
# En cas de compromission, l'attaquant a des droits limités

# Configuration dans systemd
# /etc/systemd/system/monapp.service
[Service]
User=appuser
Group=appuser
NoNewPrivileges=true
PrivateTmp=true
ProtectSystem=strict
ProtectHome=true
```

Module 4

Durcissement Système

Politique de Mots de Passe

```
# Configuration PAM pour politique de mots de passe
# Fichier : /etc/security/pwquality.conf

# Longueur minimale
minlen = 14

# Classes de caractères requises
# dcredit : chiffres (digit)
# ucredit : majuscules (uppercase)
# lcredit : minuscules (lowercase)
# ocredit : caractères spéciaux (other)
# Valeur négative = minimum requis
dcredit = -1
ucredit = -1
lcredit = -1
ocredit = -1

# Interdire les mots du dictionnaire
dictcheck = 1
```

Verrouillage des Comptes

```
# Configuration PAM pour verrouillage après échecs
# Fichier : /etc/pam.d/common-auth

# Ajouter au début du fichier
# deny=5 : verrouillage après 5 échecs
# unlock_time=900 : déverrouillage après 15 minutes
# fail_interval=900 : fenetre de comptage de 15 minutes
auth required pam_tally2.so deny=5 unlock_time=900 fail_interval=900

# Vérifier les comptes verrouillés
pam_tally2 --user=username

# Déverrouiller manuellement
pam_tally2 --user=username --reset
```

Configuration SSH Sécurisée

```
# Fichier : /etc/ssh/sshd_config

# Désactiver l'authentification root
PermitRootLogin no

# Utiliser uniquement SSH protocole 2
Protocol 2

# Authentification par clé uniquement
PasswordAuthentication no
PubkeyAuthentication yes

# Limiter les utilisateurs autorisés
AllowUsers admin deployer

# Timeout d'inactivité (5 minutes)
ClientAliveInterval 300
ClientAliveCountMax 0

# Journalisation
LogLevel VERBOSE
```

Pare-feu avec iptables

```
# Script de configuration iptables

# Effacer les règles existantes
iptables -F
iptables -X

# Politique par défaut : tout refuser
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT ACCEPT

# Autoriser le loopback (localhost)
iptables -A INPUT -i lo -j ACCEPT

# Autoriser les connexions établies
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

# Autoriser SSH (port 22)
iptables -A INPUT -p tcp --dport 22 -j ACCEPT

# Autoriser HTTP et HTTPS
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A INPUT -p tcp --dport 443 -j ACCEPT
```

Surveillance du Système de Fichiers

```
# Installation de AIDE (Advanced Intrusion Detection Environment)
apt install aide

# Initialisation de la base de référence
# Crée une empreinte de tous les fichiers surveillés
aide --init

# Copier la base initialisée
mv /var/lib/aide/aide.db.new /var/lib/aide/aide.db

# Vérification manuelle
# Compare l'état actuel avec la base de référence
aide --check

# Automatisation avec cron
# Vérification quotidienne à 2h du matin
0 2 * * * /usr/bin/aide --check | mail -s "AIDE Report" admin@exemple.com
```

Module 5

Durcissement des Applications

Durcissement Nginx

```
# /etc/nginx/nginx.conf

# Worker et connexions
worker_processes auto;
worker_rlimit_nofile 65535;

events {
    worker_connections 65535;
    multi_accept on;
}

http {
    # Masquer la version
    server_tokens off;

    # Limiter la taille des requetes (protection DoS)
    client_body_buffer_size 1K;
    client_header_buffer_size 1K;
    client_max_body_size 1M;
    large_client_header_buffers 2 1K;

    # Timeouts (protection slowloris)
    client_body_timeout 10;
    client_header_timeout 10;
    keepalive_timeout 5 5;
    send_timeout 10;
}
```

Durcissement MySQL

```sql
-- Supprimer les comptes anonymes
DELETE FROM mysql.user WHERE User='';

-- Supprimer l'accès root distant
DELETE FROM mysql.user WHERE User='root' AND Host NOT IN ('localhost', '127.0.0.1', '::1');

-- Supprimer la base de test
DROP DATABASE IF EXISTS test;

-- Créer un utilisateur applicatif avec droits limités
-- L'application n'a pas besoin des droits root
CREATE USER 'appuser'@'localhost' IDENTIFIED BY 'mot-de-passe-fort';
GRANT SELECT, INSERT, UPDATE, DELETE ON appdb.* TO 'appuser'@'localhost';

-- Pas de GRANT, DROP, CREATE pour l'utilisateur applicatif
FLUSH PRIVILEGES;
```

Durcissement PHP

```
; Fichier : /etc/php/8.x/fpm/php.ini

; Désactiver les fonctions dangereuses
; Ces fonctions permettent l'exécution de commandes système
disable_functions = exec,passthru,shell_exec,system,proc_open,popen

; Ne pas exposer PHP
expose_php = Off

; Limiter les inclusions de fichiers distants
allow_url_fopen = Off
allow_url_include = Off

; Limiter les ressources
max_execution_time = 30
max_input_time = 60
memory_limit = 128M
post_max_size = 8M
upload_max_filesize = 2M

; Mode de rapport d'erreurs en production
display_errors = Off
log_errors = On
error_log = /var/log/php/error.log
```

IDS/IPS - Fail2ban

```
# Configuration Fail2ban
# Fichier : /etc/fail2ban/jail.local

[DEFAULT]
# Durée du bannissement (10 minutes)
bantime = 600
# Fenetre d'observation (10 minutes)
findtime = 600
# Nombre d'échecs avant bannissement
maxretry = 5

[sshd]
enabled = true
port = ssh
filter = sshd
logpath = /var/log/auth.log
maxretry = 3

[nginx-http-auth]
enabled = true
filter = nginx-http-auth
logpath = /var/log/nginx/error.log
maxretry = 5
```

Monitoring et Alertes

```yaml
# Configuration Prometheus pour monitoring sécurité
# prometheus.yml

global:
  scrape_interval: 15s

alerting:
  alertmanagers:
    - static_configs:
        - targets: ['localhost:9093']

rule_files:
  - "security_rules.yml"

scrape_configs:
  - job_name: 'node'
    static_configs:
      - targets: ['localhost:9100']

  - job_name: 'nginx'
    static_configs:
      - targets: ['localhost:9113']
```

Règles d'Alerte Sécurité

```yaml
# security_rules.yml pour Prometheus

groups:
- name: security
  rules:
  - alert: HighFailedLogins
    # Alerte si plus de 10 échecs de connexion en 5 minutes
    expr: increase(ssh_failed_logins[5m]) > 10
    for: 1m
    labels:
      severity: warning
    annotations:
      summary: "Tentatives de connexion suspectes"

  - alert: UnauthorizedFileChange
    # Alerte si des fichiers système sont modifiés
    expr: node_filesystem_files_changed > 0
    for: 1m
    labels:
      severity: critical
    annotations:
      summary: "Modification de fichiers détectée"
```

Travaux Pratiques - Durcissement

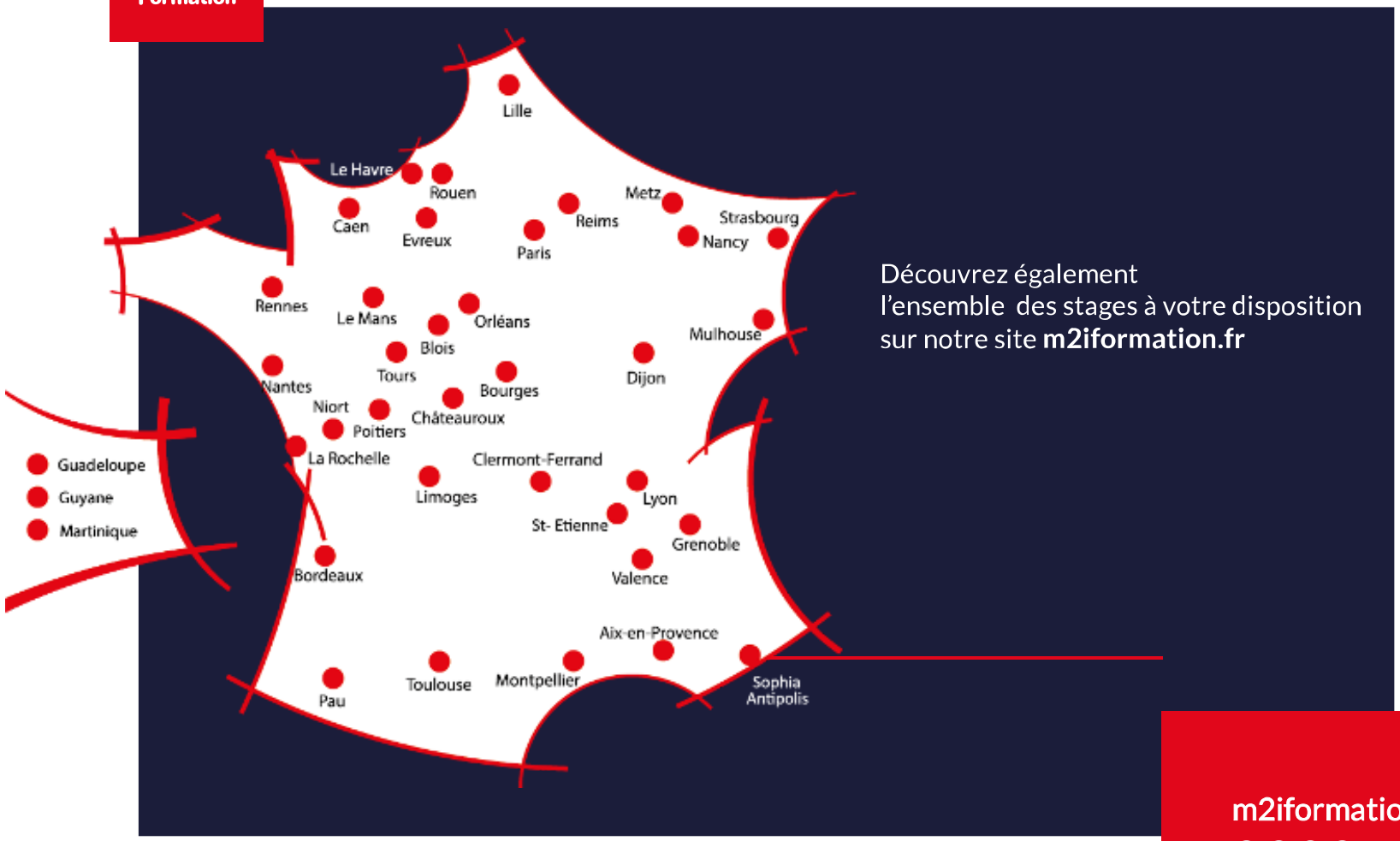
Exercice : Sécuriser l'environnement de formation

1. Configurer le pare-feu
2. Durcir la configuration SSH
3. Sécuriser le serveur web
4. Configurer Fail2ban
5. Mettre en place la surveillance avec AIDE

Travaux Pratiques - Durcissement

Vérification

Utiliser un scanner de vulnérabilités pour valider le durcissement.



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

m2iformation.fr