

Rapport exercice 2 : Test de vulnérabilités avec semgrep

Etape 1 : Intégration du dépôt git à semgrep

J'ai créé un dépôt git content tous les fichiers liés à l'application. Ensuite, j'ai lié le dépôt à semgrep cloud ce qui permet de :

- Analyser automatiquement votre code source.
- Utiliser un ensemble de règles prédéfinies (OWASP Top 10, Best Practices DevSecOps).
- Générer un rapport détaillant les vulnérabilités identifiées (avec des explications pour chaque problème).

Etape 2 : Analyse des vulnérabilités

Après avoir lié mon dépôt git à semgrep, j'ai lancé le scan. Il analyse le dépôt en ligne, génère des alertes catégorisées par High, Medium, et Low, et propose des recommandations de correction. Voici ce que présente semgrep après le scan :

The screenshot displays the Semgrep Cloud interface. On the left, a sidebar contains filters for 'Projects and branches', 'Tags', 'Status' (Open (3)), 'Severity' (High, Medium, Low), 'Confidence' (High, Medium, Low), 'Pro findings only' (toggle), 'Category' (All categories), 'Action' (Monitor, Comment, Block), and 'Rule'. The main panel shows '3 matching findings' with a 'Sort by highest severity' dropdown and 'Analyze (0)' and 'Triage (0)' buttons. The findings are:

- xml-external-entities-unsafe-entity-loader** (Severity: High, 1 finding). Description: The application is using an XML parser that has not been safely configured. This might lead to XML External Entity (XXE) vulnerabilities when parsing user-controlled input. An attacker can include document type definitions (DTDs). Location: 1/test.php:12. Language: PHP.
- xml-external-entities-unsafe-parser-flags** (Severity: High, 1 finding). Description: The application is using an XML parser that has not been safely configured. This might lead to XML External Entity (XXE) vulnerabilities when parsing user-controlled input. An attacker can include document type definitions (DTDs). Location: 1/test.php:12. Language: PHP.
- xml-dtd-allowed** (Severity: High, 1 finding). Description: The application is using an XML parser that has not been safely configured. This might lead to XML External Entity (XXE) vulnerabilities when parsing user-controlled input. An attacker can include document type definitions (DTDs) or... Location: 1/XmlReader_Tests.cs:25. Language: C#.

On remarque la présence de 2 vulnérabilités **highths** dont l'une dans le fichier **test.php** et l'autre dans le fichier **XmlReader_Test.cs**. Plus en détail :

The screenshot shows the Snyk web interface for a project named 'moubiokou8-p...'. The main panel displays a vulnerability titled 'xml-external-entities-unsafe-entity-loader' with a 'High severity' rating. The description explains that the application uses an XML parser not safely configured, leading to XML External Entity (XXE) vulnerabilities. It mentions that attackers can use Document Type Definitions (DTDs) to interact with internal or external hosts, potentially leading to Local File Inclusion (LFI), Remote Code Execution (RCE), and Server-side request forgery (SSRF). The best defense is to have an XML parser that supports disabling DTDs. The interface also shows references to websec.io and the PHP manual. On the right, the 'Activity' section shows a 'New note' and a 'Seen on' event from the 'main' branch on January 30, 2025. At the bottom, there are tabs for 'Pattern' and 'Metadata', and a code editor showing the vulnerable PHP code.

```
1 pattern-sources:
2   - patterns:
3     - pattern-either:
4       - pattern: |
```

```
1 <?php
2 $foo = new DOMDocument();
3 // ok: xml-external-entities-unsafe-entity-loader
4 $foo->loadXML($_GET['xml']);
```

On remarque qu'il s'agit d'une vulnérabilité liée à l'XML External Entity (XXE). L'injection XXE est un type d'attaque contre une application qui analyse les entrées XML. Une telle vulnérabilité peut entraîner une fuite de données, une exécution de code à distance (RCE), Denial of Service (DoS). Pour la corriger il faut donc désactiver les DTDs dans le parser XML et limiter les entités externes :

```
// Désactiver le chargement d'entités externes
libxml_disable_entity_loader(true);

$xml = strlen($_GET['xml']) > 0 ? $_GET['xml'] : '<root><content>No XML fo

$document = new DOMDocument();
//1
$document->xmlStandalone = true;

$document->loadXML($xml, LIBXML_NOENT);
```

Une fois les vulnérabilités corrigées, j'ai relancé le scan, consulté les sections High, Medium, et Low :

