

Sécurisation des applications Web

Activité 1 : Injections

I Objectifs à atteindre.....	1
II Présentation de la notion d'injection.....	1
1 Définition de la notion d'injection.....	1
2 Exemple d'injection SQL (SQLi).....	1
III Présentation des défis.....	2
IV Travaux préparatoires.....	2
V Découverte de la sensibilité SQLi.....	5
VI A vous de jouer.....	6
VII Contre-mesures.....	7

I Objectifs à atteindre

Comprendre la notion d'injection, réaliser les deux défis présentés et comprendre le code source des scripts PHP utilisés (non sécurisé et sécurisé).

II Présentation de la notion d'injection

1 Définition de la notion d'injection

Les défauts d'injection, tels que l'injection SQL ou l'injection LDAP, se produisent lorsque des **données non attendues** sont envoyées à un interpréteur en tant que commande ou requête. Ces données envoyées par l'attaquant peuvent entraîner l'exécution de commandes et ainsi permettre l'accès à des informations confidentielles. L'injection la plus connue reste l'injection SQL (SQLi).

De nombreuses applications web travaillent autour d'un **système de gestion de base de données** (SGBDR) et comportent des **formulaires** qui attendent des données fournies par les utilisateurs. Si le développeur ne travaille que sur les scénarios attendus de saisies des données, il risque de ne pas intégrer de **contrôles de validation** (input validation) ce qui peut entraîner une sensibilité à l'injection.

De nombreux outils permettent d'automatiser la **détection** et l'**exploitation** des vulnérabilités SQLi. On peut citer :

- Burpsuite ;
- Vega ;
- SQLMAP ;
- SQL ninja ;
- Arachni ;
- Script Engine de NMAP...

2 Exemple d'injection SQL (SQLi)

De nombreux exemples d'injections SQL sont disponibles sur Internet. Wikipedia donne un exemple simple qui illustre le fonctionnement d'une requête SQL afin de compromettre le mot de passe d'un utilisateur.

On considère le champ associé au *login* et le champ associé au *mot de passe* et on suppose que l'attaquant connaît le login de la victime (voir les techniques d'ingénierie sociale et de manipulation). La saisie suivante peut être mise en place si le site est sensible aux injections SQL :

- Utilisateur : admin
- mot de passe

L'apostrophe indique la fin de la zone de frappe de l'utilisateur, le code « or 1 » demande au script si 1 est vrai, or c'est toujours le cas, et les doubles tirets indiquent le début d'un commentaire.

La requête SQL pourra ainsi ressembler à ceci :

```
select uid FROM Users where Name = 'admin' AND Password = '' OR 1 -- ;
```

Le script devient ainsi programmé pour vérifier si ce que l'utilisateur saisit est vrai. Comme 1 est vrai, l'attaquant sera connecté en tant qu'administrateur.

Si l'exemple présenté paraît très simple, il existe des injections beaucoup plus compliquées qui sont testées par les outils précédemment cités.

III Présentation des défis

Deux défis sont à relever dans cette première activité.

Défi 1 : Extraction de données

Le but est d'obtenir la liste de tous les utilisateurs.

Défi 2 : Passer outre une authentification

Le but est de s'authentifier à l'aide du compte d'un autre utilisateur.

IV Travaux préparatoires

Il faut commencer par créer un compte sur Mutillidae. Pour cela, cliquer sur le lien *Login/register*.



Puis suivre le lien "Please register here".

Une fois authentifié, une indication apparaît en haut à droite de l'écran. Cette indication permettra de vérifier si certains défis sont réalisés avec succès.



Le niveau de sécurité du code mis en place est indiqué en haut de la page près des informations d'authentification.

Security Level: 0 (Hosed)	Absence de contrôle de sécurité.
Security Level: 1 (Client-side Security) Logged In	Contrôles des informations saisies coté client.
Security Level: 5 (Server-side Security) In	Contrôles des informations saisies coté serveur

Le niveau de sécurité du code mis en œuvre se modifie en cliquant sur le lien *Toggle Security*.

Toggle Security

Chaque niveau de sécurité est disponible sur le même script PHP. Ces scripts sont stockés sur le serveur dans le répertoire `/var/www/html/mutillidae/`.

Pour les deux défis de cette activité, c'est le formulaire d'authentification qui servira de base de travail.

Please sign-in

Username

patrice

Password

●●●●●●●●

Login

Dont have an account? [Please register here](#)

Certains défis nécessitent de connaître les noms des champs de formulaires utilisés. Cette information peut s'obtenir facilement en observant le code source de la page (CTRL + U).

```
<td class="label">Username</td>
<td>
  <input SQLInjectionPoint="1" type="text" name="username" size="20"
    autofocus="autofocus"
    minlength="1" maxlength="15" required="required" />
```

Cette activité peut être réalisée sans utiliser l'outil BurpSuite. Toutefois, une première prise en main est possible : elle permettrait de trouver le nom des champs du formulaire. La démarche de capture d'un premier paquet est décrite dans le document `owasp-mise_en-place` au paragraphe IV-2.3.

Première prise en main de l'outil BurpSuite :

BurpSuite peut être utilisé dans un premier temps pour découvrir le nom des champs d'un formulaire. Les étapes à suivre sont les suivantes :

1. Vérifier que le serveur Mutillidae est démarré (machine d'adresse IP 192.168.0.10).
2. Depuis la machine cliente d'adresse IP 192.168.0.11, accéder à la page d'accueil d'OWASP via le lien suivant : 192.168.0.10/mutillidae
3. Dans les options du navigateur, cliquer sur **Avancé**, puis sur **Réseau** et sur **Paramètres** afin de configurer l'utilisation du proxy BurpSuite. Indiquer **127.0.0.1** comme adresse IP et **8080** comme numéro de port dans la mesure où BurpSuite est installé sur la machine cliente :

Configuration du serveur proxy pour accéder à Internet

☐ Pas de proxy

☐ Détection automatique des paramètres de proxy pour ce réseau

☐ Utiliser les paramètres proxy du système

☒ Configuration manuelle du proxy :

Proxy HTTP :

127.0.0.1

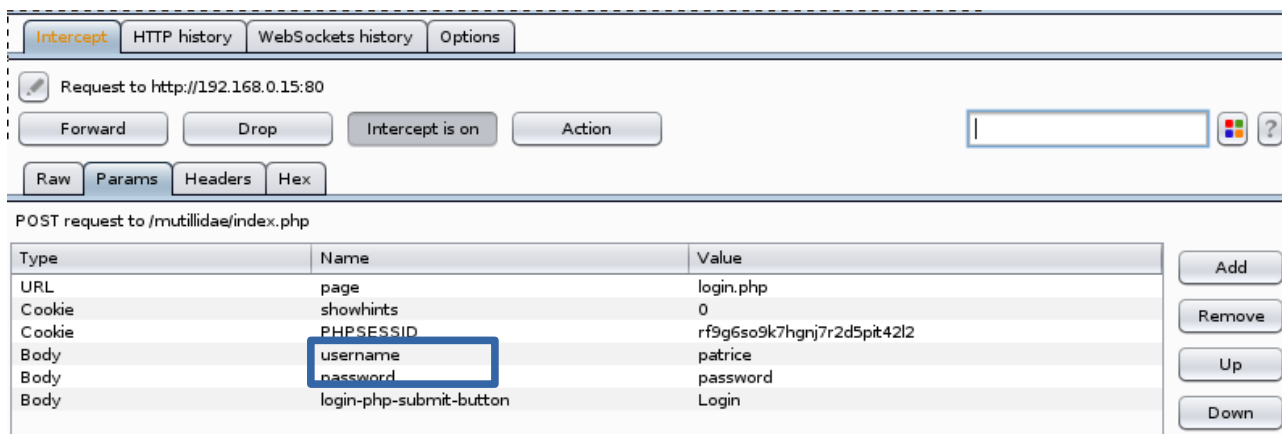
Port :

8080

- Démarrer BurpSuite en suivant le cheminement suivant : **Temporary project => Use Burp defaults**. Aller dans l'onglet **Proxy** puis sur **Intercept**, vérifier que le proxy est désactivé (**Intercept is off**).
- Retourner sur la page d'accueil de Mutillidae et cliquer sur le lien **Login/Register** en haut à gauche.



- Saisir un login dans le champ **Username** et un mot de passe dans le champ **Password** sans cliquer sur le bouton **Login**.
- Dans l'outil BurpSuite, activer le proxy en cliquant sur **Intercept is off**. Le bouton devient **Intercept is on**.
- Revenir sur la page d'authentification de Mutillidae et cliquer sur le bouton **Login**.
- Revenir sur BurpSuite et cliquer sur le bouton **Forward** dans le sous onglet **Intercept** de l'onglet **Proxy**. Le nom des champs est visible dans l'onglet **Params**.



Le mot de passe capturé ici est celui de notre propre compte saisi depuis notre machine cliente. Rien de merveilleux pour le moment, ce qui nous intéresse est le nom des champs utilisés. L'objectif était de se familiariser avec BurpSuite. En effet, le nom des champs peut s'obtenir en observant le code source de la page web (CTRL + U).

Travail à faire 1

- Créer un compte permettant de vous authentifier sur la plate forme.
- Utiliser une méthode de votre choix afin de découvrir les noms des champs *login* et *mot de passe* du formulaire d'authentification.
- Positionner le niveau de sécurité du code à 0 (Hosed).

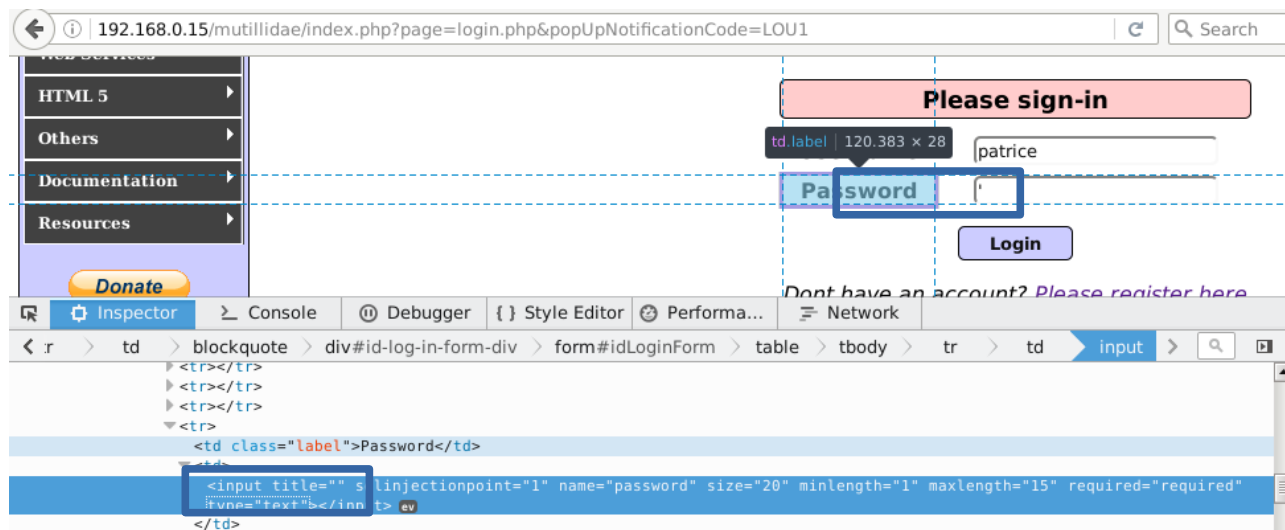
V Découverte de la sensibilité SQLi

Une fois les travaux préparatoires effectués, il est nécessaire de passer à la phase de découverte des vulnérabilités SQLi.

La plateforme Mutillidae offre des pages sensibles à la faille SQLi. Bien évidemment, tous les sites web ne donnent pas de réponses positives aux tests de vulnérabilités. Cela dépend du niveau de sécurité de leur code.

Afin de valider la sensibilité SQLi, aller sur la page permettant de se connecter. Il faut aussi penser à vérifier que le niveau de sécurité sélectionné est 0 (Hosed).

Il suffit alors de saisir une quote dans le champ associé au mot de passe.



Dans cet exemple, l'outil *Web developer* de Firefox a été utilisé afin de rendre visible les données saisies dans le champ associé au mot de passe. Cette manipulation n'est pas indispensable pour relever les défis présentés. Elle sert uniquement à illustrer la saisie de la quote.

Lors du clic sur le bouton *Login*, une erreur apparaît dévoilant la sensibilité SQLi.

Error Message

Failure is always an option	
Line	170
Code	0
File	/var/www/html/mutillidae/classes/MySQLHandler.php
Message	<pre>/var/www/html/mutillidae/classes/MySQLHandler.php on line 165: Error executing query: connect_errno: 0 errno: 1064 error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '''' at line 2 client_info: 5.5.50 host_info: 127.0.0.1 via TCP/IP) Query: SELECT * FROM accounts WHERE username='patrice' AND password='' (0) [Exception]</pre>
Trace	<pre>#0 /var/www/html/mutillidae/classes/MySQLHandler.php(262): MySQLHandler->doExecuteQuery('SELECT * FROM a...') #1 /var/www/html/mutillidae /classes/SQLQueryHandler.php(350): MySQLHandler->executeQuery('SELECT * FROM a...') #2 /var/www/html/mutillidae/user-info.php(191): SQLQueryHandler->getUserAccount('patrice', '') #3 /var/www/html/mutillidae/index.php(615): require_once('/var/www/html/m...') #4 {main}</pre>
Diagnostic Information	Error attempting to display user information

[Click here to reset the DB](#)

Ce message d'erreur est particulièrement instructif pour une personne malveillante.

Travail à faire 2 Tester une détection manuelle de la sensibilité SQLi.

VI A vous de jouer

Le but est de relever les deux défis présentés dans le paragraphe 3 de cette activité. Une recherche sur Internet permet d'obtenir de nombreux exemples d'injections à tester.

Premier défi : Extraction de données

Le but est d'obtenir la liste de tous les utilisateurs. Pour lancer le défi, il faut suivre le cheminement suivant : OWASP 2017 => A1 – Injection (SQL) => SQLi – Extract Data => User Info (SQL).



Le nom du script PHP sur le serveur est *user-info.php*.

En temps normal, cette page permet d'afficher le détail des informations d'un compte utilisateur.

Please enter username and password to view account details

Name

Password

View Account Details

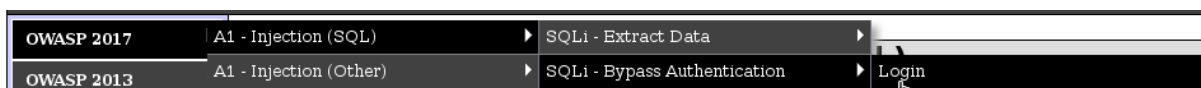
Dont have an account? [Please register here](#)

Results for "patrice".1 records found.

Username=patrice
Password=password
Signature=

Deuxième défi : Passer outre une authentification

Le but est de s'authentifier à l'aide du compte d'un autre utilisateur de votre choix. Pour lancer le défi, il faut suivre le cheminement suivant : OWASP 2017 => A1 – Injection (SQL) => SQLi – Bypass Authentication => Login.



Le nom du script PHP sur le serveur est *login.php*. En temps normal, cette page offre un formulaire d'authentification.

Travail à faire 3

Q1. Réaliser les deux défis présentés en testant les injections suivantes :

- ❖ 'or ('a' = 'a') or '
- ❖ ' or username='admin

Q2. Reporter sur votre documentation les injections testées.

VII Contre-mesures

La contre-mesure à ces vulnérabilités passe par la mise en place d'un codage sécurisé. En modifiant le niveau de sécurité les tests précédents doivent échouer.

Travail à faire 4

- Q1.** Modifier le niveau de sécurité et vérifier que les tests d'exploitation des failles SQLi échouent.

En utilisant le fichier user-info.php, répondre aux questions suivantes.

- Q2.** Quel niveau de sécurité est nécessaire pour protéger contre cette attaque ?
Q3. A ce niveau, quels sont les contrôles réalisés ?
Q4. Est-ce que le contrôle HTML aurait suffi à protéger contre cette injection SQL.
Q5. Comment la validation javascript est-elle déclenchée ?
Q6. Quels sont les contrôles réalisés par la validation Javascript ?

Prolongement possible :

Reprendre une application existante (TP/PPE) et mettre en place le niveau de sécurité 5.