

Cybersécurité

C2 - Pourquoi les
injections SQL
existent-elles toujours ?

Sommaire

- 1) Mise en contexte
- 2) Attaques réelles
- 3) Présentation d'un outil
- 4) Exemple d'utilisation
- 5) Solutions pratiques





Mise en contexte

Encore en tête du classement TOP 10 de l'OWASP, les injections SQL (SQLi) représentent, 25 ans après leur apparition au grand public, toujours une menace bien réelle sur les applications Web.


Comment expliquer leur persistance alors que les technologies ne cessent d'évoluer: l'utilisation de framework ORM (object-relational mapping, interface entre l'application et la base de données), PHP implémente PDO et les requêtes préparées depuis 2004.

Injection SQL: Attaques réelles

En 2002, Jeremiah Jacks dérobe les coordonnées et identifiants bancaires de 200 000 personnes grâce à une injection SQL sur le site de Guess.com.

En 2003, il réussit à trouver une faille SQL en moins d'une minute permettant d'accéder à 500 000 numéros de cartes de crédit.






Présentation de Burp Suite

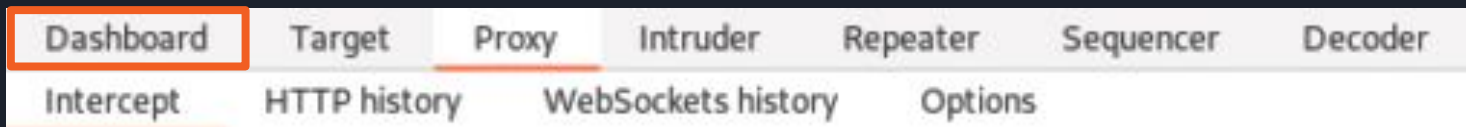
Pour réaliser ou pour simuler une attaque de ce genre, le volume d'informations à récupérer étant élevé, il est préférable d'utiliser des logiciels qui travaillent plus rapidement que manuellement.

Un des logiciels les plus utilisés dans le monde de la cybersécurité est Burp Suite. Il permet entre autre d'effectuer des tests de pénétration.






Présentation de Burp Suite

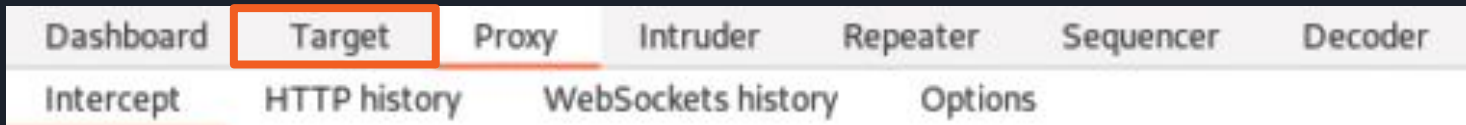


Le tableau de bord affiche toutes les tâches en cours et les logs de connexion en temps réel.

D'ici, on peut lancer un scan ou ajouter de nouvelles tâches.




Présentation de Burp Suite

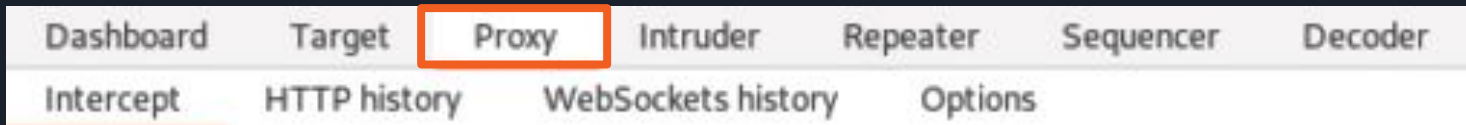


Cet onglet permet d'obtenir l'arborescence du site web (les fichiers) comme l'inspecteur dans un navigateur.

Il permet aussi de voir les requêtes et les réponses envoyées depuis chaque page.



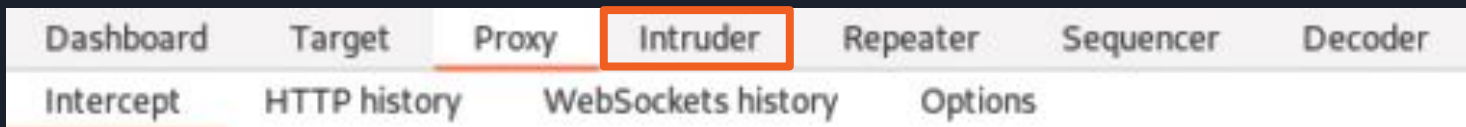
Présentation de Burp Suite



Cet onglet permet d'activer le proxy qui s'occupe de récupérer les requêtes HTTP des sites web. C'est ici que nous pouvons contrôler le comportement du site pas à pas.

Pour intercepter les requêtes HTTPS, il est nécessaire d'importer un certificat dans le navigateur web.

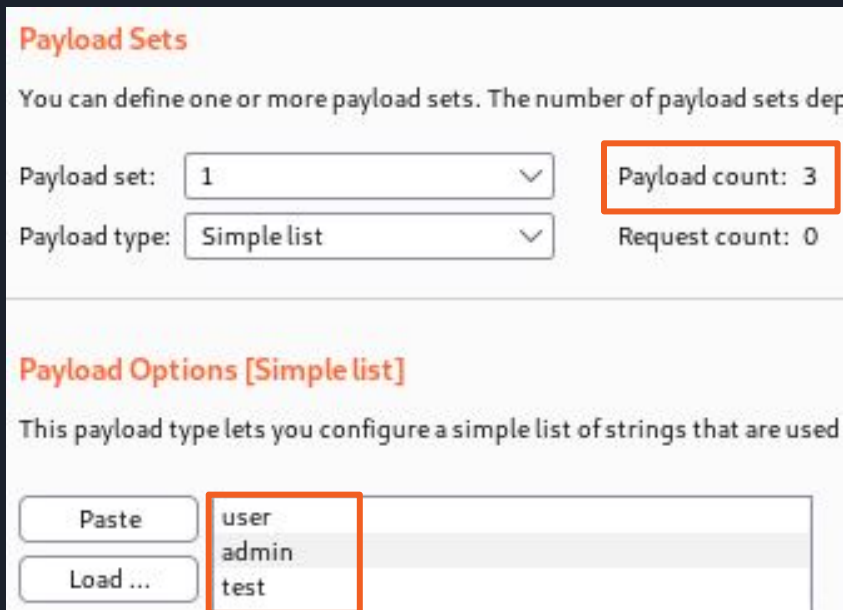
Présentation de Burp Suite



L'introduction de code se fait dans cette partie. On peut simuler ou effectuer une attaque en choisissant son type et ses paramètres.



Présentation de Burp Suite



Payload Sets

You can define one or more payload sets. The number of payload sets depends on the number of requests in the selected scope.

Payload set: 1 ▼ **Payload count: 3**

Payload type: Simple list ▼ Request count: 0

Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used to replace the payload.

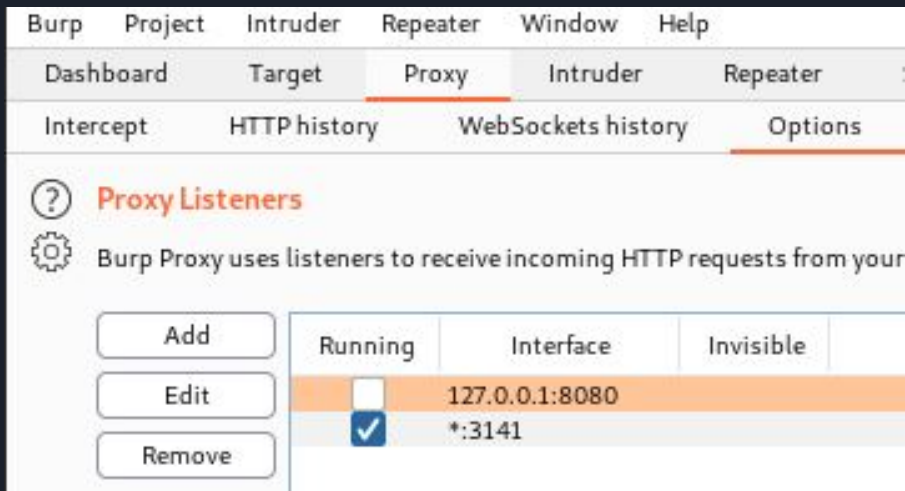
Paste Load ...

user
admin
test

On peut aussi simuler des attaques de type ***brute force*** en saisissant plusieurs identifiants différents par exemple.

Exemple d'utilisation de Burp Suite

On active un *listener* sur une interface externe (toutes les adresses du port 3141) et on désactive l'interception des requêtes.



Intercept is off

Exemple d'utilisation de Burp Suite

On écoute le trafic intercepté qui est visible dans:

- **Target-Site Map:** arborescence du site
- **Proxy-History:** ordre chronologique





Finalité de Burp Suite

Burp Suite est une suite complète d'outils de cybersécurité. Elle existe en plusieurs versions dont la Community et la Pro.

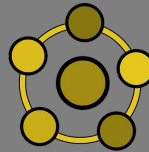
Avec de tels outils, il est normal que les SQLi persistent.

La récupération des données sensibles est donc facile quelque soit la méthode d'envoi utilisée (POST ou GET).

Voyons quelles sont les couches de sécurité supplémentaire à ajouter afin de pallier à cette faille.

Schématisation des couches

Interrogation des données par l'utilisateur
Fonctionnalités métier
Base de données



JSHTML, CSS



Interface utilisateur (partie HTML)

Sécuriser au maximum le formulaire en limitant les actions et les saisies de l'utilisateur:

- Limiter le nombre de caractères (minlength, maxlength)
- Définir le bon type de champs (text, email, password)
- Ajouter les attributs nécessaires (required, pattern)



Interface utilisateur (partie HTML)

Exemple d'utilisation de l'attribut pattern:

```
<label>Veuillez saisir votre numéro de téléphone  
  (<input name="tel1" type="tel" pattern="[0-9]{10}" size="5"/>)-
```

pattern permet de préciser une expression régulière.
Cela veut dire que la valeur du champs devra respecter la
contrainte du formulaire.



Interface utilisateur (partie HTML)

Autres exemples de l'attribut pattern:

```
<div class="c100">
  <label for="prenom">Prénom : </label>
  <input type="text" id="prenom" name="prenom"
    required pattern="^[A-Za-z ' -]+$" maxlength="20">
</div>
<div class="c100">
  <label for="mail">Email : </label>
  <input type="email" id="mail" name="mail"
    required pattern="^[A-Za-z]+@{1}[A-Za-z]+\.{1}[A-Za-z]{2,}$">
</div>
```



Interactions locales (partie JS)

Créer des fonctions de vérifications des champs du formulaire:

- format valide de l'adresse mail
- caractères autorisés ou non
- limiter l'expérience utilisateur quand nécessaire
 - interdire la modification d'un champs
 - interdire l'envoi d'un champs vide



Interactions locales (partie JS)

Exemple de fonction JS:

```
function onSubmitOfLoginForm(/*HTMLFormElement*/ theForm){  
    var lUnsafeCharacters = /[`~!@#$%^&*()-_+=\[\]\{\}\|\;'\:".,/<>?]/;  
    if(!ValidateInput == "TRUE"){  
        if (theForm.username.value.length > 15 ||  
            theForm.password.value.length > 15){  
            alert('Username too long. We dont want to allow too many  
characters.\n\n');  
            return false;  
        }  
    }  
}
```



Interactions locales (partie JS)

Exemple de fonction JS (suite):

```
        if (theForm.username.value.search(\\UnsafeCharacters) > -1 ||  
            theForm.password.value.search(\\UnsafeCharacters) > -1){  
            alert('Dangerous characters detected.');
```

```
            return false;  
        }// end if  
    }// end if(\\ValidateInput)  
    return true;  
}
```

Interrogation serveur

- Utilisation de PDO avec PHP
 - Requêtes préparées puis exécutées
- Créer des fonctions de vérification des données

```
        'role_id'      => $role_details['id'],
        'resource_id' => $resource_details['id'],
    );
    if ( $this->rule_exists( $resource_details['id'], $role_details['id'] ) ) {
        if ( $access == false ) {
            // Remove the rule as there is currently no need for it
            $details['access'] = !$access;
            $this->_sql->delete( 'acl_rules', $details );
        } else {
            // Update the rule with the new access value
            $this->_sql->update( 'acl_rules', array( 'access' => $access ) );
        }
    }
    foreach( $this->rules as $key=>$rule ) {
        if ( $details['role_id'] == $rule['role_id'] && $details['resource_id'] == $rule['resource_id'] ) {
            if ( $access == false ) {
                unset( $this->rules[ $key ] );
            } else {
                $this->rules[ $key ]['access'] = $access;
            }
        }
    }
}
```



Interrogation serveur

Exemple de fonction PHP utilisant une requête SQL:

```
// Get input
$id = $_REQUEST[ 'id' ];

// Check database
$query = "SELECT first_name, last_name FROM users WHERE user_id =
'$id'";
```

Code vulnérable

L'entrée de l'utilisateur est simplement concaténée à la commande sans être vérifiée au préalable

- cela permet de passer n'importe quel caractère

```
// Feedback for end user
echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname:
{$last}</pre>";
}
```



Interrogation serveur

Bonne pratique:

- Utilisation des requêtes préparées avec PDO
 - elles sont émulées par PDO

Exemple 1: Sécurisation de la connexion à la BDD

```
$connexion = new PDO('mysql:dbname=dbtest;host=127.0.0.1;charset=utf8',  
'utilisateur', 'motDePasse');  
$connexion->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);  
//La ligne suivante indique à PDO de bien générer une erreur fatale si un  
problème survient. On peut détecter et gérer les problèmes de cette manière.  
$connexion->setAttribute(PDO::ATTR_ERRMODE,  
PDO::ERRMODE_EXCEPTION);
```



Interrogation serveur

Bonne pratique:

Exemple 2: Préparation d'une requête

```
try {  
    // first connect to database with the PDO object.  
    $db = new \PDO("mysql:Host=localhost;dbname=xx;charset=utf8", "xx", "xx", [  
        PDO::ATTR_EMULATE_PREPARES => false,  
        PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION  
    ]);  
} catch(\PDOException $e){  
    // if connection fails, show PDO error.  
    echo "Error connecting to mysql: " . $e->getMessage();  
}
```




Interrogation serveur

Bonne pratique:

Exemple 2: Exécution de la requête

```
if($_POST && isset($_POST['color'])){  
    // preparing a statement  
    $stmt = $db->prepare("SELECT id, name, color FROM Cars WHERE color = ?");  
  
    // execute/run the statement.  
    $stmt->execute(array($_POST['color']));  
  
    // fetch the result.  
    $cars = $stmt->fetchAll(PDO::FETCH_ASSOC);  
    var_dump($cars);  
}
```



Interrogation serveur

Bonne pratique:

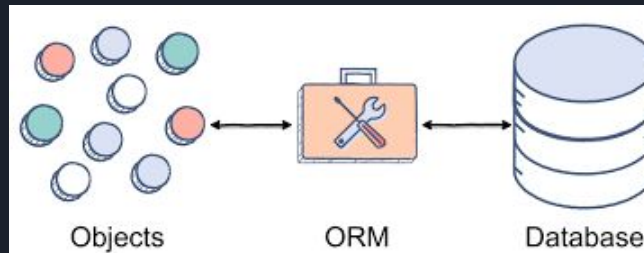
Exemple 3: Insertion dans une table

```
$stmt="INSERT into user(name,email,password) VALUES(:name,:email,:password)";
try{
    $pstmt=$dbh->prepare($stmt);//$dbh database for executing mysql query
    $pstmt->bindParam(':name',$name,PDO::PARAM_STR);
    $pstmt->bindParam(':email',$email,PDO::PARAM_STR);
    $pstmt->bindParam(':password',$password,PDO::PARAM_STR);
    $status=$pstmt->execute();
    if($status){ //next line of code
    }
}catch(PDOException $pdo){
    echo $pdo->getMessage();
} //bindParam ou bindValue: associe une valeur à un paramètre (cf php.net)
```

ORM (Object-Relational-Mapping)

Méthode d'accès aux données d'une BDD qui utilise une syntaxe orientée objet au lieu d'utiliser le langage SQL

- Créer les classes objet pour que la BDD corresponde aux objets
 - Utilisation de framework (exemples: Symfony, Laravel)
 - Architecture MVC (Model View Controller)





Référence

<https://www.digitemis.com/pourquoi-les-injections-sql-existent-elles-toujours/>

```
        'role_id'      => $role_details['id'],
        'resource_id' => $resource_details['id'],
    );
    if ( $this->rule_exists( $resource_details['id'], $role_details['id'] ) ) {
        if ( $access == false ) {
            // Remove the rule as there is currently no need for it
            $details['access'] = !$access;
            $this->_sql->delete( 'acl_rules', $details );
        } else {
            // Update the rule with the new access value
            $this->_sql->update( 'acl_rules', array( 'access' => $access ) );
        }
    }
    foreach( $this->rules as $key=>$rule ) {
        if ( $details['role_id'] == $rule['role_id'] && $details['resource_id'] == $rule['resource_id'] ) {
            if ( $access == false ) {
                unset( $this->rules[ $key ] );
            } else {
                $this->rules[ $key ]['access'] = $access;
            }
        }
    }
}
```