SQL Injection Attack Lab

The development of this document was partially funded by the National Science Foundation under Award No. 1303306 and 1318814. This lab was imported into the Labtainer framework by the Naval Postgraduate School, Center for Cybersecurity and Cyber Operations under National Science Foundation Aware No. 1438893.

Présentation

L'injection SQL est une technique d'injection de code qui exploite les vulnérabilités de l'interface entre les applications Web et les serveurs de base de données. La vulnérabilité est présente lorsque les entrées de l'utilisateur ne sont pas correctement vérifiées dans les applications Web avant d'être envoyées aux serveurs de base de données principaux.

De nombreuses applications Web prennent les entrées des utilisateurs, puis utilisent ces entrées pour construire des requêtes SQL, afin que les applications Web puissent obtenir des informations de la base de données. Les applications Web utilisent également des requêtes SQL pour stocker des informations dans la base de données. Ce sont des pratiques courantes dans le développement d'applications Web. Lorsque les requêtes SQL ne sont pas soigneusement construites, des vulnérabilités d'injection SQL peuvent se produire. L'attaque par injection SQL est l'une des attaques les plus courantes contre les applications Web.

Dans cet atelier, nous avons créé une application Web vulnérable aux attaques par injection SQL. Notre application Web inclut les erreurs courantes commises par de nombreux développeurs Web. L'objectif est de trouver des moyens d'exploiter les vulnérabilités d'injection SQL, de démontrer les dommages qui peuvent être causés par l'attaque et de maîtriser les techniques qui peuvent aider à se défendre contre ce type d'attaques.

Démarrer le laboratoire

Le laboratoire est lancé à partir du répertoire de travail labtainer sur votre hôte sur votre hôte ou votre machine virtuelle Linux. Exécutez la commande:

```
labtainer sql-inject
```

Les terminaux virtuels résultants comprennent : un terminal (shell bash) connecté à un ordinateur **client** "**client**" et un terminal (shell bash) connecté à un **serveur « web-server »**. Un navigateur Firefox démarrera également et s'exécutera sur le composant client.

Remarque pour les instructeurs

Si l'instructeur prévoit d'organiser des sessions de laboratoire pour ce laboratoire, nous suggérons que les documents de base suivants soient couverts dans les sessions de laboratoire :

1. Comment utiliser la machine virtuelle et les Labtainers, le navigateur Web Firefox et les outils Web Developer / Network.
2. Brève introduction de SQL. Il suffit de couvrir la structure de base des instructions SELECT, UPDATE et INSERT. Un didacticiel SQL en ligne utile est disponible sur http://www.w3schools.com/sql/.
3. Comment faire fonctionner la base de données MySQL (uniquement les bases).
4. Brève introduction de PHP. Il suffit de couvrir les bases. Les étudiants ayant une formation en C/C++, Java ou autre langage devraient être capables d'apprendre ce langage de script assez rapidement.

Tâches

Nous avons créé une application Web et l'hébergeons sur www.SEEDLabSQLInjection.com. Cette application Web est une simple application de gestion des employés. Les employés peuvent consulter et mettre à jour leurs informations personnelles dans la base de données via cette application Web. Il y a principalement deux rôles dans cette application Web : Administrateur est un rôle privilégié et peut gérer les informations de profil de chaque employé ; L'employé est un rôle normal et peut afficher ou mettre à jour ses propres informations de profil. Toutes les informations sur les employés sont décrites dans le tableau suivant.

User	Emploee ID	Password	Salary	Birthday	SSN
Admin	99999	seedadmin	400000	3/5	43254314
Alice	10000	seedalice	30000	9/20	10211002
Boby	20000	seedboby	50000	4/20	10213352
Ryan	30000	seedryan	90000	4/10	32193525
Samy	40000	seedsamy	40000	1/11	32111111
Ted	50000	seedted	110000	11/3	24343244

Tache 1: Console MySQL

L'objectif de cette tâche est de se familiariser avec les commandes SQL en jouant avec la base de données fournie. Nous avons créé une base de données appelée Users, qui contient une table appelée Credential ; la table stocke les informations personnelles (par exemple, eid, mot de passe, salaire, ssn, etc.) de chaque employé. L'administrateur est autorisé à modifier les informations de profil de tous les employés, mais chaque employé ne peut modifier que ses propres informations. Dans cette tâche, vous devez jouer avec la base de données pour vous familiariser avec les requêtes SQL.

MySQL est un système de gestion de base de données relationnelle open source. Nous avons déjà configuré MySQL dans le composant « serveur ». Le nom d'utilisateur est root et le mot de passe est *seedubuntu*.

- Veuillez vous connecter à la console MySQL dans le terminal virtuel du serveur à l'aide de la commande suivante :

```
$ mysql -u root -pseedubuntu
```

Après la connexion, vous pouvez créer une nouvelle base de données ou en charger une existante.

- Comme nous avons déjà créé pour vous la base de données Users, il vous suffit de charger cette base existante à l'aide de la commande suivante :

```
mysql> use Users;
```

Vous allez afficher les tables présentes dans la base de données Users,

- Utiliser la commande suivante pour afficher toutes les tables de la base de données sélectionnée.

```
mysql> show tables;
```

Tache 2 : Attaque par injection SQL sur l'instruction SELECT

L'injection SQL est essentiellement une technique par laquelle les attaquants peuvent exécuter leurs propres instructions SQL malveillantes, généralement appelées charge utile malveillante (malicious payload). Grâce aux instructions SQL malveillantes, les attaquants peuvent voler des informations dans la base de données de la victime ; pire encore, ils peuvent être en mesure d'apporter des modifications à la base de données. Notre application Web de gestion des employés présente des vulnérabilités d'injection SQL, qui imitent les erreurs fréquemment commises par les développeurs. Le navigateur démarre à la page d'entrée de notre application Web à l'adresse www.SEEDLabSQLInjection.com, où il vous sera demandé de fournir l'ID de l'employé et le

mot de passe pour vous connecter. La page de connexion est illustrée à la Figure 1. L'authentification est basée sur l'ID de l'employé et Mot de passe, de sorte que seuls les employés qui connaissent leurs identifiants et mots de passe sont autorisés à afficher/mettre à jour les informations de leur profil. Votre travail, en tant qu'attaquant, consiste à vous connecter à l'application sans connaître les informations d'identification d'un employé.

Figure 1: La Page de connexion

Pour vous aider à démarrer cette tâche, nous expliquons comment l'authentification est implémentée dans notre application Web. Le code PHP unsafe_credential.php, situé dans le dossier /var/www/seedlabsqlinjection.com/public_html est utilisé pour effectuer l'authentification des utilisateurs. L'extrait de code suivant montre comment les utilisateurs sont authentifiés.

```
$conn = getDB();
$sql = "SELECT id, name, eid, salary, birth, ssn, phonenumber,
          address, email, nickname, Password
    FROM credential
    WHERE eid= '$input_eid' and password='$input_pwd'";
$result = $conn->query($sql))

// The following is pseudo code
if(name=='admin'){
     return All employees information.
} else if(name!=NULL){
     return employee information.
} else {
     authentication fails.
}
```

L'instruction SQL ci-dessus sélectionne les informations personnelles des employés telles que l'identifiant, le nom, le salaire, le ssn, etc. dans la table des informations d'identification. Les variables input eid et input pwd contiennent les chaînes saisies par les utilisateurs dans la page de connexion. Fondamentalement, le programme vérifie si un enregistrement correspond à l'ID et au mot de passe de l'employé ; s'il y a une correspondance, l'utilisateur est authentifié avec succès et reçoit les informations correspondantes sur l'employé. S'il n'y a pas de correspondance, l'authentification échoue.

Tâche 2.1 : attaque par injection SQL à partir d'une page Web.

Votre tâche consiste à vous connecter à l'application Web en tant qu'administrateur à partir de la page de connexion, afin que vous puissiez voir les informations de tous les employés. Nous supposons que vous connaissez le nom de compte de l'administrateur qui est admin, mais vous ne connaissez pas l'ID ou le mot de passe. Vous devez décider quoi taper dans les champs ID de l'employé et Mot de passe pour réussir l'attaque.

Task 2.2: attaque par injection SQL à partir de la ligne de commandes.

Votre tâche consiste à répéter la tâche 2.1, mais vous devez le faire sans utiliser la page Web. Dans le terminal virtuel client, vous pouvez utiliser des outils de ligne de commande, tels que curl, qui peuvent envoyer des requêtes HTTP. Une chose qui mérite d'être mentionnée est que si vous souhaitez inclure plusieurs paramètres dans les requêtes HTTP, vous devez placer l'URL et les paramètres entre une paire de guillemets simples ; sinon, les caractères spéciaux utilisés pour séparer les paramètres (tels que &) seront interprétés par le programme shell, modifiant la signification de la commande. L'exemple suivant montre comment envoyer une requête HTTP GET à notre application Web, avec deux paramètres (SUID et Mot de passe) attachés :

```
curl 'www.SeedLabSQLInjection.com/index.php?SUID=10000&Password=111'
```

Si vous devez inclure des caractères spéciaux dans les champs SUID et Mot de passe, vous devez les encoder correctement, sinon ils peuvent modifier le sens de vos demandes. Si vous souhaitez inclure des guillemets simples dans ces champs, vous devez utiliser %27 à la place ; si vous souhaitez inclure un espace blanc, vous devez utiliser %20. Dans cette tâche, vous devez gérer l'encodage HTTP lors de l'envoi de requêtes à l'aide de curl.

Tâche 2.3 : ajouter une nouvelle instruction SQL.

Dans les deux attaques ci-dessus, nous ne pouvons voler que des informations de la base de données ; ce sera mieux si nous pouvons modifier la base de données en utilisant la même vulnérabilité dans la page de connexion. Une idée consiste à utiliser l'attaque par injection SQL pour transformer une instruction SQL en deux, la seconde étant l'instruction de mise à jour ou de suppression. En SQL, le point-virgule (;) est utilisé pour séparer deux instructions SQL. Veuillez décrire comment vous pouvez utiliser la page de connexion pour que le serveur exécute deux instructions SQL. Essayez l'attaque pour supprimer un enregistrement de la base de données et décrivez votre observation.

Tâche 3 : Attaque par injection SQL sur l'instruction UPDATE

Si une vulnérabilité d'injection SQL se produit dans une instruction UPDATE, les dommages seront plus graves, car les attaquants peuvent utiliser la vulnérabilité pour modifier les bases de données. Dans notre application de gestion des employés, il existe une page Modifier le profil (Figure 2) qui permet aux employés de mettre à jour les informations de leur profil, y compris le surnom, l'e-mail, l'adresse, le numéro de téléphone et le mot de passe. Pour accéder à cette page, les employés doivent d'abord se connecter.

Lorsque les employés mettent à jour leurs informations via la page Modifier le profil, la requête SQL UPDATE suivante sera exécutée. Le code PHP implémenté dans le fichier unsafe_edit.php est utilisé pour mettre à jour les informations de profil de l'employé. Le fichier PHP se trouve dans le répertoire /var/www/seedlabsqlinjection.com/public_html.

```
$conn = getDB();
$sql = "UPDATE credential SET nickname='$nickname',
          email='$email',
          address='$address',
          phonenumber='$phonenumber',
          Password='$pwd'
          WHERE id= '$input_id' ";
$conn->query($sql))
```

Figure 2: Modifier le Profil

Tâche 3.1 : attaque par injection SQL sur l'instruction UPDATE — modifier le salaire.

Comme indiqué dans la page **Modifier le profil**, les employés peuvent uniquement mettre à jour leurs surnoms, e-mails, adresses, numéros de téléphone et mots de passe ; ils ne sont pas autorisés à modifier leurs salaires. Seul l'administrateur est autorisé à modifier les salaires. Si vous êtes un employé malveillant (par exemple Alice), votre objectif dans cette tâche est d'augmenter votre propre salaire via cette page **Modifier le profil**. Nous supposons que vous savez que les salaires sont stockés dans une colonne appelée salary.

Tâche 3.2 : attaque par injection SQL sur l'instruction UPDATE : modifiez le mot de passe d'autres personnes.

En utilisant la même vulnérabilité dans la déclaration UPDATE ci-dessus, les employés malveillants peuvent également modifier les données d'autres personnes. L'objectif de cette tâche est de modifier le mot de passe d'un autre employé, puis de démontrer que vous pouvez vous connecter avec succès au compte de la victime en utilisant le nouveau mot de passe. L'hypothèse ici est que vous connaissez déjà le nom de l'employé (par exemple Ryan) que vous voulez attaquer. Une chose à mentionner ici est que la base de données stocke la valeur de hachage des mots de passe au lieu de la chaîne de mot de passe en texte brut. Vous pouvez à nouveau regarder le code unsafe_edit.php pour voir comment le mot de passe est stocké. Il utilise la fonction de hachage SHA1 pour générer la valeur de hachage du mot de passe.

Pour vous assurer que votre chaîne d'injection ne contient aucune erreur de syntaxe, vous pouvez tester votre chaîne d'injection sur la console MySQL avant de lancer la véritable attaque sur notre application web.

Tâche 4 : Contre-mesure — instruction préparée

Le problème fondamental de la vulnérabilité d'injection SQL est l'impossibilité de séparer le code des données. Lors de la construction d'une instruction SQL, le programme (par exemple, un programme PHP) sait quelle partie constitue des données et quelle partie du code. Malheureusement, lorsque l'instruction SQL est envoyée à la base de données, le délimiteur a disparu ; les limites que l'interpréteur SQL voit peuvent être différentes des limites d'origine définies par les développeurs. Pour résoudre ce problème, il est important de s'assurer que la vue des limites est cohérente dans le code côté serveur et dans la base de données. Le moyen le plus sûr consiste à utiliser une instruction préparée.

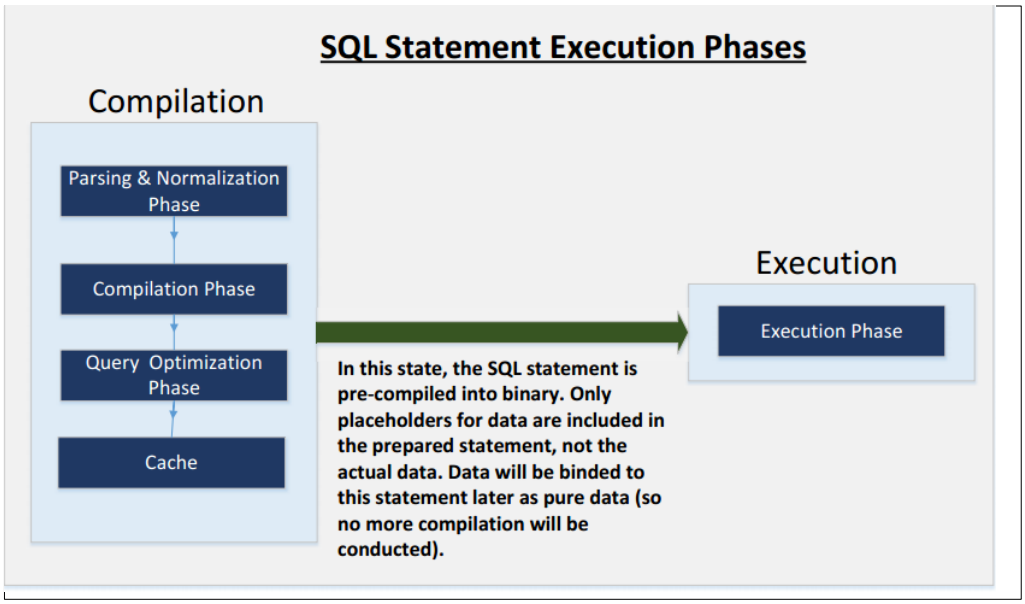

Figure 3: Workflow de l'instruction préparée

Pour comprendre comment l'instruction préparée empêche l'injection SQL, nous devons comprendre ce qui se passe lorsque le serveur SQL reçoit une requête. Le flux de travail de haut niveau de la façon dont les requêtes sont exécutées est illustré à la figure 3. Dans l'étape de compilation, les requêtes passent d'abord par la phase d'analyse et de normalisation, où une requête est vérifiée par rapport à la syntaxe et à la sémantique. La phase suivante est la phase de compilation où les mots-clés (par exemple, SELECT, FROM, UPDATE, etc.) sont convertis dans un format compréhensible par les machines. Fondamentalement, dans cette phase, la requête est interprétée. Dans la phase d'optimisation de la requête, le nombre de plans différents est pris en compte pour exécuter la requête, parmi lesquels le plan le mieux optimisé est choisi. Le plan choisi est stocké dans le cache, donc chaque fois que la prochaine requête arrive, elle sera vérifiée par rapport au contenu du cache ; s'il est déjà présent dans le cache, les phases d'analyse, de compilation et d'optimisation des requêtes seront ignorées. La requête compilée est ensuite passée à la phase d'exécution où elle est réellement exécutée.

L'instruction préparée apparaît après la compilation mais avant l'étape d'exécution. Une instruction préparée passera par l'étape de compilation et sera transformée en une requête précompilée avec des espaces réservés vides pour les données. Pour exécuter cette requête précompilée, des données doivent être fournies, mais ces données ne passeront pas par l'étape de compilation ; à la place, ils sont directement connectés à la requête précompilée et envoyés au moteur d'exécution. Par conséquent, même s'il y a du code SQL à l'intérieur des données, sans passer par l'étape de compilation, le code sera simplement traité comme faisant partie des données, sans aucune signification particulière. C'est ainsi que l'instruction préparée empêche les attaques par injection SQL.

Voici un exemple de la façon d'écrire une instruction préparée en PHP. Nous utilisons une instruction SELECT dans l'exemple suivant. Nous montrons comment utiliser une instruction préparée pour réécrire le code vulnérable aux attaques par injection SQL.

```
$conn = getDB();
$sql = "SELECT name, local, gender
     FROM USER_TABLE
     WHERE id = $id AND password ='$pwd' ";
$result = $conn->query($sql))
```

Le code ci-dessus est vulnérable aux attaques par injection SQL. Il peut être réécrit comme suit

```
$conn = getDB();
     $stmt = $conn->prepare("SELECT name, local, gender
     FROM USER_TABLE
```

```
WHERE id = ? and password = ? ");
// Bind parameters to the query
$stmt->bind_param("is", $id, $pwd);
$stmt->execute();
$stmt->bind_result($bind_name, $bind_local, $bind_gender);
$stmt->fetch();
```

En utilisant le mécanisme d'instruction préparé, nous divisons le processus d'envoi d'une instruction SQL à la base de données en deux étapes. La première étape consiste à envoyer uniquement la partie du code, c'est-à-dire une instruction SQL sans les données réelles. C'est l'étape de préparation. Comme nous pouvons le voir dans l'extrait de code ci-dessus, les données réelles sont remplacées par des points d'interrogation (?). Après cette étape, nous envoyons ensuite les données à la base de données en utilisant bind_param(). La base de données traitera tout ce qui est envoyé à cette étape uniquement comme des données, et non plus comme du code. Il lie les données aux points d'interrogation correspondants de l'instruction préparée. Dans la méthode bind param(), le premier argument "is" indique les types des paramètres : "i" signifie que les données dans $id ont le type entier, et "s" signifie que les données dans $pwd ont la chaîne taper.

Pour cette tâche, veuillez utiliser le mécanisme d'instruction préparé pour corriger les vulnérabilités d'injection SQL que vous avez exploitées dans les tâches précédentes. Ensuite, vérifiez si vous pouvez toujours exploiter la vulnérabilité ou non.

Des lignes directrices

Testez la chaîne d'injection SQL. Dans les applications du monde réel, il peut être difficile de vérifier si votre attaque par injection SQL contient une erreur de syntaxe, car généralement les serveurs ne renvoient pas ce type de messages d'erreur. Pour mener votre enquête, vous pouvez copier l'instruction SQL du code source php vers la console MySQL. Supposons que vous ayez l'instruction SQL suivante et que la chaîne d'injection soit ’ ou 1=1;#.

```
SELECT * from credential
WHERE name=’$name’ and password=’$pwd’;
```

Vous pouvez remplacer la valeur de $name par la chaîne d'injection et la tester à l'aide de la console MySQL. Cette approche peut vous aider à construire une chaîne d'injection sans erreur de syntaxe avant de lancer la véritable attaque par injection.

Arrêter le labtainer

Lorsque le laboratoire est terminé, ou si vous souhaitez arrêter de travailler pendant un certain temps, dans le terminal qui vous a permis de le lancer, exécutez : stoplab

Vous pouvez toujours redémarrer le Labtainer et continuer votre travail. Lorsque le Labtainer est arrêté, un fichier zip est créé et copié dans un emplacement affiché par la commande « stoplab ». Une fois le laboratoire terminé, vous pouvez envoyer ce fichier zip au formateur pour correction éventuelle.