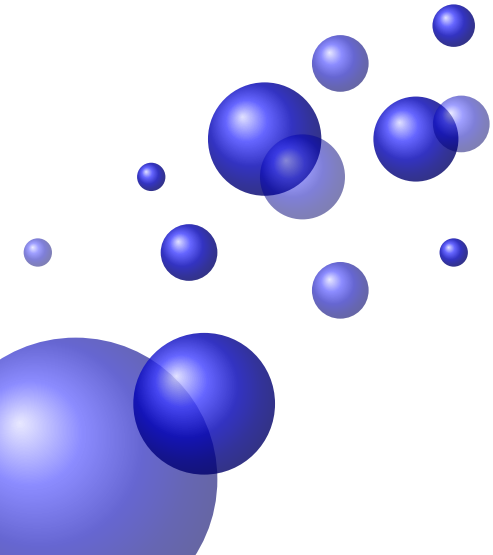
Distribution trouée Damn Vulnerable Web Application

Florian Barbarin - Abdelkader Beldjilali - Alexis Letombe



Le 24 avril 2017

Table des matières

Introduction

Comme TV5 monde en 2015, les pertes des entreprises victimes de cyberattaques se comptent souvent en dizaines de millions d'euros. L'Open Web Application Security Project (OWASP : https://www.owasp.org), publie régulièrement la liste des 10 menaces les plus critiques qui concernent les applications web. Une manière de s'en prémunir consiste à pratiquer le hacking web éthique. C'est là qu'entre en scène l'outil DVWA. Damn Vulnerable Web App (http://www.dvwa.co.uk), est un environnement PHP qui permet de se former à la sécurité des sites web. Le hacker en herbe peut ainsi se former et tester légalement ses compétences sur une application hébergée en local.

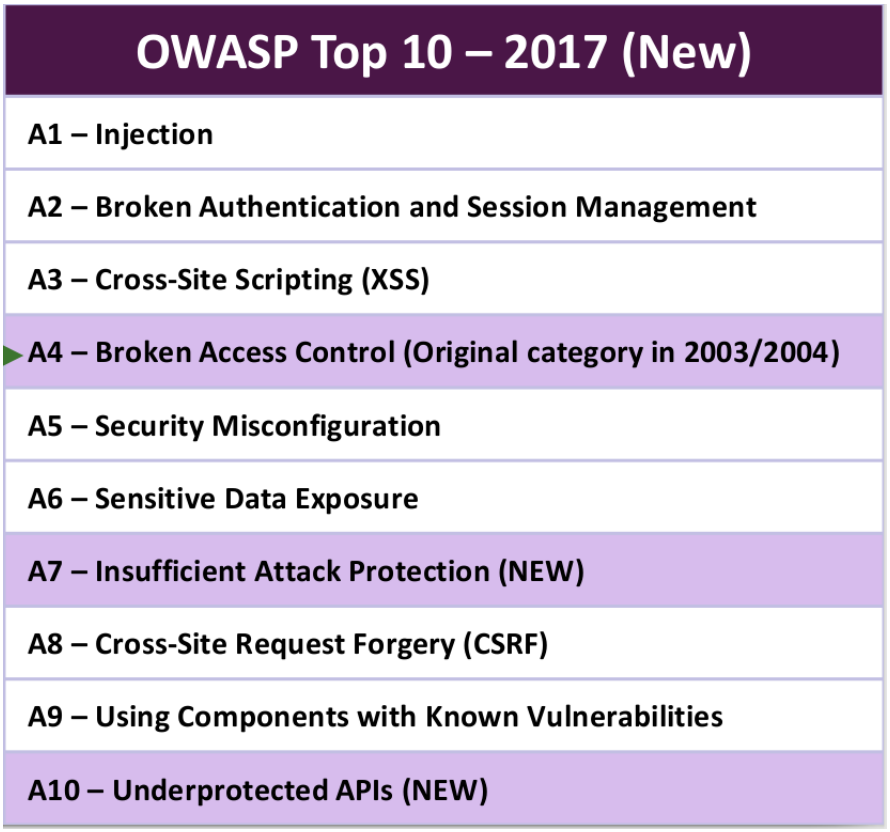


FIGURE 1 – Le top 10 des menaces web 2017 publiées par l'OWASP

1 Installation de la plateforme de test

Une solution possible consiste à installer la distribution kali et le site web DVWA sur une machine virtuelle virtualBox. Il suffit ensuite de modifier la configuration réseau NAT de la machine virtuelle en accès par pont. La commande hostname -I sur la machine virtuelle fournit son adresse IP, par exemple 192.168.1.8. Il ne reste plus qu'à se connecter via firefox au site dvwa via http ://192.168.1.8/dvwa/. Le login par défaut est admin/password.

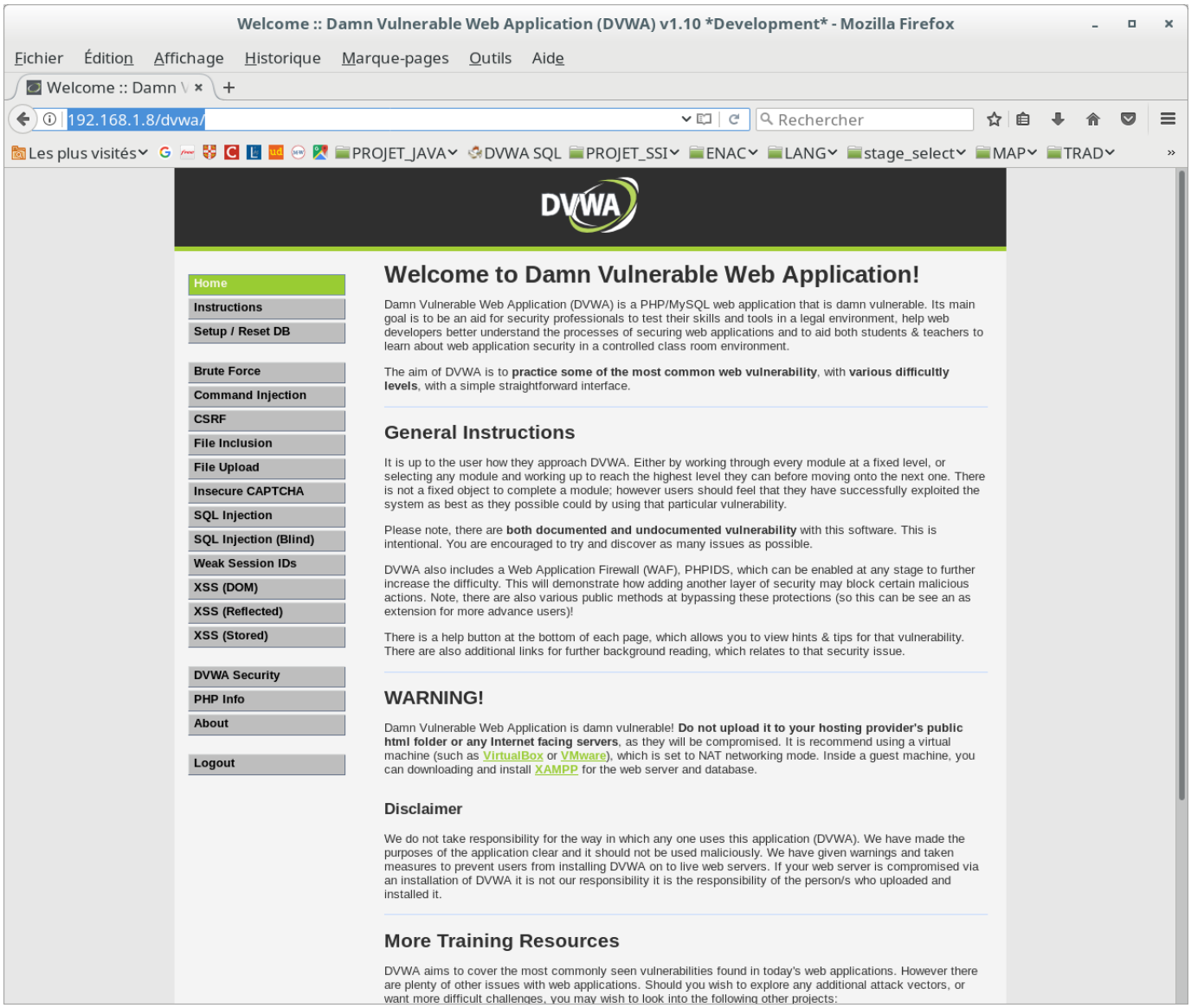


FIGURE 2 – Accès à DVWA à partir du navigateur de la machine hôte. Le site web DVWA est installé sur une machine virtuelle kali linux.

2 type d'attaque

2.1 Description de la vulnérabilité

2.2 Exploitation de la vulnérabilité

2.3 Contre-mesure

3 File inclusion

De nombreux langages de programmation permettent d'inclure des portions de code contenues dans d'autres fichiers que celui en cours d'exécution. Le mécanisme mis à disposition permet de recopier dans le script principal le code contenu dans un autre fichier. Cette procédure est transparente à l'œil de l'utilisateur et peut-être très avantageuse pour le développeur d'un site internet.

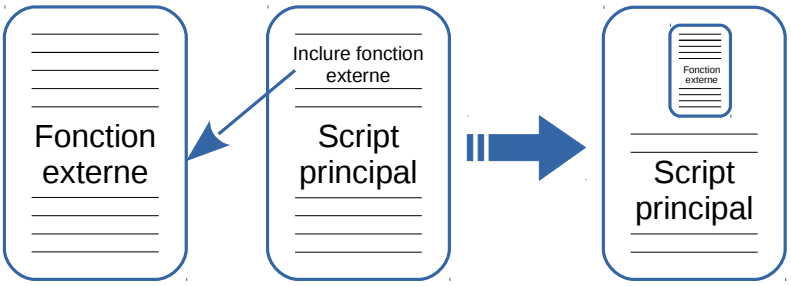


FIGURE 3 – Mécanisme d'inclusion d'un fichier

En effet, inclure du code contenu dans un autre fichier permet, entre autre, les deux utilisations suivantes :

— inclure des portions de code différentes en fonction de choix de l'utilisateur ou de l'environnement de ce dernier ;

— inclure des portions de code utilisées dans plusieurs scripts (par exemple une fonction de connexion à une base de données) afin de ne pas avoir à recopier les mêmes lignes à différents endroits et de ne modifier qu'un seul fichier en cas de modification de la fonction.

3.1 Description de la vulnérabilité

La principale vulnérabilité connue dans le mécanisme que nous venons d'expliciter intervient lorsque l'inclusion d'un script est gérée par une variable pouvant être contrôlée par un attaquant. On se retrouve alors plutôt dans le premier cas d'utilisation indiqué, c'est à dire inclure des portions de code différentes en fonction de choix de l'utilisateur ou de l'environnement de ce dernier. En effet, dans le second cas d'utilisation, l'inclusion du fichier est généralement écrite "en dur" dans le script principal et ne peut donc pas être facilement modifié par un attaquant.
On remarque dans le schéma 3.1 que dans le cas où un attaquant peut avoir accès à la variable permettant de sélectionner le script légitime, celui-ci peut en modifier le contenu de deux façons. On parle alors de Local File Inclusion (LFI) et de Remote File Inclusion (RFI).

3.1.1 Local File Inclusion

Une fois que l'attaquant est en capacité de modifier le contenu de la variable indiquant le nom du script à inclure, celui-ci peut y indiquer un chemin local (i.e. directement sur le serveur) vers un script contenant du code malveillant. Il peut s'agit d'un script que l'attaquant a au préalable placé sur le serveur ou d'un script déjà présent qui effectue des opérations pouvant porté atteinte à la disponibilité de la machine voire à l'intégrité ou la confidentialité des données.

3.1.2 Remote File Inclusion

L'attaquant peut également indiquer dans la variable un chemin distant (i.e. vers un autre serveur) pointant vers un script contenant du code malveillant. Cette technique a pour avantage de faciliter

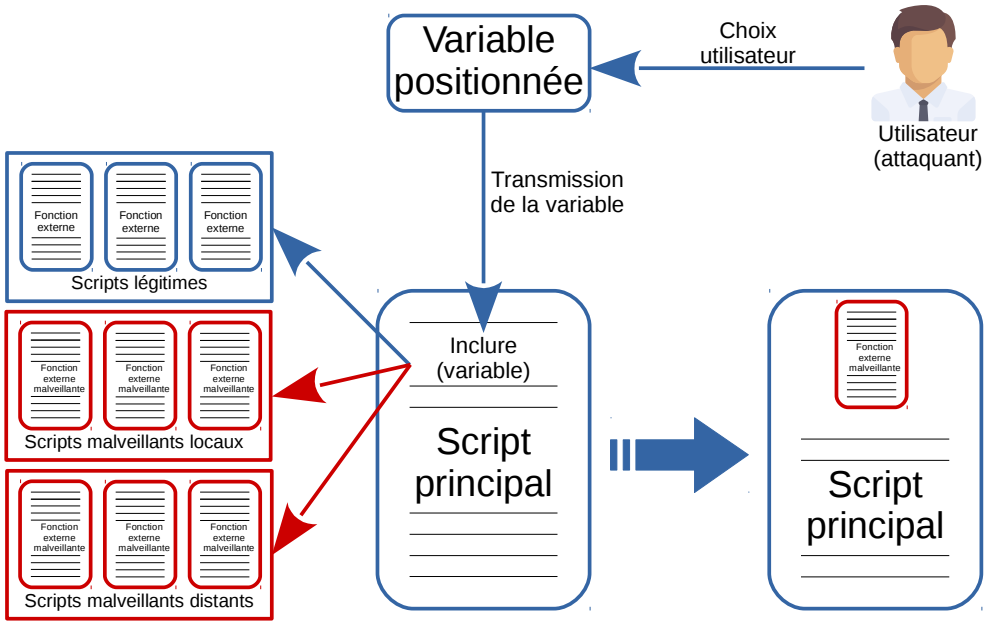


FIGURE 4 – Vulnérabilité d'inclusion d'un fichier

la gestion du contenu du script malveillant par l'attaquant qui peut y inclure toutes les fonctionnalités qu'il souhaite voire le faire évoluer en fonction de la réponse de la machine attaquée.
Dans les deux cas, les scripts malveillants sont recopiés au sein du code du script principal qui sera au final exécuté par le serveur. Cette vulnérabilité offre donc de vastes possibilités à un attaquant qui peut alors faire exécuter par un serveur n'importe quelles fonctionnalités qu'il souhaite.

3.2 Exploitation de la vulnérabilité

3.3 Contre-mesure

4 File upload

4.1 Description de la vulnérabilité

4.2 Exploitation de la vulnérabilité

4.3 Contre-mesure

5 Insecure CAPTCHA

5.1 Description de la vulnérabilité

5.2 Exploitation de la vulnérabilité

5.3 Contre-mesure

6 Injection SQL

6.1 Description

L'injection SQL, en anglais SQL injection ou SQLi en abrégé, est une des attaques les plus dangereuses. Comme pour le Cross Site Scripting présenté dans la suite de ce document, il s'agit ici de tirer parti de l'absence de filtrage des entrées utilisateurs. Cette absence de contrôles permet à un hacker d'insérer du code qui sera interprété par l'analyseur cible, par exemple SQL.

Dans la cas particulier de l'injection SQL et du site DVWA, les requêtes SQL, imbriquées dans des scripts PHP qui récupèrent les saisies des utilisateurs, peuvent être détournées sur la base de la syntaxe du langage.

6.2 Exploitation

La base de données contient 5 utilisateurs identifiés par les entiers de 1 à 5. La mission proposée par le DVWA est de voler leurs mots de passe par injection SQL.
On règle la "DVWA security" sur low de manière à avoir un site web "damn vulnerable". On saisit dans le champ User Id, une simple apostrophe i.e. '. Le site retourne le message "*You have an error in your SQL syntax ; check the manual that corresponds to your MariaDB server version for the right syntax to use near ''''' at line 1*"

Cette simple apostrophe démontre que le site est vulnérable pour deux raisons : d'abord on sait que nos saisies sont interprétées directement par l'analyseur SQL ; elle ne sont pas filtrées. Ensuite parce que le site est "bavard".
Un clic sur le bouton "View Source" affiche le code PHP de la page. On constate, en effet, qu'on peut saisir n'importe quoi dans le champ User Id, il sera transmis sans modification à la requête $query via $id.
L'injection SQL suivante :

```
' UNION select password, last_name from users#
```

donne la requête suivante en remplaçant $id dans le script PHP :

```
$query  = "SELECT first_name, last_name FROM users WHERE user_id = '' UNION
          SELECT password, last_name from users#   ';";
```

Elle indique donc qu'on effectue l'union au sens mathématique des éléments recueillis par les deux requêtes. Le premier SELECT donne l'ensemble vide, le second donne tous les mots de passe et noms de la table users. On obtient donc les mots de passe faussement associés au champ "First name". Ces mots de passe sont cryptés. On pourra utiliser des techniques de révélation par ingénieurie sociales, recherche internet, force brute, dictionnaires ou rainbow tables. L'outil John the ripper peut entrer en action. On note que Smith est certainement aussi admin car ces deux noms d'utilisateurs ont le même hash donc le même mot de passe. Le hacker peut être confiant quant à la suite des opération car Smith ne semble pas être un adepte de la SSI. Le caractère # en fin d'injection évite que PHP n'interprête la suite du code en particulier les caractères apostrophes et guillements qui donneraient une erreur SQL.
NB : C'est une technique répandue que de forcer l'analyseur SQL à ignorer le reste de la requête, en utilisant le symbole commentaire SQL double tiret - - les symboles de commentaires PHP dièse #, / */, // pour assurer que ce qui suit l'injection ne sera pas interprété.*

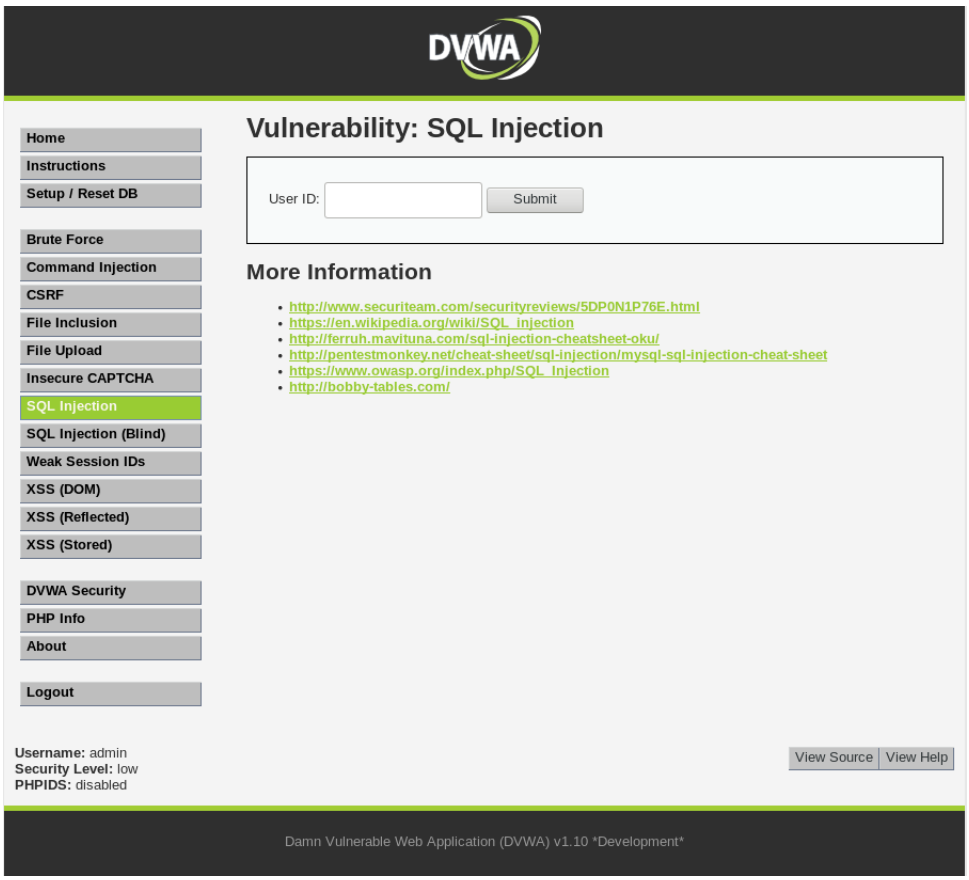


FIGURE 5 – Les saisies incorrectes donnent lieu à un message d'erreur SQL qui informe le hacker potentiel de l'absence de protection contre les SQLi. D'autres informations importantes sont dévoilées comme le type de base de donnée, ici MariaDB, version libre de MySql rachetée par Oracle.

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

Un injection SQL peut aussi donner accès au système de fichier comme le montre l'injection ci-après :

```
[ ' UNION ALL SELECT load_file('/etc/passwd'),null # ]
```

Si l'on part à la recherche des méthodes et fonctions que l'on peut utiliser en PHP afin de se protéger de certains caractères, on peut croiser le chemin de la fonction PHP mysql_real_escape_string. Celle-ci est souvent utilisée pour protéger les requêtes SQL intégrant des paramètres envoyés par l'utilisateur. Néanmoins nous allons voir que cette fonction peut être déjouée, c'est d'ailleurs le but dans ce niveau DVWA.
Si l'on schématise, il faut donc faire passer des caractères comme des guillemets ou des apostrophes, sans inscrire dans l'URL les caractères en question. Un joyeux défi ! Pour ceux qui ont l'habitude de travailler sur les technos web, vous avez peut être déjà entendu parlé de l'encodage, et plus spécifiquement de l'encodage HTTP ou "HTML URL Encoding". Pour ceux qui ne voient pas de quoi il s'agit, je vous conseille de faire un tour sur cette page : http ://www.w3schools.com/tags/ref_urlencod L'HTML URL Encoding permet entre autre de faire passer au travers une URL des caractères spéciaux ou des lettres avec accent qui pourraient ne pas être interprétés correctement par les

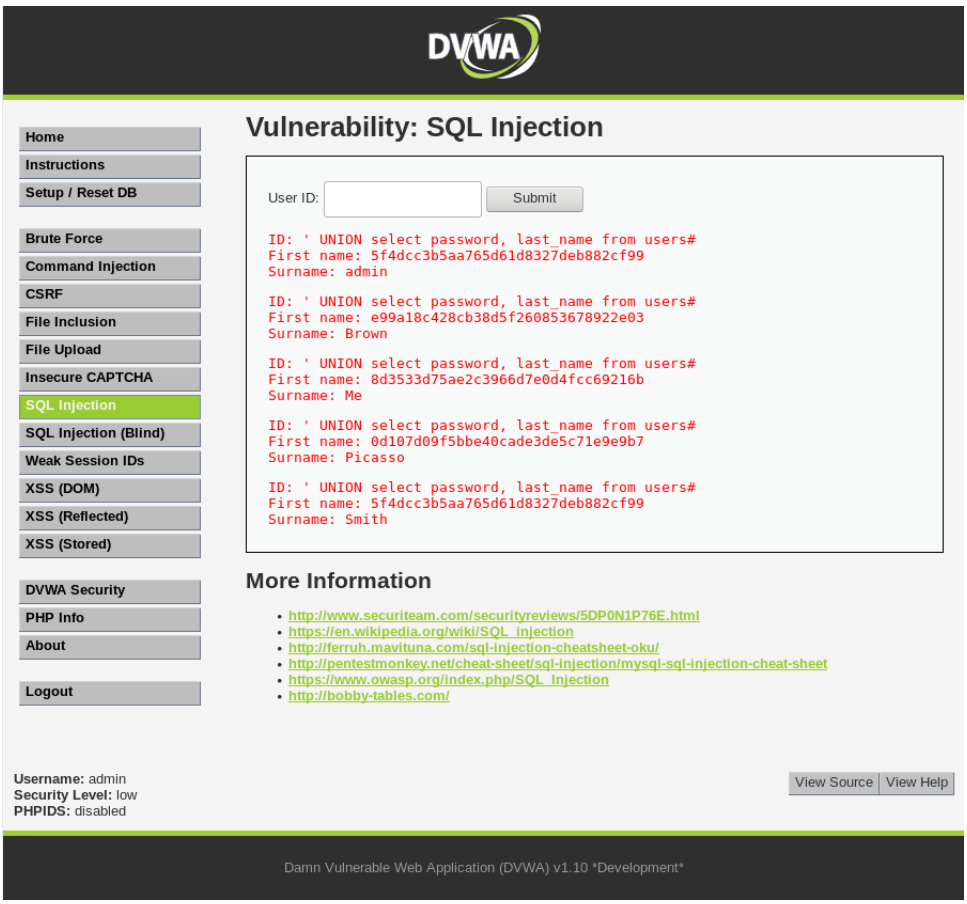


FIGURE 6 – Vol des mots de passe par l'injection SQLi sur site non protégé.

navigateurs ou les serveurs web. Ainsi, l'utilisation d'un encodage basé sur les caractères ASCII est utilisé (notamment la valeur "Hex" des lettres : http ://table-ascii.com/)

`En utilisant l'HTTP URL Encoding, un espace devient un %20 dans l'URL, un  "!" devient u`

`Cela nous permet donc ici de faire passer une guillemet ou une apostrophe de façon encod`

6.3 Contre-mesure

6.3.1 utilisation de

7 Injection SQL aveugle

Blind SQL injection ou BSQLi

7.1 Description

When an attacker executes SQL injection attacks, sometimes the server responds with error messages from the database server complaining that the SQL query's syntax is incorrect. Blind SQL injection is identical to normal SQL Injection except that when an attacker attempts to exploit an application, rather then getting a useful error message, they get a generic page specified by the developer instead. This makes exploiting a potential SQL Injection attack more difficult but not impossible. An attacker can still steal data by asking a series of True and False questions through SQL statements, and monitoring how the web application response (valid entry retunred or 404 header set).

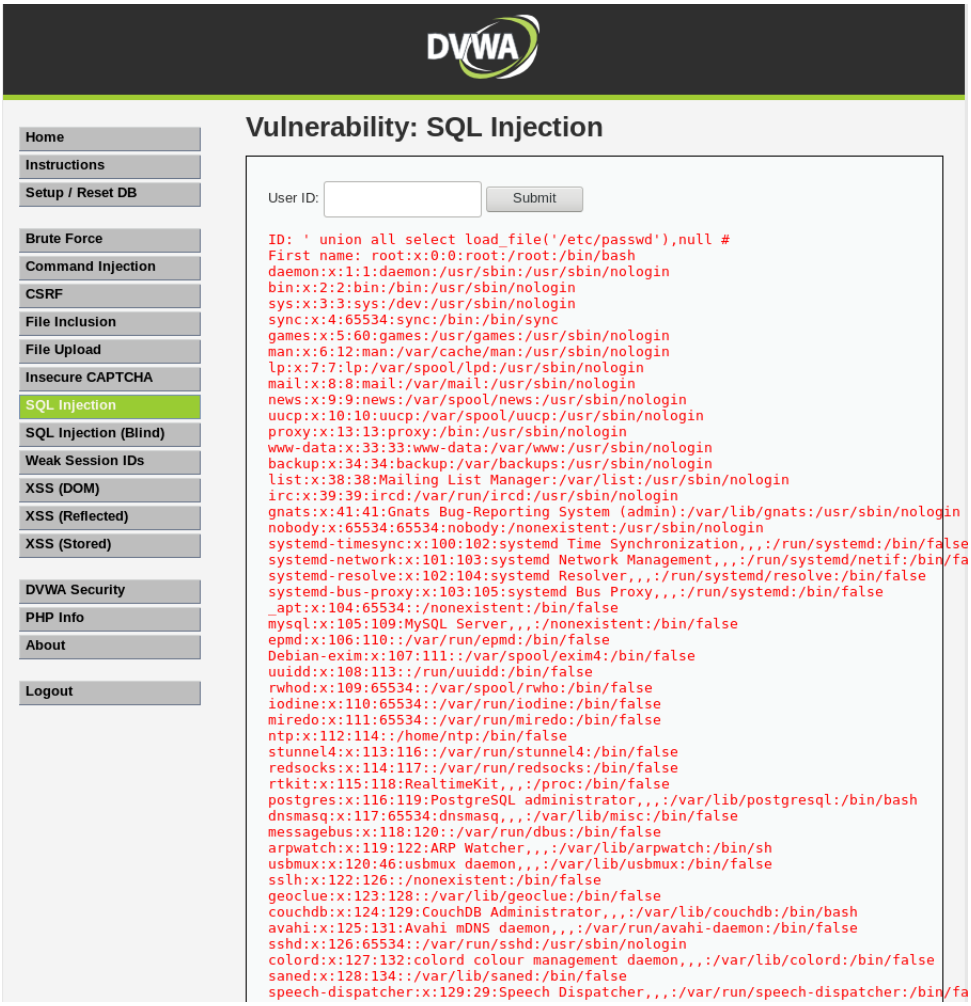


FIGURE 7 – Récupération du fichier /etc/passwd via la commande load_file :

"time based" injection method is often used when there is no visible feedback in how the page different in its response (hence its a blind attack). This means the attacker will wait to see how long the page takes to response back. If it takes longer than normal, their query was successful.

7.2 Exploitation

7.3 Contre-mesure

8 Attaques XSS

Reflected XSS

8.1 Description

désignées au choix par les acronymes CSS ou XSS et qui seront Un site web qui fournit d'une part un service de recueille d'informations via des formulaires et d'autre part un service de publication sur le site de ces mêmes informations

8.2 Exploitation

8.3 Contre-mesure

9 Attaques XSS enregistrées

9.1 Description

9.2 Exploitation

9.3 Contre-mesure

10 Conclusion et perspectives

Nous sommes arrivés au terme de ce voyage dans le monde de la sécurité web au travers de l'étude de DVWA mais nous n'avons fait qu'égratiner le sommet de la partie émergée de l'iceberg. On retiendra cependant les 3 types d'attaques les plus dangereuses : ***les injections, l'authentification et le XSS*** *et deux principes généraux : toujours* ***filtrer les saisies des utilisateurs****, par exemple en utilisant les expressions rationnelles ou une API et penser à* ***configurer le serveur web utilisé*** *notamment pour qu'il soit le moins bavard possible ou qu'il ne donne pas accès à des répertoires sensibles.*

De façon plus générale, que ce soit dans le cadre du développement web ou dans tout autre langage, les techniques de programmation sécurisée et la veille technologique sont devenues primordiales dans un monde essentiellement connecté et en évolution rapide. Une maîtrise complète de toutes les techniques reste cependant théorique. C'est pourquoi, il est préférable d'utiliser des API, bibliothèques et framework web éprouvés et patchés.

Bibliographie

[ACI15] *ACISSI.* Sécurité informatique - Ethical Hacking. *ENI Editions, 2015.*

[GA10] *Philippe Oechslin Gildas Avoine, Pascal Junod.* Sécurité informatique 2ème édition. *Vuibert, 2010.*

Damn Vulnerable Web App (DVWA) est un site web vulnérable. Il permet aux professionnels de la sécurité de tester leurs compétences et leurs outils sans enfreindre la loi et aux developpeurs web de mieux comprendre les principes de sécuri-sation web. C'est aussi un outil pédagogique pour les enseignants et les étudiants. C'est dans cet esprit que ce rapport a été rédigé.