REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la recherche Scientifique

UNIVERSTE SAAD DAHLEB DE BLIDA 1

FACULTE DES SCIENCES

DÉPARTEMENT D'INFORMATIQUE

Mémoire de Fin d'Etude de

Master en SÉCURITÉ DES SYSTÈMES D'INFORMATION

Thème

Audit de sécurité web, Réalisation et implémentation d'un « workflow »qui analyse et exploit des vulnérabilités web

Domain : Mathématique et Informatique

Filière : Informatique

Spécialité : Sécurité des Systèmes d'Information

Encadré par :

M^me^. BOUSTIA Narhimene

M^r^. BELMEHDI Emir Fares

Rédigé par :

OUNOUGHI Abdelhak

TAGHANE Hichem Messaoud

Residente :
Mme CHIKHI Imane

Année Universitaire : 2016/2017

Sommaire

Liste des Tableaux

Liste des Figures :

Résumé:

De nous jours les applications web sont devenues un moyen incontournable. Néanmoins le vecteur le plus important des attaques cybercriminel cibles les applications web. Alors l'entreprise UNIDEES qui est spécialiser dans la sécurité des systèmes informatiques a proposé une solution offensif qui se concrétise à développer un scanner de vulnérabilités existant dans une application web, un produit 100% Algerian sans utilisation de logiciel open source lors de son développement, on 'a bâtisse ce scanner web « PenTwister » il a pour but de détecter, classifier et afficher les vulnérabilités existantes dans une application web, Il est spécialisé dans deux des vulnérabilités les plus répondues selon OWASP « Open Web Application Security Project » qui sont les injection SQL et XSS « Cross-Site Scripting », Ce dernier est doté de 2 composants majeurs qui sont un crawler qui nous permet d'avoir toutes les variables d'entrée disponibles, et un jeu de test aléatoire appeler Fuzzer qui simule un comportement malveillant, son but et d'essayer d'exploiter l'application souhaiter évalue, et il est doté d'une étape capitale qui est audit web ou bien audit des vulnérabilités pour but d'évalué la sévérité de ces dernières, suivant un standard appeler CVSS « Common Vulnerability Scoring System » qui est un standard d'évaluation de la sévérité des vulnérabilités. Comme ça Pentwister détecte les vulnérabilités web et nous aide à sécuriser nos applications web.

Introduction Générale

« World Wide Web » une technologie qui a changé notre quotidien, aujourd'hui l'internet fait du monde un endroit plus agréable à vivre, le meilleur endroit pour exprimer la créativité, faire de l'innovation, avoir des services, l'Internet améliore la qualité de vie et continue de le faire.

Auparavant, La distribution des produits et de services exigeait des ressources importantes à l'entreprise, le marketing et la publicité traditionnelle coutait très chère. Pour la plupart des entreprises, l'Internet entraine quotidiennement des changements dans la nature des pratiques commerciales, les entreprises et les organisations se trouvent confronter à suivre la cadence et à adopter la dernière technologie, cette dernière créait une facilité d'accès vers un marché moderne et des nouvelles fonctions attribuées.

Les systèmes d'informations, en l'occurrence les sites web représentent non seulement un support opérationnel pour les organisations, mais un associé prépondérant quant à l'évolution et le positionnement par rapport au marché.

Autrement dit, les sites web sont devenus la vertèbre principale des systèmes d'information qui représentent le visage de l'organisation ou la vitrine virtuelle de cette dernière.

L'accessibilité publique aux sites web pour les clients, fournisseurs, partenaires, expose ces derniers à des risques engendrés par des vulnérabilités qui peuvent nuire aux systèmes par le biais des attaques et qui peuvent conduire à des situations extrêmes telles que, la compression des effectifs, la réduction des salaires voir jusqu'à la faillite.... sans oublier l'accessibilité interne qui est de loin la plus dangereuse.

Ceci est dû, à la non maturité de l'organisation en terme de mettre en œuvre une politique et un système de sécurité fiable et efficace, qui peut garantir la disponibilité des servies, l'intégrité de l'information ainsi que la confidentialité. D'où les exigences et la satisfaction des normes suivies.

Notre travail est de réaliser un programme qui détecte et exploite les vulnérabilités d'injection qui existe dans une application web pour minimiser les risques et prévenir les dégâts que l'organisation fait face.

Notre mémoire est structurée comme suit :

Le chapitre 1 : on présente un état de l'art ou on va détailler l'évolution des Technologies web puis parler des vulnérabilités qui existent, les contres mesures et protection ainsi que les scanners web.

Le chapitre 2 : on présente notre solution : Architecture, fonctionnent, description des entités et le mode d'emploi.

Le chapitre 3 : on présente la réalisation et l'implémentation : Les outils utilisés, aperçu sur le déroulement de l'application.

Et on clôtura le tout avec une conclusion générale et des perspectives.

Chapitre 1
Technologie et
Vulnérabilité WEB

1. Introduction

Le WEB sans doute est l'une des innovations majeures du siècle courant, on a eu le privilège d'assister à une évolution impressionnante et très rapide de son mode d'utilisation et sa structure. On est conduit à constater son impact qui est très grand sur notre quotidien, cette technologie a changé à jamais notre vie et continue à le faire.

2. Technologie WEB

a. Web 1.0 :

Le web 1.0, est le premier visage de cette technologie qui a révolutionné le monde, crée en 1991, appelé de nous jours « web traditionnel », connu pour être statique et focalisé sur la distribution d'informations, seuls les professionnel ont le droit d'y publier alors les simples utilisateurs n'ont que le droit de lire « c'est une technologie (Read-Only) ». Peut être synonyme à un grand magasine électronique. Ces principaux obstacles sont ces programmes très chère, une interaction avec les utilisateurs très faible sans oublier sa lenteur ainsi que sa lourdeur. Pour cela en 2000, cette approche est remise en question et laisse place à son successeur.[1]

b. Web 2.0 :

Le web 2.0, est la seconde version de cette technologie, une toute nouvelle approche dynamique qui se focalise sur la communication d'où son nom **« web social »**, Elle se base sur l'échange et le partage de l'information grâce à des textes, vidéos, images ou autres, d'où l'apparition des réseaux sociaux.

On constate la forte interaction de l'utilisateur contrairement à la version précédente et l'un des inconvénients les plus importants est la fouille de données.

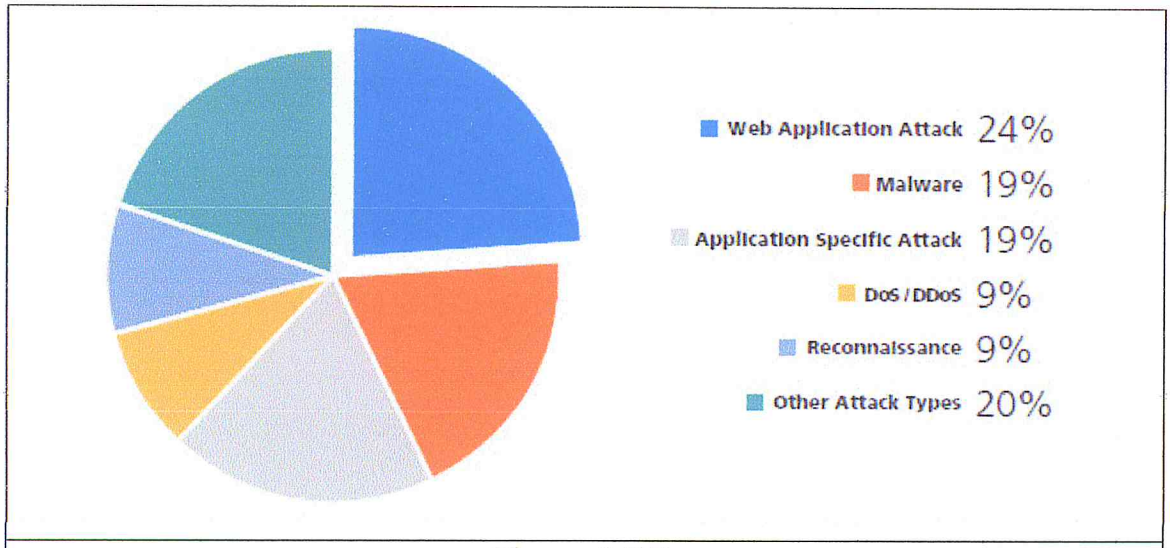
En 2010, une nouvelle approche est apparue pour améliorer la version précédente. [1]

c. Web 3.0 :

Le web 3.0, une nouvelle approche dite sémantique ou bien « web smart » qui a pour but d'organiser la masse d'informations disponible en faction du besoin de chaque individu tout on essaye de faire le lien entre la réalité et le monde virtuel. Techniquement c'est le résultat de la consolidation des contenus dynamiques : Transforme le web en une géante base de

données. Il faut aussi noté que cette nouvelle technologie répond parfaitement aux besoins d'utilisateur mobiles. Le seul point préoccupe, c'est la sécurité car cette approche porte atteinte directement à la vie privée. [1]

3. Vulnérabilités WEB



Figures 1 [15]

Cette illustration montre que les applications web sont le vecteur d'attaque le plus large car ils sont devenus des moyens incontournables dans notre vie et au même titre qu'un system exploitation ou le réseau qui présente des failles depuis des années, une grande variété de vulnérabilité existent et se trouve dans les applications web, certaines plus critiques que d'autre. Plusieurs base de donnes connue ont pour but d'énumérer et d'évaluer ces vulnérabilités avec des statistiques qui indique leur importances, telles que CVE (Common vulnérabilités and exposure), NVD (National Vulnerability Database) ...Ces bases de donnes répertorie tous les types des vulnérabilités incluant celle qui ciblent les application web. Cette multiplication de vulnérabilité a poussé quelque communauté à naitre pour améliorer la sécurité des applications web, parmi ces communautés, OWASP (Open web application Security Project) et WASC (Web application Security Consortium).

OWASP dans l'un de ces projet les plus connue sous le nom du "TOP 10" a créer une classification des vulnérabilités composé de dix classe les plus critiques. Son objectif est de

sensibiliser tout le monde du IT (Information Technologie) pas que les développeurs, concepteurs.

La dernière version 2013 d'OWASP Top 10 liste les vulnérabilités par ordre d'importance :

3.1. Failles d'injection

Une faille d'injection, telle l'injection SQL, OS et LDAP, se produit quand une donnée non fiable est envoyée à un interpréteur en tant qu'élément d'une commande ou d'une requête. Les données hostiles de l'attaquant peuvent duper l'interpréteur afin de l'amener à exécuter des commandes fortuites ou accéder à des données non autorisées.

Exemple de scénarios d'attaque:

Scénario #1: Une application utilise des données non fiables dans la construction de l'appel SQL vulnérable suivant:

String query = "SELECT * FROM accounts WHERE custID='" + request.getParameter("id") +"'";

Scénario #2: Pareillement, la confiance aveugle d'une application aux frameworks peut déboucher sur des requêtes toujours vulnérables :

Query = session.createQuery("FROM accounts WHERE custID='" +request.getParameter("id") + "'");

Dans les deux cas, l'attaquant modifie le paramètre 'id' dans son navigateur et envoie: ' or '1'='1. Par exemple:

http://example.com/app/accountView?id=' or '1'='1

Le sens des deux requêtes est modifié pour retourner toutes les lignes de la table accounts. Les pires attaques peuvent altérer des données, voire invoquer des procédures stockées. [2]

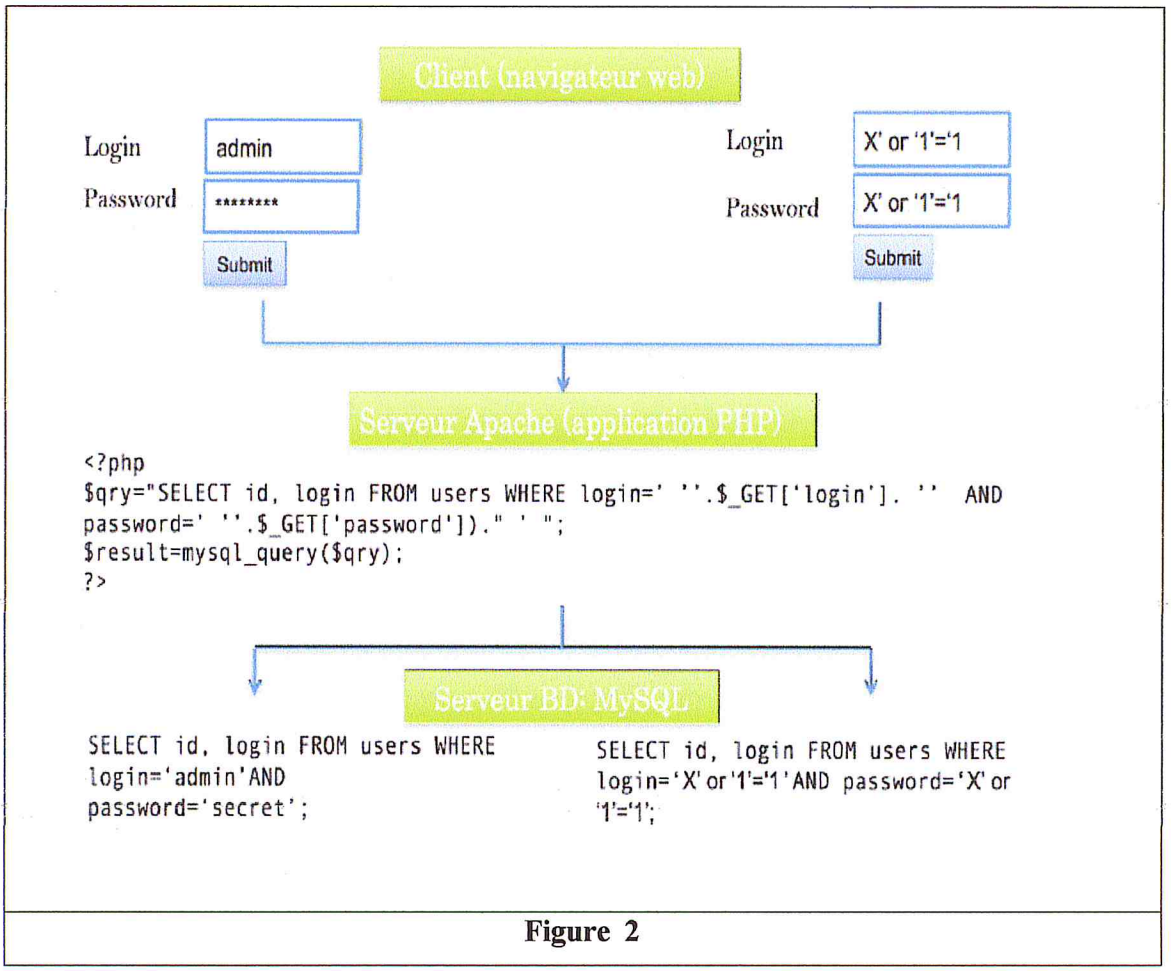


Figure 2

3.2. Violation de gestion d'authentification et de session

Les fonctions applicatives relatives à l'authentification et la gestion de session ne sont souvent pas mises en œuvre correctement, permettant aux attaquants de compromettre les mots de passe, clés, jetons de session, ou d'exploiter d'autres failles d'implémentation pour s'approprier les identités d'autres utilisateurs.

Exemple de scénarios d'attaque :

Scénario #1: Une application de réservation de billets d'avion expose les identifiants de session dans l'URL par réécriture:

http://example.com/sale/saleitems;jsessionid= 2P0OC2JSNDLPSKHCJUN2JV?dest=Hawaii

Un utilisateur authentifié sur le site veut informer ses amis de la vente. Il envoie le lien ci-dessus sans savoir qu'il fournit aussi son ID de session. En cliquant sur le lien, ses amis utiliseront sa session et sa carte de crédit.

Scénario #2: Les timeouts de l'application ne sont pas définies correctement. Un utilisateur accède au site via un ordinateur public. Au lieu de sélectionner "déconnexion", l'utilisateur ferme simplement le navigateur et s'en va. Un attaquant utilise le même navigateur une heure plus tard, et ce navigateur est encore authentifié.

Scénario #3: Un attaquant interne ou externe obtient un accès à la base des mots de passe du système. Les mots de passe ne sont pas correctement chiffrés, exposant les mots de passe de tous les utilisateurs à l'attaquant. [2]

3.3. Cross-site Scripting (XSS)

Les failles XSS se produisent chaque fois qu'une application accepte des données non fiables et les envoie à un browser web sans validation appropriée. XSS permet à des attaquants d'exécuter du script dans le navigateur de la victime afin de détourner des sessions utilisateur, défigurer des sites web, ou rediriger l'utilisateur vers des sites malveillants

Exemple de scénario d'attaque :

L'application utilise des données non fiables dans la construction du fragment HTML sans l'avoir validée ou échappée au préalable :

(String) page += "<input name='creditcard' type='TEXT' value='" + request.getParameter("CC") + "'>";

L'attaquant modifie le paramètre 'CC' dans leur navigateur pour:
'><script>document.location='http://www.attacker.com/cgi-bin/cookie.cgi?foo='+document.cookie</script>'.

Cela provoque l'envoi de l'ID de session de la victime au site web de l'attaquant, permettant à l'attaquant de détourner la session en cours de l'utilisateur. A noter que les attaquants peuvent aussi utiliser XSS pour tromper les contremesures mises en place pour se protéger des attaques CSRF. [2]

3.4. Référence directe non sécurisée à un objet

Une référence directe à un objet se produit quand un développeur expose une référence à un objet d'exécution interne, tel un fichier, un dossier, un enregistrement de base de données ou

une clé de base de données. Sans un contrôle d'accès ou autre protection, les attaquants peuvent manipuler ces références pour accéder à des données non autorisées

Exemple de scénario d'attaque :

L'application utilise une valeur non vérifiée dans une requête SQL accédant à des informations d'un compte :

```
String query = "SELECT * FROM accts WHERE account = ?";
        PreparedStatement pstmt =
        connection.prepareStatement(query , … );

       pstmt.setString( 1, request.getParameter("acct"));
        ResultSet results = pstmt.executeQuery( );
```

L'attaquant modifie le paramètre « VAR » dans son navigateur afin d'envoyer le numéro de compte qu'il souhaite. Si le paramètre n'est pas correctement vérifié, l'attaquant peut accéder à n'importe quel compte, au lieu d'être limité au sien. [2]

Exemple :

http://example.com/app/accountInfo?VAR=Numero_Compte

3.5. Mauvaise configuration de sécurité

Une bonne sécurité nécessite de disposer d'une configuration sécurisée définie et déployée pour l'application, contextes, serveur d'application, serveur web, serveur de base de données et la plate-forme. Tous ces paramètres doivent être définis, mis en œuvre et maintenus, car beaucoup ne sont pas livrés sécurisés par défaut. Cela implique de tenir tous les logiciels à jour.

Exemple de scénarios d'attaque :

Scénario #1: La console d'administration du serveur d'application est automatiquement installée et non désactivée. Les comptes par défaut ne sont pas modifiés.

L'attaquant découvre la console, utilise le compte par défaut et prend le contrôle.

Scénario #2: Le listage des répertoires est activé. L'attaquant le découvre et peut lister les répertoires et trouver les fichiers. L'attaquant trouve et télécharge vos classes java compilées qu'il décompile . Il identifie une faille de contrôle d'accès.

Scénario #3: La configuration du serveur d'application autorise l'affichage d'état de la pile à

l'utilisateur. Les attaquants apprécient ces messages d'erreurs.

Scénario #4: Le serveur d'application est livré avec des exemples d'applications non supprimés de votre serveur de production. Exemple d'application contient des vulnérabilités connues utilisables par l'attaquant pour compromettre le serveur. [2]

3.6. Exposition de donnes sensibles

Beaucoup d'applications web ne protègent pas correctement les données sensibles telles que les cartes de crédit, identifiants d'impôt et informations d'authentification. Les pirates peuvent voler ou modifier ces données faiblement protégées pour effectuer un vol d'identité, de la fraude à la carte de crédit ou autres crimes. Les données sensibles méritent une protection supplémentaire tel un chiffrement statique ou en transit, ainsi que des précautions particulières lors de l'échange avec le navigateur

Exemples de scénarios d'attaque :

Scénario #1: Un site web protège des numéros de carte de crédit au moyen d'une fonction de chiffrement transparent (TDE) du SGBD. Cette méthode induit également un déchiffrement transparent des données lorsqu'elles quittent la base. En exploitant une injection SQL, l'attaquant récupère ainsi les données en clair…

Scénario #2: Un site public ne requiert pas SSL lors de la navigation dans la section authentifiée. Un acteur malveillant se connecte à un réseau sans-fil en libre accès et collecte le trafic d'un utilisateur. Il récupère le jeton d'une session authentifiée et accède ainsi aux données et privilèges de l'utilisateur dans l'application.

Scénario #3: En exploitant une faille dans une fonction d'envoi de fichiers, un acteur malveillant obtient la base de condensés (hashs) de mots de passe. Les condensés ayant été générés sous la forme simple sans sel (salt), une attaque par table arc-en-ciel (rainbow table) lui révèle les mots de passe. [2]

3.7. Manque de contrôle d'accès au niveau fonctionnel

Pratiquement toutes les applications web vérifient les droits d'accès au niveau fonctionnel avant de rendre cette fonctionnalité visible dans l'interface utilisateur. Cependant, les applications doivent effectuer les mêmes vérifications de contrôle d'accès sur le serveur lors de l'accès à chaque fonction. Si les demandes ne sont pas vérifiées, les attaquants seront en

mesure de forger des demandes afin d'accéder à une fonctionnalité non autorisée[2]

Exemples de scénarios d'attaque :

Scenario 1: L'attaquant se contente de visiter les URLs ciblées. Les URLs suivantes nécessitent d'être authentifié et les droits d'administration sont requis pour "admin_getappInfo".

http://exemple.com/app/getappInfo

http://exemple.com/app/admin_getappInfo

Une vulnérabilité existe si un utilisateur non authentifié peut accéder à une de ces pages ou si un utilisateur authentifié mais non privilégié peut accéder à "admin_getappInfo". Dans ce dernier cas, cela peut permettre à l'attaquant d'identifier d'autres fonctionnalités d'administration non protégées.

Scenario 2: Une page utilise un paramètre "action" pour spécifier la fonctionnalité à invoquer, et les différentes actions requièrent des privilèges différents. Une vulnérabilité existe si ces privilèges ne sont pas vérifiés.

3.8. Falsification de requête intersites(CSRF)

Une attaque CSRF (Cross Site Request Forgery) force le navigateur d'une victime authentifiée à envoyer une requête HTTP forgée, comprenant le cookie de session de la victime ainsi que toute autre information automatiquement inclue, à une application web vulnérable. Ceci permet à l'attaquant de forcer le navigateur de la victime à générer des requêtes dont l'application vulnérable pense qu'elles émanent légitimement de la victime

Exemple de scénario d'attaque :

Une application permet à un utilisateur de soumettre une requête de changement d'état, qui ne requiert aucun secret :

http://example.com/app/transferFunds?amount=1500&destinationAccount=4673243243

L'attaquant peut donc forger une requête pour transférer de l'argent du compte de la victime sur son propre compte, et la cacher dans une balise image, ou dans une balise iframe,

stockée sur un site sous son contrôle :

<img

src="http://example.com/app/transferFunds?amount=1500&destinationAccount=attackersAcct#" width="0" height="0"

/>

Si la victime visite l'un des sites de l'attaquant, alors qu'elle est toujours authentifiée sur le site, son navigateur inclura les données de session utilisateur dans la requête forgée et cette dernière aboutira. [2]

3.9. Utilisation de composant avec des vulnérabilités connues

Les composants vulnérables, tels que bibliothèques, contextes et autres modules logiciels fonctionnent presque toujours avec des privilèges maximum. Ainsi, si exploités, ils peuvent causer des pertes de données sérieuses ou une prise de contrôle du serveur. Les applications utilisant ces composants vulnérables peuvent compromettre leurs défenses et permettre une série d'attaques et d'impacts potentiels

Exemple de scénarios d'attaque :

Les risques liés à la vulnérabilité d'un composant peuvent être très variés, allant d'un malware simple voir complexe ciblant une organisation voulue. Puisque la plupart des composants s'exécutent avec les privilèges maximum de l'application, toute faille dans un de ces composants peut avoir un impact majeur. Les deux composants vulnérables suivants ont été téléchargés 22 millions de fois en 2011.

- Apache CXF Authentification Bypass – En ne fournissant pas de jeton d'authentification, les attaquants pouvaient faire appel à n'importe quels web services avec l'ensemble des privilèges. (Apache CXF est un framework opensource à ne pas confondre avec le serveur applicatif Apache.)
- Spring Remote Code Exécution – Un abus de l'implémentation du langage d'expression de Spring permettait aux attaquants d'exécuter du code arbitraire et ainsi de prendre le contrôle du serveur.

Toutes les applications utilisant l'une de ces bibliothèques vulnérables est vulnérable aux attaques de ces composants directement accessible aux utilisateurs de l'application. D'autres bibliothèques vulnérables, utilisées plus profondément dans l'application, seraient plus difficilement exploitable. [2]

3.10. Redirections et renvois non validés

Les applications web réorientent et redirigent fréquemment les utilisateurs vers d'autres pages et sites internet, et utilisent des données non fiables pour déterminer les pages de destination. Sans validation appropriée, les attaquants peuvent réorienter les victimes vers des sites de phishing ou de malware, ou utiliser les renvois pour accéder à des pages non autorisées

Exemples de scénarios d'attaque :

Scénario #1: Une application possède une page "redirect.jsp" disposant d'un seul paramètre nommé "url". Un attaquant forge une URL permettant de rediriger les utilisateurs vers un site malveillant (tentative de phishing ou installation de malwares).

Exemple :

http://www.example.com/redirect.jsp?url=evil.com

Scénario #2: Une application effectue des renvois pour rediriger les utilisateurs sur certaines pages internes. Pour simplifier le renvoi, certaines pages utilisent un paramètre contenant la page où doit être renvoyer l'utilisateur. Dans ce cas, un attaquant crée une URL satisfaisant les contrôles d'accès de l'application et le redirigeant ensuite vers une fonction d'administration à laquelle il ne devrait pas avoir accès. [2]

Exemple :

http://www.example.com/boring.jsp?fwd=admin.jsp

4. Contre mesure et protection

Aujourd'hui les développeurs doivent adopter des différentes méthodes et techniques pour être en mesure de faire front à des menaces basiques mais surtout les menaces avancées permanentes (API) qui peuvent durer un laps de temps très grand.

Certaines mesures de protection ont été proposées :

- La prévention : utiliser les meilleurs pratiques de développement pour concevoir une application de qualité, et empêcher les menaces de lui porter attente.
- La détection : utiliser des tests et des vérifications pour détecter et éliminer les menaces.
- La mise en œuvre des mesures de protection : c'est le fait de mettre en œuvre des barrières de défense qui nous permettent de nous prévenir, détecter et d'être en mesure de tolérer les menaces et intrusion.

- Evaluation : c'est l'estimation de l'impact des risques et l'efficacité des mesures de protection.

4.1. La prévention

Pour respecter cette mesure les développeurs doivent programmer leur code source selon un ensemble de règle et de modèle de programmations strict et précis, qui permet de concevoir une application de qualité avec le but de minimisé le risque et d'atténué les menaces pendant tout son cycle de vie.

Il s'avère que c'est très difficile de sécurisé une application déjà existant, alors il est fondamentale de prendre en considération les aspects de sécurité durant les phases de développement pour essayer de maitrise le maximum des menaces.

Parmi les processus de réalisation d'applications sécurisées on trouve le guide proposer par OWASP « Application Security Vérification Standard » (ASVS), pour mettre en œuvre une stratégie de sécurité afin d'améliorer le processus de réalisation par exemples.

Il existe aussi le projet OWASP "Enterprise Security API" (ESAPI) qui contient un ensemble de contrôle qui sert à s'assurer que l'écriture du code source est respectée et conforme au contrôle de sécurité standard.

L'usage de ces règles peut beaucoup aider à minimiser le risque mais ça reste insuffisant.

4.2. La Détection

Le programmeur est exposé à faire des erreurs lors de la programmation. Pour cela, la détection et l'élimination des erreurs est une tache très importante. Deux étapes sont nécessaires :

Cette mesure se constitue de 2 méthodes de test :

Vérification statique pour détectes des vulnérabilités dans des applications, cette technique peut détecter les erreurs de programmation avant même que l'application ne soit déployé dans le marché.

Vérification dynamique appelé aussi scanner de vulnérabilité, c'est une approche boite noire ou bien boite blanche qui n'hésite pas à exploiter l'application comme un pirate. Ces scanners web peuvent détecter beaucoup de type de vulnérabilité.

Il faut noter aussi que les deux méthodes statique et dynamique sont complémentaire, il faut les appliqué toutes les deux pour avoir des résultats optimales.

4.3. La mise en œuvre des mesures de protection

Cette mesure consiste à mettre en œuvre des systèmes physique ou bien logique pour but de détecter des comportements malveillantes destiné à des applications web, ces systèmes sont connue sur le nom des : Firewall, WAF (web app firewall) , IDS , IPS …..

4.4. Evaluation Continue

Cette étape a pour but de s'assurer de l'efficacité des autres mesures mise en œuvre et avoir une estimation des risques qui menacent les applications, Ce risque dépend du niveau de la menace et l'efficacité de la contre mesure établie.

5. Les Scanners web

Les Applications web sont devenue de plus en plus vulnérable et exposé à des attaques qui peuvent porter préjudice aux 4 piliers de la sécurité ,tels que Plusieurs Organismes étatiques et privés ont été cibles par des attaque web , l'attaque sur le centre de visa de la Russie par une attaque injection SQL en décembre 2016, l'attaque de Tumblr en 2013 qui a réussie a divulgué plus de 50 million comptes (Nom Utilisateur + Mot de passe)

Dans cette partie, nous allons décrire les scanner web :

5.1. Définition d'un scanner web

Le scanner de sécurité des applications web est un programme qui interagit avec des applications web pour diagnostiquer des vulnérabilités présente est établie en résultat un rapport bien détaillé pour aider la sécurisation de l'application en question. [3]

5.2. Type de scanner web

a. Scanner boite noire :

Il simule des attaques depuis l'extérieur sans aucune information fournie par le client, l'objectif est de créer un scénario réel de la situation de sécurité de l'application analysé.

Des exemples de scanner Black Box :

- HP (SPI Dynamics) WebInspect & DevInspect
- IBM Rational (Watchfire) AppScan
- Cenzic Hailstorm
- NT Objectives NTO Spider
- Acunetix Web Vulnerability Scanner

b. Scanner boite blanche :

Ce type de scanner simule une attaque interne. Le client fournit toutes les informations nécessaires pour tester l'architecture complète de l'application.

Des exemples de scanner White Box :

- Fortify Source Code Analyzer
- Ounce Labs
- Coverity Prevent SQS

5.3. Fonctionnement d'un scanner

Les attaques les plus courantes concernant les serveurs Web sont les attaques d'injection qui proviennent de l'absence de test de conformité des paramètres d'URL ou des données fournies dans les champs des formulaires.

Pour vérifier si ces attaques d'injection de code sont possibles, les scanners web envoient des requêtes particulières et analysent les réponses retournées par le serveur.

La page de rejet correspond à la détection par le serveur de valeurs d'entrée malformées ou invalides. Une page d'exécution est renvoyée par le serveur suite à l'activation réussie de la requête.

Elle peut correspondre soit au scénario "normal", dans le cas d'une utilisation légitime du site, soit à un détournement de son exécution via l'exploitation réussie d'une injection de code (via des entrées non conformes).

Pour identifier les vulnérabilités d'un site Web, les scanners soumettent au site des requêtes contenant des données non conformes correspondant à des attaques potentielles.

Les réponses sont alors analysées afin d'identifier les pages d'exécution. Si une page d'exécution est identifiée, la page correspondante est considérée vulnérable.

C'est ainsi que les outils détectent l'absence de test de conformité des paramètres.

On peut distinguer deux principales classes d'approches adoptées par les scanners de vulnérabilités :

- par reconnaissance de message d'erreurs dans les requêtes renvoyées par le serveur
- par l'étude de similarité des pages renvoyées.
- Par l'étude des réponses aux réponses saines.

a. Approche par reconnaissance de messages d'erreurs :

Pour identifier les injections possibles, cette approche consiste à envoyer des requêtes d'un format particulier et chercher des motifs spécifiques dans les réponses tels que les messages d'erreurs.

L'idée fondamentale est que la présence d'un message d'erreur dans une page HTML de réponse signifie que la requête correspondante n'a pas été vérifiée par l'application Web avant d'être transmise aux fonctions.

Par conséquent, le fait que cette requête a été envoyée inchangée révèle la présence d'une vulnérabilité.

b. Approche par étude de similarité des réponses :

Le principe de cette approche consiste à envoyer différentes requêtes spécifiques aux types de vulnérabilités recherchées et à étudier la similitude des réponses renvoyées par l'application en utilisant une distance textuelle. En fonction des résultats obtenus et de critères bien définis, on conclut sur l'existence ou non d'une vulnérabilité.

Parmi les inconvénients de cette approche :

La distance considérée pour l'étude de similarité considère la fréquence des mots sans tenir compte de l'ordre des mots dans un texte.

Ignorer l'ordre des mots peut amener à ignorer la sémantique d'une page et à nouveau peut amener à mal juger si deux pages sont identiques ou non. Par exemple, les pages suivantes partagent le même vocabulaire, mais elles correspondent à une authentification réussie et échouée respectivement :

- You are authenticated, you have not entered a wrong login.
- You are not authenticated, you have entered a wrong login.

Conclusion :

Pour conclure de nous jours le web est indispensable, néanmoins il est exposé à des attaques et comporte plusieurs vulnérabilités, parmi les solutions de protection on trouve les scanner web vu leur importance, dans le chapitre qui suit on va vous présenter « PenTwister: Vulnerability Scanner ».

Chapitre 2

« PenTwister »

Vulnérabilité Scanner

1. Introduction

Notre but est de réaliser un scanner web qui peut détecter, classifier et afficher les vulnérabilités existantes dans une application web. On traitera 2 des vulnérabilités les plus répondues d'après OWASP TOP 10.

- Injection (SQLi)
- Cross Site Scripting (XSS)

Ce dernier est doté de 2 composants majeurs :

a. Un Crawler qui nous permet d'avoir toutes les variables d'entrée disponibles.
b. Un Fuzzer qui simule un comportement malveillant, son but et d'essayer d'exploiter l'application.

Les résultats positifs seront alors affichés dans un tableau de bord sous forme d'un rapport.

Le scanner a aussi un scan passif qui fournit toutes les informations nécessaires. Sachons aussi que toutes les informations sont mémorisées dans une base de données pour faciliter le stockage et la manipulation des données.

Toutefois l'administrateur peut crées des sessions pour qu'un intervenant puise lancer le scan.

Dans cette partie nous allons décrire et présenter la conception de notre Scanner avec ces différents modules.

2. Mise en œuvre de la solution

2.1. Vue globale :

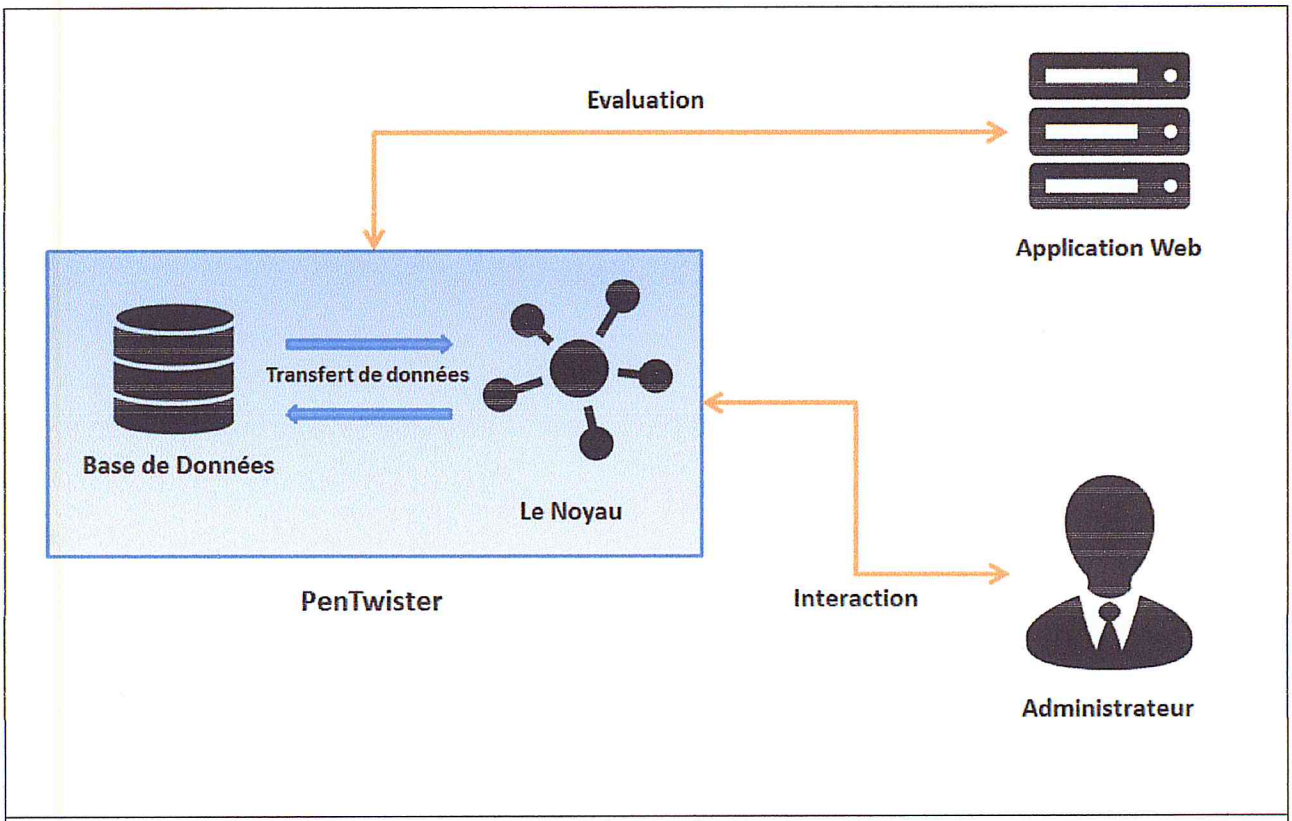


Figure 3

La première interaction est de s'authentifié avec l'application « PenTwister » et de se présenter en tant qu'Admin, puis lancer un scan active qui va faire crawler l'url présenter, le Fuzzer, déterminer la sévérité des résultats trouvé, les sauvegarder dans la base de donnée puis les afficher dans un format clair.

2.2. Fonctionnement :

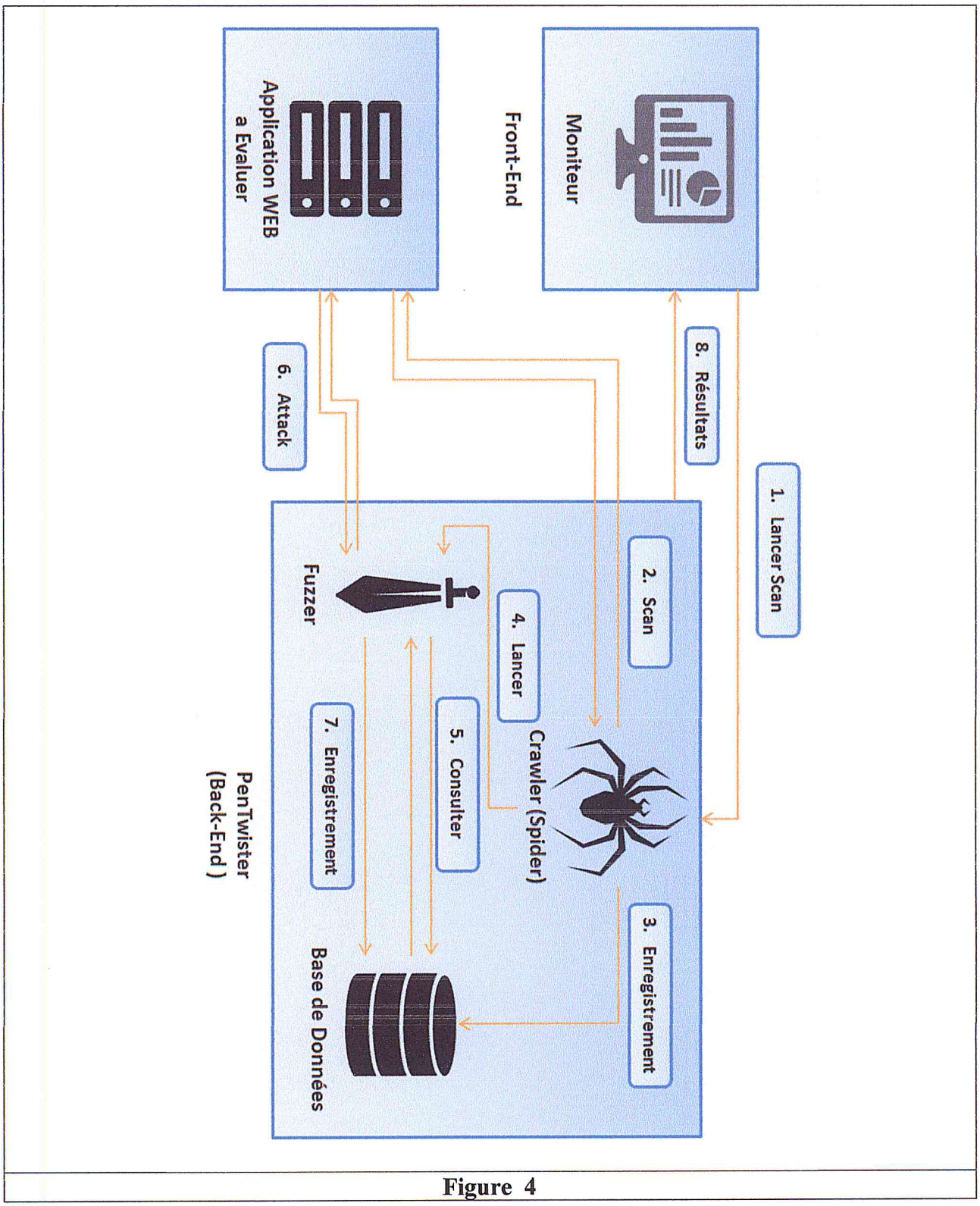


Figure 4

2.2.2. Scenario des scans avec PenTwister :

A. Scan Active :

Action	Description
1	Lancement du crawler
2	Crawler toutes les pages du site WEB
3	Obtention des résultats du crawler
4	Stockage des résultats du crawler dans la base de données
5	L'utilisation des résultats obtenus pour effectuer des injections
7	Lancement des injections
8	Obtention des résultats
8	Stockage des résultats du fuzzer dans la base de données
9	Affichage de tous les résultats dans le moniteur
10	Exécution du scan passive
11	Exécution du scan passive
12	Stockage des résultats du scan passif dans la base de données
13	Affichage de tous les résultats dans le moniteur

Tableau1

2.3. Crawler

2.3.1. Définition :

Un WEB Crawler aussi nommé Web Spider est un moteur de recherche qui a pour but de créer des indexations de data trouvé dans des pages web avec une manière méthodologique. Ce dernier collecte les informations pour aider a illustré des diffèrent statistiques. Beaucoup de moteur cherche l'utilise pour optimiser leur recherche comme (google,yahoo,youtube…)

2.3.2. Fonctionnement du crawler

Un crawler en général commence avec une liste de URL à visiter, ces URL sont nommé « Racines », Lors d'exploration des racines, le crawler analyse la page web et identifie tous les hyperliens dans la page et les ajoute à la liste à visiter. Ensuite le crawler analyse ces nouveaux liens pour de nouveau hyperliens et ainsi de suite, récursivement.

Le Crawler envoie des requêtes HTTP pour des documents sur d'autre machines sur internet, tout comme un navigateur web fait. C'est un concept de base derrière la mise en œuvre du web crawler, mais la mise en œuvre de ce concept ne s'arrête pas ici. La section suivant décrit les difficultés liées à la mise en place de ce system.

2.3.3. Difficulté dans la mise en place d'un Web Crawler optimisé et efficace

Il existe plusieurs caractéristiques importantes du web qui génèrent un scenario difficile au crawler :

- Grand volume de pages web.
- L'Authentification
- Les différents liens externes incorporés dans les pages web comme Google, Facebook, etc…
- Existence de plusieurs autres documents comme les images, vidéos, fichiers compressé etc.…
- Taux de changement sur les pages web.

Un grand volume de pages Web et les différents liens externe qui existe implique que le web crawler soit suffisamment intelligent pour prioriser les téléchargements, exclure les documents non utiles pour ne pas se perdre infiniment à crawler une application.

Un autre problème est que les pages Web sur Internet changent très souvent, par conséquent, au moment où le Crawler télécharge la dernière page à partir d'un site, la page peut changer ou une nouvelle page a été mise à jour sur le site.

2.3.4. Adopter la stratégie correcte

Les difficultés rencontrées dans la mise en place d'un Crawler Web efficace indiquent clairement qu'il devient essentiel d'explorer le Web non seulement d'une manière évolutive mais efficace, si une quantité raisonnable de qualité ou de fraîcheur des pages Web doit être maintenue. Il s'ensuit qu'un chasseur doit choisir soigneusement à chaque étape les pages à visiter prochainement.

Ainsi, l'implémentation d'un Crawler d'exploration doit définir un comportement méthodologique :

- Sélection du meilleur algorithme pour décider quelle page télécharger.
- Stratégie pour éviter de surcharger les sites Web.
- Stratégie pour éviter de recharger les sites Web déjà vue et les documents non utiles.
- Stratégie comment revoir les pages pour vérifier les mises à jour.
- Stratégie pour éviter de le robot.txt sur les sites Web.

2.3.5. Sélection d'un bon algorithme

Etant donné la taille actuelle du web, il est essentiel que le programme du crawler télécharge qu'une fraction de la page web et pas la page entière, mais on doit garder en esprit que l'algorithme doit s'assurer que ces pages web sont bien choisies en fonction de leur importance.

2.3.6. Nos algorithmes d'exploration

a. Exploration simple :

Cette méthode utilise le concept de base du crawler ou on analyse les pages téléchargé pour trouver de nouveaux liens à visiter.

b. Exploration profonde :

Cette méthode exige que le crawler télécharge autant de ressources que possible, de cette façon le crawler peut trouver tous les liens et les chemins dans un site web particulier

Exemple :

On demande la réponse de cette page : https://www.page.com/1/2/3/index.php

On va alors tentera d'explorer /1, /2 et /3.

L'avantage de cette méthode est qu'elle est très efficace a trouvé des ressources isolées.

c. Exploration focalisé :

Cette méthode exige que le crawler évite de télécharge les pages similaires et les pages non désirables, Nous souhaitons pouvoir prédire la similitude d'une page avant de la télécharger. Ceci impact directement sur la performance du crawler car il devient plus objectif et rapide.

d. Exploration Avec Authentification

Cette méthode ajoute au crawler la possibilité de s'authentifier a des sites web pour pouvoir découvrir et crawler des pages qui exclus l'interaction non authentifié.

Chaque application qui oblige une authentification, utilise une codification de session qui la vérifie à chaque requête demandé, donc on s'authentifie avec Nom Utilisateur et mot de passe et puis on sauvegarde ce cookie pour pouvoir accéder à l'application web.

Exemple :

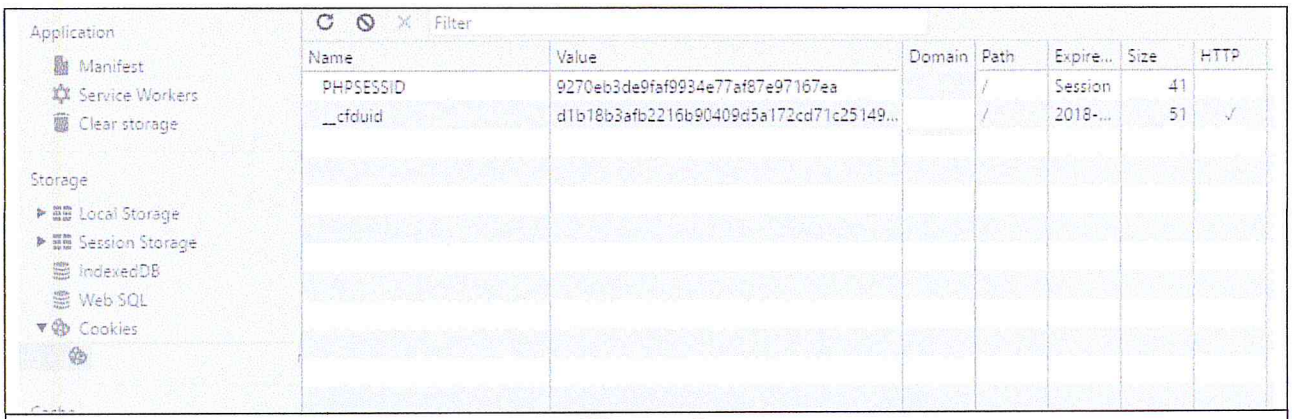


Figure 5

e. Exploration malveillante

Cette méthode exige que le crawler, évite le Protocol robot.txt qui empêche les agents de navigation web.

Ce protocole utilise des comparaisons de sous-chaînes simples pour correspondre aux motifs définis dans le fichier robots.txt. Donc, en utilisant ce fichier robots.txt, nous devons nous assurer que nous utilisons des combinaisons de Caractère ajouté au chemin du répertoire ou bien de changer les agents par defaults pour éviter l'exclusion.

Exemple d'un fichier robot.txt qui indique a tous les agents de navigation de ne pas entrer dans quatre répertoires d'un site web :

```
User-agent: *
Disallow: / cgi-bin /
Disallow: / images /
Disallow: / tmp /
Disallow: / private /
```

Figure 6

2.3.7. Stratégie d'exploration

Beaucoup de site web adopte une stratégie de sécurité pour se protéger et parmi ces stratégies la et la vitesse, le temps entre chaque requête reçu. Donc notre crawler a besoin d'éviter de se faire bloquer en changeant le délai d'envoi de chaque requête.

2.3.8. Pseudo Code pour un crawler web

Son mode d'emploi et le suivant :

En premier lieu, il faut collecte toutes les navigations du site web et leurs états, en se basant sur une recherche exhaustive qui permet d'avoir toutes les navigations disponibles sur le site en question. Pour arriver à ça, il est nécessaire d'avoir un point de départ qui est le nom du domaine du site. Notre seconde manœuvre consiste à stocker les requêtes envoyées en commençant par le point de départ : Pour stocker une requête, il faut analyser les lignes HTML de la page et trouver des liens, le premier lien nous permet de construire la requête suivante et les autres seront empiler dans une fille d'attente pour une analyse ultérieure. Sachons que cette manœuvre peut être infinie, on définit un paramètre à notre algorithme qui est la profondeur maximale et qui représente la borne à ne pas dépasser.

Une fois arrivé à la borne et la page HTML n'a plus de liens, dans une telle situation on dépile un lien de la file d'attente et on le soumet à un contrôle pour éviter les doublures et enlever les liens externe, après ceci on recommence la première étape.

Algorithme crawler (url)

Prenez la première URL dans la liste des URL.
Marquez cette URL comme URL déjà recherchée.

Bien que on pas crawler toute la liste répétez:

Si le protocole d'URL n'est pas http :
break;

si robot.txt dissalow l'URL :
Eviter_robot(url)

Ouvrir l'URL

Si l'URL ouverte n'est pas un fichier HTML :
Break;

Analyser le fichier HTML

Si le texte html contient d'autres hyperliens :

Si hyperlien trouvé est un fichier HTML :
si l'URL n'est pas marquée comme recherché :
Marquer cette URL comme URL déjà recherchée.

Sinon, si le type de fichier est demandé par l'utilisateur
Ajoutez à la liste des fichiers trouvés.

/*
Collecter tous les variables trouvés dans chaque réponse après chaque itération et les vérifier pour éviter la duplication .... etc
*/

2.4. Identification et Classification des Paramètres d'entrée:

Au moment du l'exploration et l'analyse des hyperliens, notre crawler fait une autre tâche et qui est la récolte des paramètres à exploiter dans l'étape suivante.

Pour ce faire on a adopté trois méthodes de classification de potentielles vulnérabilités qui sont :

2.4.1. Collecter les variables d'entrée à partir du « form »

```
<div class="vulnerable_code_area">

    <h3>User ID:</h3>

    <form action="#" method="GET">
        <input type="text" name="id">
        <input type="submit" name="Submit" value="Submit">
    </form>


</div>
```

Figure 7

Lors de l'analyse des pages HTML :

- La premiere etape et de mémoriser les balises qui commence par <form> et se termine par </from>.
- La seconde étape vise à distinguer si la méthode HTTP utilisé du form est bien « GET » ou « POST».
- La dernière étape consiste à séparer les forms SQL et XSS.

Alors la classification sera comme suit :

- Méthode POST + Parametre Simple => SQLI POST
- Méthode POST + Balise <textarea> => XSS POST
- Methode GET + Parametre Simple => SQLI GET
- Methode GET + Balise <textarea> => XSS GET

2.4.2. Collecter des variables d'entrée à partir des liens

Cette méthode s'appuie sur l'application des expressions régulière sur les liens URL des pages web. Ensuite, on mémorise les variables obtenues, exemple :

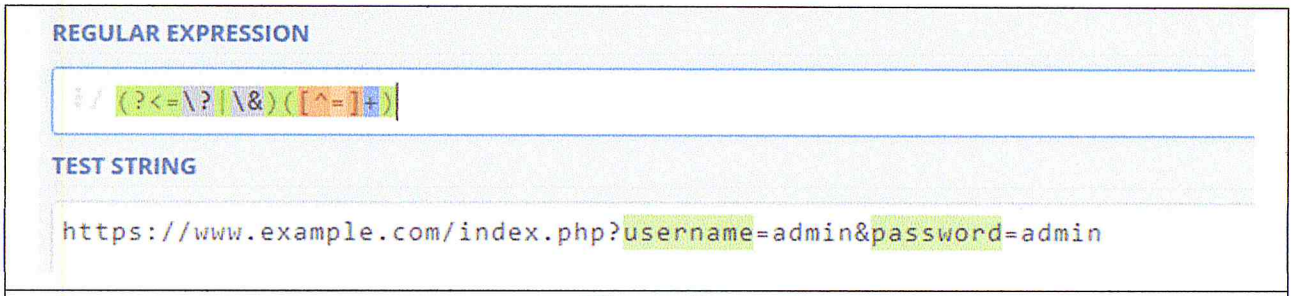


Figure 8

2.4.3. Collecter les variables d'entrée à partir des Balise HTML

Cette méthode vient après les 2 méthodes précédentes. Si on n'est pas dans un formulaire, on vise à chercher les variables d'entrée entre les autres balises comme qui se trouve dans les lignes HTML des pages web, on mémorise ensuite chaque variable qui se situe entre et .

Exemple :

```
<!DOCTYPE html>
<html>
        <head>
                <title>test</title>
        </head>
        <body>
                <div id= ' code Vulnerable ' >
                    <input name= ' id' />
                </div>
        </body>
</html>
```

Figure 9

2.5. Test à données aléatoires (Fuzzer)

2.5.1. Définition

Le test à données aléatoires ou bien Le Fuzzing est une technique qui consiste à injecter des données bien spécifique dans les variables d'entrées collectées précédemment par le crawler afin de tester l'application web et détecter des potentiel failles de sécurité.

2.5.2. Taxonomie Fuzzing

Terme	Définition
Fuzzing Stupide	Envoie des requête au hasard sans prise en compte la structure des donnes
Fuzzing intelligent	Envoie des requêtes au hasard avec prise en compte la structure des donnes
White Box Fuzzing	Envoie des donnes mal formées pour but de tester et de ne pas exploiter
Black Box Fuzzing	Envoie des donnes mal formé pour but d'exploiter l'application

Tableau2

2.5.3. Mode d'emploi

Son mode d'emploi est le suivant :

En premier lieu, il faut récupérer les variables d'entrées stocker dans la base de données qui en était identifier et classifier dans l'étape collection des variables d'entrées.

En second lieu, il faut préparer les injections et les attaques en les stockant dans la base de données, suivant une catégorisation et une spécialisation de chaque injection conformément à la classification de l'étape précédente. Une fois les variables et les injections prêtes, on commence l'étape fuzzing, qui contient deux cas de figure:

- Requête à une variable :

Dans ce cas, on injecte les attaques une après l'autre, et on passe à la prochaine étape qui est la détection des vulnérabilités.

- Requête à n-variable :

Ce cas est plus complexe, on n'a adopté un algorithme récursive qui nous permet de traiter toutes les combinaisons possibles, l'algorithme est le suivant :

Algorithme Fuzzer (lien,Parametres[])

Si **Paramètres[]** sont pas tous prêt :
    Préparer **Paramètres[]**
    Fuzzer (lien, **Paramètres[Paramètre - 1]**)
Sinon :
    Envoyer Requête normale (sans injection)
    Pour chaque injection :
        Préparer Requête à l'aide de **Paramètre[],** lien, cookies et injection
        Envoyer Requête avec Injection

        Réponse = Requête avec injection – Requête normale
        Si Réponse n'est pas vide :
            Si Réponse n'existe pas dans **Résultats[]** :
                Ajouter Réponse au **Résultat[]**
Sauvegarder **Résultats[]**

2.5.4. Optimisation de Notre Fuzzer

Il existe 3 caractéristiques importantes ont été prises en compte pour la mise en place de notre Fuzzer :

- On a évité à sauvegarder les réponses similaires en utilisant la distance de Levenshtein.
- On ajoute L'Authentification au Fuzzer.
- On a utilisé la technologie Multithread pour chaque paramètre à Fuzzer.

Distance de Levenshtein :

La distance de Levenshtein est une distance mathématique donnant une mesure de la similarité entre deux chaînes de caractères. Elle est égale au nombre minimal de caractères qu'il faut supprimer, insérer ou remplacer pour passer d'une chaîne à l'autre.

Dans notre cas on utilise la distance de Levenshtein sauvent dite (LD) pour comparer la page web cible avec ces réponses après injection d'attaques bien spécifiques, la manœuvre consiste à récupérer la mesure de changement qui y est un nombre entier et l'analyser ensuite. Si le nombre est faible ça veut dire que le changement et presque nulle, ici c'est souvent une réponse négatif du server par exemple « login failed ». Mais si le nombre est d'une valeur très grande alors le changement est très important, on conclue donc que la page cible a changé et par conséquent notre attaque a bien marcher.[4]

2.6. Audit de Vulnérabilités

L'analyse des vulnérabilités découverte dans les scans est une étape cruciale pour améliorer la sécurité. Chaque vulnérabilité trouvée dans le processus d'analyse doit passer par un test d'audit pour confirmer sa validité et on le fait par algorithme qui calcule le score CVSS en fonction de la facilité d'exploitation, la capacité d'exécution à distance, l'exigence d'accès physique et d'autres critères....

Ce score CVSS varie entre [1.0 et 10.0], c'est un system d'évaluation standardisé utilisé dans les tests de conformité de l'industrie. Cette évaluation est constituée de 3 mesures appelées métrique :

- La métrique de base est unique et immuable.
- La métrique temporelle est unique mais évolue avec le temps.
- La métrique d'environnement évolue avec l'évolution des systèmes d'informatique.

Le Standard a attribué pour chaque composant de chaque métrique une valeur bien définit qu'il faut ensuite calculer pour sortir avec un résultat global qui détermine la sévérité de la vulnérabilité.[5]

- **Niveau de gravité :**

CVSS qui est une métrique de vulnérabilité standard de l'industrie qui attribue un niveau de sévérité pour chaque intervalle de valeur comme suit :

Gamme CVSS V3	Gravité
[0.1 - 3.9]	Faible
[4.0 - 6.9]	Moyen
[7.0 - 8.9]	Haute
[9.0 - 10.0]	Critique

Tableau3

On peut aussi grouper les gravité Haute et Critique dans un seule intervalle.

Voici quelque exemple de vulnérabilités qui peuvent entrainer un niveau de gravité donné :

a) Niveau Critique :

Les vulnérabilités dont le score dans la gamme critique comportent généralement la plupart des caractéristiques suivantes:

- L'exploitation de cette vulnérabilité entrainera des dégâts majeurs.
- L'exploitation est simple et rapide, l'attaquant n'a pas besoin de compétences ou des connaissances sur la victime.

Pour ce niveau il est conseillé de réparer ou de mettre à niveau le plus tôt possible.[5]

b) Niveau Elevé :

Les vulnérabilités dans la cette gamme ont généralement certaines des caractéristiques suivantes:

- La vulnérabilité est difficile à exploiter.
- L'exploitation pourrait entraîner des privilèges élevés.
- L'exploitation pourrait entraîner une perte importante de données ou des temps d'arrêt. [5]

c) Niveau Moyen :

Les vulnérabilités qui obtiennent un score dans la moyenne ont généralement certaines des caractéristiques suivantes:

- Des vulnérabilités qui obligent l'attaquant à manipuler les victimes individuelles via des tactiques d'ingénierie sociale.
- Les vulnérabilités de déni de service difficiles à configurer.
- Exploits qui exigent qu'un attaquant réside sur le même réseau local que la victime.

- Vulnérabilités où l'exploitation n'offre qu'un accès très limité.
- Des vulnérabilités nécessitant des privilèges d'utilisateur pour une exploitation réussie.[5]

d) Niveau Faible :

Les vulnérabilités à faible portée ont généralement très peu d'impact sur les activités d'une organisation. L'exploitation de ces vulnérabilités nécessite habituellement un accès au système local ou physique.[5]

2.7. Rapport des Scan

Au final, vient la dernière partie qui est le reporting. C'est une partie indispensable et très importante et elle se divise en deux catégories :

A. Rapport en temps réelle :

Ce reporting ce fait lors du scan, ou on montrera les différents statistiques :

- Nombre de Pages trouvées.
- Nombre de paramètres trouvés.
- Nombre d'injections réussies.
- Nombre de requêtes envoyées.
- La sévérité des vulnérabilités.

B. Rapport normale :

Ce rapport est simple ou on sauvegarde les résultats du scan dans un fichier avec toutes les statistiques.

2.8. Base de données

La base de données sert à stocker l'ensemble des informations de notre scanner web.

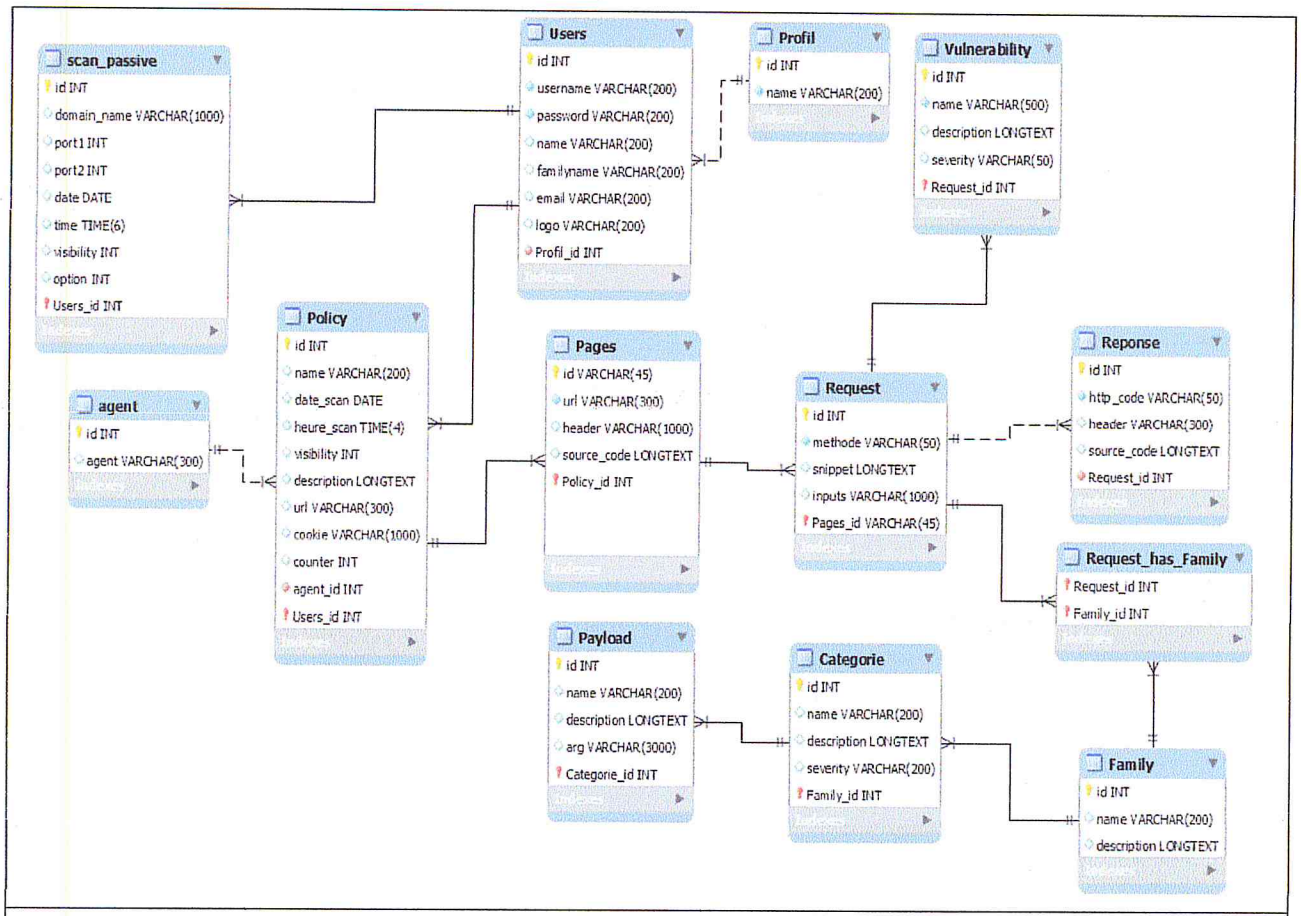


Figure 10

2.9. Conclusion :

Dans ce chapitre, on a proposé un scanner web qui peut détecter et classifier des vulnérabilités en simulant un comportement malveillant d'une manière rapide et optimale, et délivrer un rapport détaillé à la fin de la procédure.

Dans le chapitre qui suit, on va s'intéresser à l'implémentation et outils de développement.

Chapitre 3
Réalisation et Implémentation

Réalisation et Implémentation

1. introduction :

L'environnement de développement exigé par la société d'accueil, est un environnement fonctionnant sur un système d'exploitation Windows avec un mode d'accès en client léger (ie: accès via le serveur et aucune application installée au niveau des postes de travail).

L'architecture de développement préconisée est l'architecture n-tiers.

2. les outils utilisés :

2.1. Django :

Django est un cadre de développement web source ouverte en python. il a pour but de rendre le développement web 2.0 simple et rapide. pour cette raison, le projet a pour slogan « le framework web pour les perfectionnistes sous pression ». développé en 2003 pour le journal local de lawrence (kansas), django a été publié sous licence bsd à partir de juillet 2005.

Django est simple d'accès, riche de nombreux outils et peut être complété par de très nombreuses applications. [6]

2.2. Python 2.7.13 :

Python est un langage de programmation objet, multi-paradigme et multiplateformes. il favorise la programmation impérative structurée, fonctionnelle et orientée objet. il est doté d'un typage dynamique fort, d'une gestion automatique de la mémoire par ramasse-miettes et d'un système de gestion d'exceptions,il est ainsi similaire à perl, ruby, scheme, smalltalk et tcl.

le langage python est placé sous une licence libre proche de la licence bsd3 et fonctionne sur la plupart des plates-formes informatiques, des supercalculateurs aux ordinateurs centraux4, de windows à unix avec notamment gnu/linux en passant par macos, ou encore android, ios, et aussi avec java ou encore .net. il est conçu pour optimiser la productivité des programmeurs en offrant des outils de haut niveau et une syntaxe simple à utiliser.

il est également apprécié par certains pédagogues qui y trouvent un langage où la syntaxe, clairement séparée des mécanismes de bas niveau, permet une initiation aisée aux concepts de base de la programmation. [7]

2.3. Pycharm :

Pycharm est un environnement de développement intégré (abrégé edi en français ou en anglais : ide (integrated development environment)) utilisé pour programmer en python.

il offre l'analyse de code, un débogueur graphique, la gestion des tests unitaires, l'intégration de logiciel de gestion de versions, et supporte le développement web avec django.

il est développé par l'entreprise tchèque jetbrains. il est multi-plateforme et fonctionne sous windows, mac os x et linux. il est décliné en édition professionnelle, réalisé sous licence propriétaire, et en édition communautaire réalisé sous licence apache. [8]

2.4. html5

Le html (« hypertext mark-up language ») est un langage dit de « marquage » (de « structuration » ou de « balisage ») dont le rôle est de formaliser l'écriture d'un document avec des balises de formatage. les balises permettent d'indiquer la façon dont doit être présenté le document et les liens qu'il établit avec d'autres documents. il permet notamment la lecture de documents sur internet à partir de machines différentes, grâce au protocole http, permettant d'accéder via le réseau à des documents repérés par une adresse unique, appelée url. [9]

2.5. CSS

Css est un langage de style qui définit la présentation des documents html. par exemple, css couvre les polices, les couleurs, les marges, les lignes, la hauteur, la largeur, les images d'arrière-plan, les positionnements évolués et bien d'autres choses. attendez de voir. html peut être (mal) utilisé pour la présentation des sites web. mais css offre plus d'options et se montre plus précis et sophistiqué. css est pris en charge par tous les navigateurs actuels. [10]

2.6. Mysql workbench

Mysql workbench est un outil visuel unifié pour les architectes de bases de données, les développeurs et dba. mysql workbench propose une modélisation de données, le développement de sql et les outils d'administration complets pour la configuration du serveur,

gestion des utilisateurs, la sauvegarde, et bien plus encore. mysql workbench est disponible sur windows, linux et mac os x. [11]

2.7. Scrapy

Scrapy est un API open-source permettant la création de robots d'indexation. Développé en python, il dispose d'une forte communauté, offrant de nombreux modules supplémentaires. la première version stable a été publiée en septembre 2009. Depuis, l'équipe de développement publie régulièrement de nouvelles versions dans le but d'enrichir le Framework en fonctionnalité. L'objectif principal est d'obtenir un api stable pour la version 1.02. Le Framework dispose d'une communauté et un support active. [12]

2.8. jquery

jquery est une bibliothèque javascript libre et multi-plateforme créée pour faciliter l'écriture de scripts côté client dans le code html des pages web2. la première version est lancée en janvier 2006 par john resig.

la bibliothèque contient notamment les fonctionnalités suivantes :

parcours et modification du dom (y compris le support des sélecteurs css 1 à 3 et un support basique de xpath) .

événements.

effets visuels et animations .

manipulations des feuilles de style en cascade (ajout/suppression des classes, d'attributs…) .

ajax .

plugins .

utilitaires (version du navigateur web…).[13]

2.9. Javascript

Javascript est un langage de programmation de scripts principalement employé dans les pages web interactives mais aussi pour les serveurs2 avec l'utilisation (par exemple) de node.js. c'est un langage orienté objet à prototype, c'est-à-dire que les bases du langage et ses principales interfaces sont fournies par des objets qui ne sont pas des instances de classes, mais qui sont chacun équipés de constructeurs permettant de créer leurs propriétés, et notamment une propriété de prototypage qui permet d'en créer des objets héritiers personnalisés. en outre, les fonctions sont des objets de première classe. le langage supporte le paradigme objet,

impératif et fonctionnel. javascript est le langage possédant le plus large écosystème grâce à son gestionnaire de dépendances npm, avec plus de 350 000 paquets3.
javascript a été créé en 1995 par brendan eich. il a été standardisé sous le nom d'ecmascript en juin 1997 par ecma international dans le standard ecma-262. le standard ecma-262 en est actuellement à sa 7e édition. javascript n'est depuis qu'une implémentation d'ecmascript, celle mise en œuvre par la fondation mozilla. l'implémentation d'ecmascript par microsoft se nomme jscript, tandis que celle d'adobe systems se nomme actionscript. [14]

3. Aperçu sur le déroulement de l'application :

3.1. Login page

Page d'authentification est la première interaction avec notre application

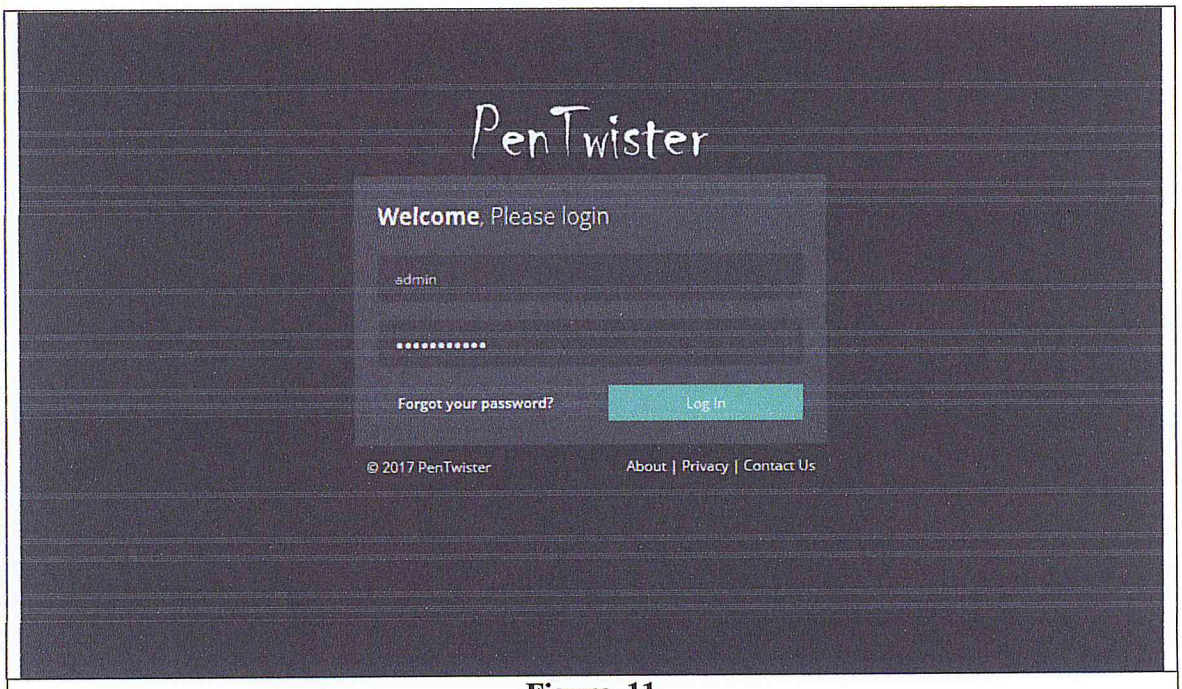


Figure 11

3.2. Définition d'une politique :

Pour scanner une application faut ajouter une politique comme suit :

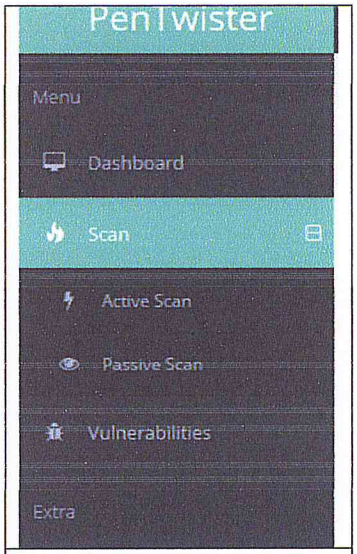


Figure 12

Puis accéder à « Active scan » :

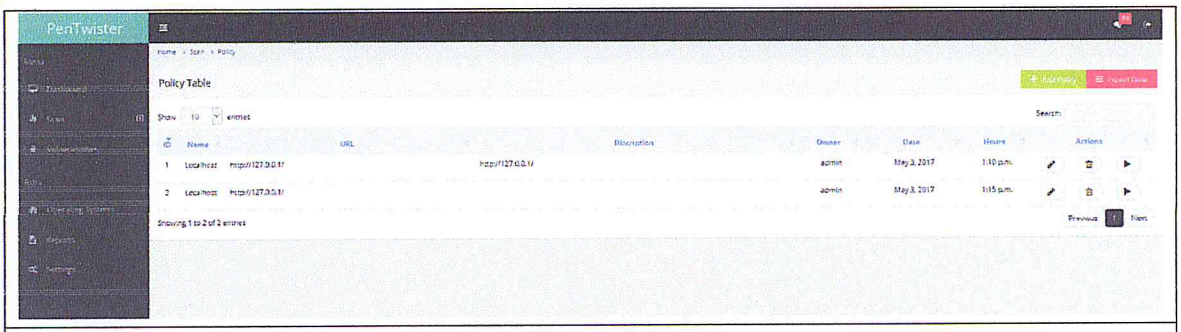


Figure 13

On peut ajouter une Politique en cliquant sur « Add policy » qui nous donne ce qui suit :

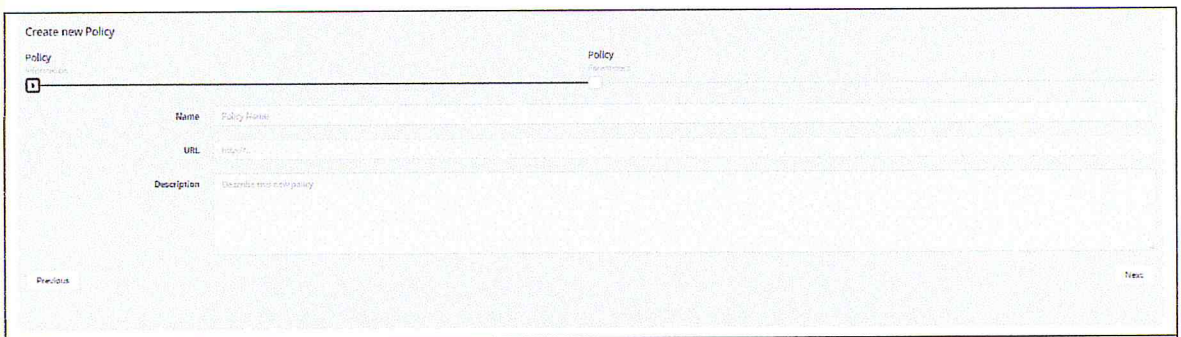


Figure 14

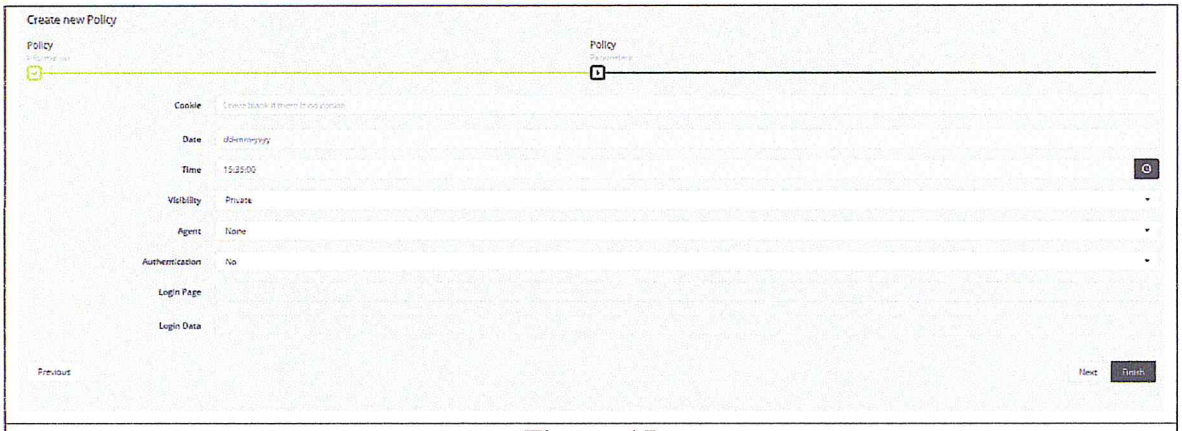


Figure 15

Apres on peut supprimer, éditer ou lancer un scan en cliquant sur les boutons à droite:

Editer :

Supprimer :

Lancer Scan :

En cliquant sur lancer Scan on aura le rapport en temps réelle comme suit :

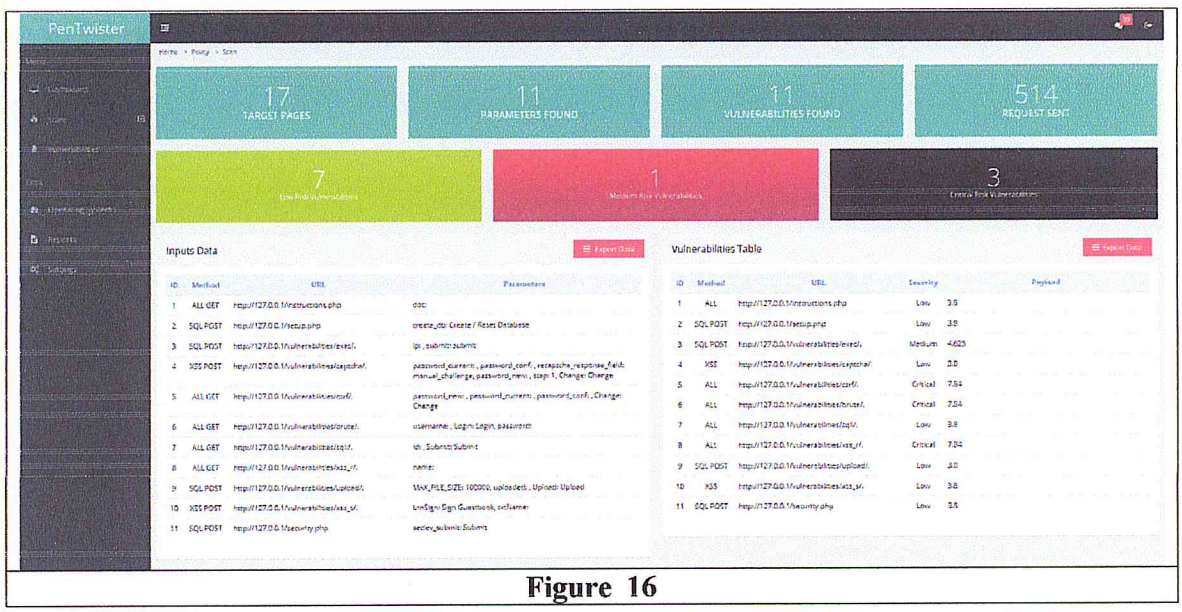


Figure 16

Conclusion

Dans ce chapitre, nous avons énumérer les outils et le langage utiliser dans la réalisation de notre application « PenTwister » en démontrant aussi son déroulement, En effet, nous avons achevé l'implémentation, tout en respectant la conception élaborée. En d'autres termes, nous détenons maintenant une version acceptable du logiciel, installée dans notre environnement de développement. Ainsi que nous avons prévenu la plateforme sous laquelle le système sera installé dans l'environnement des utilisateurs.

Conclusion générale :

De nos jours, les applications web sont incontournables dans notre vie quotidienne grâce à leur capacité à répondre à un grand nombre de besoin, leur popularité ne cesse d'augmenter du à la diversité de fonctionnalités proposées par le développement web qui accroît de jour en jours. Cependant ce développement web vas de pair à égal avec une multiplicité de vulnérabilités, cela implique que le vecteur d'attaque contre les serveurs web est de plus en plus important, ainsi que la complexité des attaques augmente exponentiellement, et sachons qu'un serveur web n'est jamais sûr à 100%, les applications web sont confronté à un très grand risque qui menace les trois piliers de la sécurité informatique, la confidentialité, la disponibilité et l' intégrité. De ce fait pour faire face à ces menaces, il est fondamental de mettre au point une stratégie et de développer des techniques qui puisse détecter les vulnérabilités existantes dans les applications web pour faciliter la sécurisation de ces derniers, la solution proposé dans le cadre de ce projet et de réaliser un scanner web qui vise à attendre ces objectifs :

- ✓ Détection des vulnérabilités présentes dans une application web.
- ✓ Quantification et classification des vulnérabilités détectées.
- ✓ Minimisé le temps du scanne on utilisant la technologie multithread.
- ✓ Création d'un rapport bien détaillé et précis.
- ✓ Conception d'un scanner facile à exploiter, fiable et optimisé pour élever le niveau de sécurité des applications web.

Ainsi la valeur ajoutée du scanner web réalisée est la fiabilité, la précision des diagnostiques, et en particulier la facilité de manipulation. Nous pouvons envisager plusieurs plans de future pour notre scanner web comme traiter les autres points de «OWASP» top 10 et enrichir notre « fuzzer » de plus d'attaques pour des résultats encore plus optimaux.

- [12]« https://media.readthedocs.org/pdf/scrapy/1.0/scrapy.pdf» pour la définition et la documentation de Scrapy , consulté le 31-06-2017

- [13]« http://glossaire.infowebmaster.fr/jquery/» pour la définition de jquery , consulté le 31-05-2017

- [14]« http://glossaire.infowebmaster.fr/javascript/» pour la définition de Javascript , consulté le 31-05-2017

- [15]« https://www.calyptix.com/top-threats/top-5-cyber-attack-types-in-2016-so-far/» pour l Top Cyber Attaques en 2016 , consulté le 2-6-2017