Sécurité et vulénrabilités
Évaluation FINALE - TD

[Saisir votre nom et prénom]

# PRÉSENTATION

#

## Description

Ce travail a pour but d’analyser des vulnérabilités liées à l’usage de certains protocoles réseaux afin de proposer une amélioration de la sécurité informatique d’un système d’information.

## À titre informatif

Les outils abordés dans ce support sont uniquement utilisés à des fins éthiques (Hacking étique) et pédagogiques. Leur usage est formellement interdit en dehors de ce cadre sur un réseau tiers sans autorisation explicite.

## Composition du labo

La maquette qui vous est proposé pour le laboratoire dispose de 3 machines pouvant être virtuelles:

* Un client Windows ou Linux
* Un attaquant Kali Linux
* Un serveur Metasploitable

## Schéma dans le scénario MITM

Le schéma ci-dessous illustre les machines utilisées dans ce travail en prenant pour exemple le cas d’une attaque MITM.

L'attaque de l'homme du milieu ou man-in-the-middle attack (MITM), parfois appelée attaque de l'intercepteur, est une attaque qui a pour but d'intercepter les communications entre deux parties, sans que ni l'une ni l'autre ne puisse se douter que le canal de communication entre elles a été compromis.


# I) Vérification de l’intégrité

## Téléchargement d’un logiciel

Le logiciel Notepad++ servira de modèle pour effectuer la vérification d’intégrité

* Aller sur le site https://notepad-plus-plus.org/ puis aller dans la rubrique téléchargement.
* Sur la page de téléchargement, la somme de contrôle peut être affichée en cherchant bien, avec indication de l’algorithme de hachage utilisé.

Par exemple, pour ce qui est de la somme de contrôle en SHA1 :

* Télécharger la dernière version de Notepad++ correspondant à votre machine cliente (Windows ou Linux) depuis le site officiel du logiciel
	1. Relever, depuis le site officiel, la somme de contrôle associée au fichier téléchargé.

Somme de contrôle relevé ->

* 1. Expliquer à quoi sert une somme de contrôle.

Explications ->

* 1. Quelles sont les principales différences entre les algorithmes MD5 et SHA256 ?

|  |  |
| --- | --- |
| ALGORITHMES | EXPLICATIONS |
| MD5 |  |
| SHA256 |  |

* 1. Une somme de contrôle permet-elle de garantir la confidentialité des échanges ?

Explications ->

## Vérification de la somme de contrôle

Dans un environnement Windows, la somme de contrôle de la même ressource peut être générée à l’aide d’une commande PowerShell : Get-FileHash npp.7.5.4.Installer.x64.exe -Algorithm SHA1 | Format-List

Il existe plusieurs outils en ligne de commande qui permettent de calculer des sommes de contrôle d’un fichier. Par exemple, sous Linux, l’outil **sha1sum** peut être utilisé.

La somme de contrôle calculée doit être identique à celle indiquée sur le site officiel. Il faut aussi faire attention à l’algorithme utilisé qui doit correspondre à celui indiqué sur le site officiel.

La comparaison peut s’effectuer à l’aide de la commande **diff** sous Linux une fois la valeur de l’empreinte du fichier extraite.

* 1. Vérifier que la somme de contrôle du logiciel téléchargé est authentique.

Explications ->

* 1. Conclure sur l’intérêt des calculs de sommes de contrôle pour des entreprises.

Explications ->

# IV) Sécurité d’un site web

## Découverte de l’application Mutillidae

Pour cette partie, vous allez travailler à partir de l’application web Mutillidae ainsi que la base de données « owasp10 ». Pour cela, une modification de configuration sur la machine virtuelle « Metasploitable 2 » est nécessaire.

* 1. Vérifier, en ligne de commande, l’existence de la base de données sur la machine.

Capture ->

Voici un rappel de commandes de base pour l’administration de bases de données d’un serveur MySQL :

* *SHOW DATABASES ;* ⇒ Liste les bases disponibles.
* *USE nom\_de\_la\_base ;* ⇒ Définit la base de données à utiliser (sur laquelle se positionner).
* *SHOW TABLES ;* ⇒ Liste les tables d’une base de données.
* *DESCRIBE nom\_de\_la\_table ;* ⇒ Affiche la structure d’une table (champs et types).
	1. Donner la structure de cette base de données.

Capture ->

Pour l’application web Mutillidae, les paramètres d’accès à la base de données sont définis dans le fichier « config.inc » situé à la racine des sources de l’application (dans /var/www/mutillidae).

* Mettre à jour ces paramètres pour utiliser la base de données « owasp10 » (les autres paramètres ne sont pas à modifier).

L’accès à l’application Mutillidae se fait ici à partir de l’URL [http://192.168.X.X/mutillidae/](http://172.16.10.5/mutillidae/).

* Se connecter sur cette application web.

## Mise en place d’un ARP Spoofing

Un hacker est capable d’empoisonner le cache ARP d’un client accédant au site web et de récupérer le mot de passe de son compte Mutillidae via une connexion non sécurisée http.

Pour cela il est nécessaire de réaliser au préalable une attaque de l’homme du milieu qui nécessite elle-même un empoisonnement du cache ARP.

* 1. Consulter le cache ARP de la machine cliente avant de réaliser l’attaque puis noter l’association entre l’adresse MAC et l’adresse IP du serveur web Metasploitable.

|  |  |
| --- | --- |
| ADRESSE MAC | ADRESSE IP |
|  |  |

* 1. Rappeler la différence entre une adresse IP et une adresse MAC.
* Empoisonner le cache ARP de la victime (client) et du serveur (mutillidae) à l’aide de la commande arpspoof –t
	1. Consulter puis noter à nouveau l’association du serveur web dans le cache ARP de la machine cliente victime. Que remarquez-vous ?

|  |  |
| --- | --- |
| ADRESSE MAC | ADRESSE IP |
|  |  |

Remarque ->

## Capture des trames

Par la suite le hacker peut facilement réaliser une capture de trames sur le protocole HTTP depuis la machine Kali. Lorsqu’un client s’authentifie sur l’application Mutillidae de la machine Metasploitable en HTTP, le pirate peut alors capturer le mot de passe saisi.

* Depuis la machine kali, ouvrir le logiciel Wireshark puis configurer une écoute sur le protocole HTTP.
* Depuis la machine cliente victime, se connecter au site Mutillidae. Créer un nouveau compte si cela est nécessaire.

* 1. À l’aide d’un analyseur de paquets depuis la machine kali du pirate, peut-on capturer en clair le mot de passe saisi par le client légitime ? Pourquoi cela est-il possible ?

Explications ->

* 1. Retrouver la requête contenant le mot de passe utilisé par le client lors de la connexion. Réaliser la capture écran de cette interception.

Capture ->

## Contre-mesures

1ère contre-mesure : chiffrement HTTPS

Le chiffrement des flux avec le protocole HTTPS n’empêche pas l’empoisonnement de cache ARP mais rend le flux capturé incompréhensible par l’attaquant.

2ème contre-mesure : inspection du cache ARP

Des outils permettent de contrôler les modifications du cache ARP afin de vérifier les modifications suspectes. On peut citer l’exemple de l’outil arpwatch.

* 1. En configurant un site en HTTPS, l’empoisonnement de cache ARP est-il toujours possible ?

Explications ->

* 1. Expliquer pourquoi il peut être important de surveiller les caches ARP de ses équipements.

Explications ->

* 1. Peut-on encore capturer le mot de passe en clair alors que les communications s’effectuent en HTTPS  ?

Explications ->

* 1. Conclure sur l’intérêt du chiffrement pour une entreprise.

Conclusion ->

## Réalisation d’une injections SQL

Pour tester l’injection SQL, vous devez vous positionner sur le menu « Owasp Top 10 → A1 Injection → SQLi Extract Data » → « User info » :

Un formulaire chargé d’afficher les informations de l’utilisateur qui se connecte est fourni pour réaliser un défi d’injection SQL :

Un attaquant va dans un premier temps essayer de récupérer des informations sur la structure de la base de données en tentant de provoquer une erreur de syntaxe au moment de l’exécution de la requête SQL utilisée pour valider l’authentification sur l’application.

Pour cela, un exemple d’identifiants peut être :

* login : admin
* mot de passe : ‘

Attention : la valeur du mot de passe est bien une apostrophe.

* Tenter une connexion avec ces paramètres.
	1. Expliquer en quoi les informations affichées peuvent être utiles pour un attaquant.

Explications ->

Les paramètres de connexion précédent (avec un caractère spécial, l’apostrophe) étant autorisés, l’attaquant sait qu’il va pouvoir tenter une attaque de type injection SQL.

Le principe est d’injecter une opération toujours vraie afin que la requête SQL exécutée au moment de la validation de l’authentification soit correcte même si le login n’existe pas dans la base.

Vous allez tester les identifiants suivants :

* login = hacker
* mot de passe = 'or 'a' = 'a
* Tenter une connexion avec ces paramètres.
	1. Expliquer l’affichage obtenu.

Explications ->

## Mise en place d’une contre-mesure

* Positionner le niveau de sécurité de l’application Mutillidae à 5 en cliquant deux fois sur le bouton « Toggle Security » (barre de menu haute).

 ⇒

* 1. Tenter à nouveau l’injection SQL précédente et expliquer le résultat obtenu.

Explications ->

Le code source gérant l’authentification sur l’application Mutillidae est dans la page « login.php » (dans /var/www/mutillidae).

* Comparer le code dans sa version sécurisée et dans sa version non sécurisée.
	1. Expliquer comment le code sécurisé de la page login.php permet d’empêcher l’injection SQL.

Explications ->

* 1. Rappeler quels sont les trois principes fondamentaux de la sécurité informatique.

Rappel ->

* 1. Proposer une conclusion sur l’intérêt d’un codage sécurisé au regard des principes fondamentaux de la sécurité.

Conclusion ->

# Annexes

## Comment fonctionne le sha256 ?

SHA-256 est une fonction de hachage cryptographique (un algorithme) qui permet d'obtenir l'empreinte numérique (hashcode - condensat) d'un fichier. Cette empreinte est, en théorie, unique, et jamais deux contenus ne peuvent produire le même condensat.

## Comment fonctionne l'algorithme MD5 ?

MD5 (Message Digest 5) est une fonction de hachage cryptographique qui calcule, à partir d'un fichier numérique, son empreinte numérique (en l'occurrence une séquence de 128 bits ou 32 caractères en notation hexadécimale) avec une probabilité très forte que deux fichiers différents donnent deux empreintes différentes.

Source : synonyme-du-mot.com

## Comparaison entre les algorithmes de chiffrement MD5 et SHA

Source : stackoverflow.com

## Le principe de Kerckhoffs

Le principe de Kerckhoffs a été énoncé par Auguste Kerckhoffs à la fin du XIXème siècle dans un article en deux parties « La cryptographie militaire » du Journal des sciences militaires. Ce principe exprime que la sécurité d'un cryptosystème ne doit reposer que sur le secret de la clé.

Autrement dit, tous les autres paramètres doivent être supposés publiquement connus. Il a été reformulé, peut-être indépendamment, par Claude Shannon : « l'adversaire connaît le système ». Cette formulation est connue sous le nom de la maxime de Shannon. Il est considéré aujourd'hui comme un principe fondamental par les cryptologues, et s'oppose à la sécurité par l'obscurité.

Le principe de Kerckhoffs n'implique pas que le système de chiffrement soit public, mais seulement que sa sécurité ne repose pas sur le secret de celui-ci. Une tendance plus récente est de considérer que quand les systèmes de chiffrement sont publics, largement étudiés et qu'aucune attaque significative n'est connue, ils sont d'autant plus sûrs.

## Présentation de Mutillidae

Mutillidae est une plateforme pédagogique mise en place par le groupe OWASP (Open Web Application Security Project) qui permet d'étudier les attaques associées aux applications web. Il s'agit de scripts PHP destinés à se familiariser avec les technologies web (protocole HTTP, AJAX…) et aux vulnérabilités qui en découlent.

Cet enseignement se fait à travers plusieurs activités qui ciblent chaque problème de sécurité du TOP10 d’OWASP (injection SQL, XSS, manipulation des en-têtes HTTP, vol de session…). Chaque activité donne accès à une page comportant un défi que l'utilisateur doit accomplir. Les scripts sont disponibles en version non sécurisée afin de tester les vulnérabilités. Des versions sécurisées permettent de vérifier que les attaques échouent avec un minimum de contrôles sur les données saisies dans les formulaires.

Ici, seule la partie sur la notion d’injection sera testée.


## Les applications web sont partout

Aujourd'hui, les applications web sont partout. Elles sont utilisées quotidiennement dans nos activités personnelles ou professionnelles (réseaux sociaux, achats en lignes, démarches administratives…). Toute entreprise ou administration se doit d'avoir un site web. Ces applications facilitent les échanges et les transactions car elles sont accessibles de partout à l'aide d'un simple navigateur sur un smartphone ou un ordinateur de bureau.

Si au début des sites web, les aspects techniques et fonctionnels étaient suffisants, ce n'est plus du tout le cas aujourd'hui. L'actualité nous rappelle régulièrement que des entreprises voient leur site web attaqué. Les conséquences peuvent être lourdes (perte de données, baisse du chiffre d'affaire, effondrement de la réputation…). Avec comme enjeu, la survie de l'entreprise selon la gravité de l'attaque subie.

En outre, Le règlement règlement général sur la protection des données (RGPD), mis en place au sein de l’Union Européenne, oblige les entreprises à assurer la sécurité des données personnelles qu’elles collectent.

## La sécurisation des applications web est indispensable

La sécurité des applications web est donc devenue un enjeu stratégique. Lors de son édition 2016, la société EY (http://www.ey.com/fr) a montré qu'une majorité des entreprises mondiales n'a pas de stratégie en matière de lutte contre les cybermenaces[[1]](#footnote-1)1.

Au delà de l'aspect fonctionnel des outils de développement, il est indispensable pour tout développeur de savoir identifier les vulnérabilités potentielles et de prendre en compte les menaces en adaptant son développement à l'aide de bonnes pratiques. La phase de test ne doit pas se limiter au fonctionnement attendu du code mis en œuvre mais elle doit aussi anticiper les utilisations malveillantes comme les injections de code SQL dans les formulaires.

Afin de mettre en place une veille stratégique sur la sécurisation des applications web, le groupe OWASP a développé une base de données qui recense la liste des incidents de sécurité recensés sur les applications web. Cette base nommée WASC-WHID (Web application Security Consortium - Web Hacking Database Project) permet de disposer de statistiques sur les failles de sécurité relevées sur les applications web. Les incidents sont déclarés et enregistrés afin d’alimenter une base de connaissance.

Le lien permettant d’accéder aux outils WHID est le suivant :

[*https://www.owasp.org/index.php/OWASP\_WASC\_Web\_Hacking\_Incidents\_Database\_Project*](https://www.owasp.org/index.php/OWASP_WASC_Web_Hacking_Incidents_Database_Project)

# Source ayant permis l’élaboration de ce TP

https://www.reseaucerta.org/

# Commentaires

Saisir des commentaires ci-dessous si vous souhaitez exprimer quoi que ce soit par rapport à votre travail.

Sujet adapté par Antoine DRIGET à l’intention de l’ESGI

-

Good Luck.

1. https://www.lemondeinformatique.fr/actualites/lire-55-des-entreprises-mondiales-n-identifient-pas-les-vulnerabilites-67146.html [↑](#footnote-ref-1)