Travaux Pratiques
Exploitation des applications Web

1. Objectifs

1.1. Audit d'une application Web

L'objectif de cette séance et de vous mettre dans la peau d'un auditeur, plus communément appelé « Pentester », et d'identifier les vulnérabilités présentes sur une application Web.

1.2. Configuration de la machine

Pour cette séance, vous aurez à votre disposition une VM configurée avec un serveur HTTP sur lequel est publiée une application Web vulnérable.

Pour accéder à l'application, rendez-vous à l'URL suivante :
http://ip-de-ma-vm

1.3. La démarche

Un Pentester, après autorisation de l'audit par le propriétaire de l'application, et signature de documents contractuels définissant l'objectif et les limites du Pentest (Obligation juridique), devra observer en générale la démarche suivante :

- Reconnaissance
- Identification des services
- Identification des vulnérabilités
- Exploitation des vulnérabilités
- Elévation de privilèges

A la fin du Pentest il vous est demandé de restituer un rapport d'audit permettant au commanditaire d'apprécier :

- Les tests réalisés
- Les vulnérabilités identifiées
- Les menaces associées
- Vos recommandations

Dans le cadre de notre audit, vous serez guidés, mais nous simulerons un Pentest de type boîte noire, durant laquelle la seule information qui vous aura été communiquée est l'adresse IP du serveur.

1.4. Outillages

Vous aurez à votre disposition une machine virtuelle Kali Linux intégrant en grande partie les outils permettant de faciliter l'audit :

- Firefox et les plugins suivants :
  - HackBar
  - Web developer ou Firebug
  - Live HTTP Headers
- Netcat
- Burp Suite
- Nmap
- Hydra
- JtR aka John The Ripper

1.5. Structure du lab

Nous allons découvrir les grandes familles de failles applicatives Web faisant partie du TOP 10 de l'OWASP. Cette séance nous permettra d'apprendre à exploiter les failles web afin de vous acculturer à la sécurité des applications web, aux risques inhérents aux mauvaises pratiques de développement, ainsi qu'à leurs impacts dans le monde de l'entreprise.

2. Exploitation des vulnérabilités

2.1. Reconnaissance

Pour pouvoir exploiter les vulnérabilités d'une application Web, il est avant tout important d'effectuer une reconnaissance réseau du serveur afin de déterminer les technologies, les services et les ports TCP qu'elle utilise.

Votre premier objectif est de scanner le serveur Web à l'aide de l'outil Nmap et d'identifier les services accessibles.

Les questions que vous devez vous poser :

- Quel est le port TCP utilisé par l'application Web ? La menace associée ?
- Quel est le Framework de développement sur lequel s'appuie l'application ? L'exploitation d'une vulnérabilité diffère selon la technologie utilisée.

Dans votre rapport, copiez le résultat du scan Nmap et mettez en évidence les technologies utilisées.

2.2. Identification des services

Nous venons de scanner le serveur Web et NMAP vous a remonté plusieurs services. Vérifiez à l'aide de l'outil Netcat les bannières de chacun des services, et listez leurs numéros de versions ainsi que les vulnérabilités connues associées qui vous semble les plus critique.

Service	Port TCP/UDP	Version	Vulnérabilités

2.3. Connexion à l'application Web

Connectez-vous à l'application Web et identifiez ses différents composants.

Les questions à se poser :

- Authentification utilisateur présente ? Si oui, à quel niveau ? Quel est la fonctionnalité protégée ?
- Est-il possible de se créer un compte invité ? Peut-on uploader des fichiers ?

Listez les liens qui vous semblent intéressants pour attaquer l'application…

2.4. Identification des bannières via les entêtes HTTP

Nous avons pu identifier précédemment les versions des différentes technologies utilisées par l'application, mais si celles-ci avait été protégées par un firewall, un reverse-proxy ou autre, vous n'auriez pas pu les identifier aussi facilement.

A l'aide de Burp ou de Firefox (plugins Live-http Headers ou Tamper Data), relevez au travers des entêtes HTTP de réponses que vous obtenez en navigant sur l'application, les technologies utilisées par l'application et les versions associées.

Les questions à se poser :

- Est-il normal de disposer des bannières des services ? Est-ce que ceci constitue une vulnérabilité ? Pourquoi ? Que recommanderiez-vous ?
- Disposez-vous du même niveau d'information que le scanner réseau ?

2.5. Identification du Back-End

Le but ici est de voir une autre manière d'identifier le backend, dans le cas où le scan réseau ne nous retournerait rien.
Nous avons vu dans le paragraphe précédent que nous n'avons pas pu identifier le back-end sur lequel s'appuie l'application. Les entêtes HTTP ne sont qu'une approche permettant d'identifier les services. Il existe bien d'autres manières : l'une d'elles consiste à interagir avec l'application pour lui faire générer des erreurs à partir de valeurs non attendues.

A vous de jouer ! Essayez de remplacer les valeurs par d'autres non attendues, et générez des erreurs pour identifier le type de base de données sur lequel s'appuie l'application.

Les questions à se poser :

- Pourquoi est–il important de disposer de la nature de base de données utilisée ?

2.6. OWASP - Injection SQL - Manuelle

Nous avons toutes les informations nécessaires nous permettant de commencer à "attaquer" l'application. Dans cet exercice, vous allez devoir tenter de trouver les vulnérabilités de type "injection SQL". Référez-vous au guide l'OWASP disponible sur Internet et au site pentestmonkey.org (cheatsheet SQL)

Les questions à se poser :

- A quoi ressemble la requête utilisée sur la page ?
- Quel est l'entrée et le résultat attendus ?
- Quels sont les caractères pouvant générer une rupture de la requête ?
- Comment modifier l'entrée attendue ? … etc…

Votre objectif, à travers différentes injections, est :

- de remonter le numéro de version de la base de données
- ainsi que le mot de passe du compte root

2.7. OWASP - Injection SQL - SQLMap

Vous venez d'exploiter manuellement une vulnérabilité de type "injection SQL" assez simple. Dans de nombreux cas, vous pouvez vous retrouver en présence de "blind injection" qui ne laisse pas entrevoir d'erreurs ni de résultat. Bien qu'il soit possible d'effectuer des centaines de requête manuellement, ceci est beaucoup trop chronophage. Il serait également possible d'écrire un script d'exploitation python, mais il y a plus rapide : l'outil SQLMap est un "Quick Win" pour ce qui est de l'exploitation d'injection SQL, dans la mesure où vous maîtrisez le fonctionnement et le comportement de l'outil.

Dans cet exercice, utilisez SQLMap pour exploiter la vulnérabilité et remonter le "hash" du mot de passe de l'utilisateur root. Attention, il vous est interdit d'utiliser l'option –all vous devez générer plusieurs requêtes avec une démarche logique pour récupérer les informations demandées.

2.8. OWASP – Local File Inclusion - /etc/passwd

L'inclusion de fichier local reste une vulnérabilité que l'on retrouve souvent dans les applications Web. Notre site Web en possède une, à vous de l'identifier et de l'exploiter. Votre objectif est d'afficher le contenu du fichier /etc/passwd.

Essayez également d'afficher le fichier /etc/shadow, ce qui est beaucoup plus intéressant.

Les questions à se poser :

- Comment identifier visuellement ce type de vulnérabilité ?
- Pourquoi n'est-il pas possible d'afficher certains fichiers ?

2.9. OWASP – Local File Inclusion – Fichier PHP

Exploitez la faille de type LFI pour afficher le contenu du fichier de configuration de la connexion à MySQL (« /connection.php »).

Les questions à se poser :

- Pourquoi ai-je une page blanche ?? Quelle est la particularité de ce fichier ?
- Comment faire pour que le contenu ne soit pas i…té ?

2.10. OWASP – Remote File Inclusion

L'inclusion de fichiers distants est une autre faille très appréciée au travers des pentests car elle a l'avantage d'offrir un plus large champ d'exploitation. L'idée est qu'au lieu de se contenter de récupérer du contenu local (comme vu ci-dessus), nous allons essayer d'appeler du contenu distant.

Identifiez à quel endroit de l'application vous pourriez inclure du contenu qui ne serait pas présent sur la machine mais stocké autre part.

2.11. OWASP – Remote File Inclusion

A vous d'identifier comment exploiter cette RFI, le but étant d'obtenir un shell distant sur votre serveur web.

Les questions à se poser :

- Comment créer un reverse-shell en php ?
- Comment inclure notre reverse shell dans l'application web pour qu'il soit exécuté et que nous récupérions une connexion sur le serveur ?

2.12. OWASP – File Upload - WebShell

Dans cet exercice, votre objectif est de tenter d'obtenir un "WebShell" au travers d'une vulnérabilité de type « File Upload ». Pour pouvoir exploiter une telle vulnérabilité il est nécessaire de disposer d'une fonctionnalité permettant d'envoyer des fichiers au travers du site Web. Bien souvent des mécanismes de sécurité simples ou complexes sont mis en places afin de vérifier les fichiers uploadés, et de bloquer les fichiers autres que ceux attendus.
A vous de jouer… ! Votre objectif est de créer un simple Webshell permettant, une fois uploadé, d'exécuter des commandes système sur le serveur afin de récupérer la version de l'OS ainsi que le nom de l'utilisateur utilisé pour l'exécution du serveur Web.

Les questions à se poser :

- Quels est le type de mécanisme mis en place ?
- Comment pourrait-on le détourner simplement ?
- Comment créer un webshell ?

2.13. OWASP - Command Execution – Simple

Notre site possède différents outils. En réfléchissant sur la manière dont ils pourraient fonctionner et la manière dont ils ont pu être codés, trouvez un moyen de détourner leur exécution initiale afin de pouvoir exécuter des commandes système sur le serveur.

Vous devez ici aussi retrouver l'identifiant de l'utilisateur sous l'identité duquel le serveur Web fonctionne.

2.14. OWASP - Command Execution – Reverse Shell

A l'aide de votre vulnérabilité type "Command Execution", trouvez le moyen d'ouvrir un reverse shell afin d'obtenir un accès interactif sur le serveur hébergeant l'application web.

2.15. OWASP – XSS Reflected

Dans cet exercice, vous allez devoir trouver où et comment exploiter une faille de type XSS. Ce type de vulnérabilité est très souvent utilisé car elle permet de créer des redirections ou de voler du contenu.

Les questions à se poser :

- Qui interprète le contenu d'une XSS ? (serveur/client)
- Avec quel langage exploiter une XSS ?
- Quelles sont les ouvertures possibles avec une telle faille ?

A l'aide des informations récoltées, tentez de voler le cookie de la session en cours.

2.16. OWASP – XSS Stored

A l'instar de la "XSS reflected", la "XSS stored" est réutilisable plusieurs fois puisque son contenu est intégré de manière plus ou moins permanente au site.
De la même manière que l'exercice précédent, trouvez le moyen de stocker votre XSS et volez le cookie de la session en cours.

2.17. .htaccess bypass http method

La section d'administration de ce site est protégée par un mécanisme de type htaccess.
Grâce aux exercices et informations récoltées jusqu'à présent, trouvez un moyen de d'outrepasser le blocage réalisé par le htaccess et accédez à la partie administration (située dans /admin).
Pour cette étape, vous aurez besoin de l'outil Burp.

2.18. Password .htaccess

Une autre vulnérabilité présente sur notre site, et également vue précédemment, vous permettrait de récupérer les mots de passes utilisés dans le mécanisme de protection htaccess de la section admin.
Vous devrez trouver un moyen de récupérer le hash de l'utilisateur admin, puis de le "casser" à l'aide des autres outils à votre disposition sur votre machine Kali linux.