2013/2014

M2 – Informatique - SSI

GULDNER Geoffrey
LACAVE Valentin
SOIHILI Abdou
HURIER Médéric

[CAHIER DES CHARGES - D'EXPLOITATION]

SOMMAIRE

I. Introduction

Dans ce rapport, nous allons vous présenter le travail global du groupe concernant le projet de synthèse de cette année. Pour ce faire, nous avons convenus que chacun rédige une partie concernant son travail et ses charges et que tout serait mis en commun à la fin; c'est pourquoi vous rencontrerez l'utilisation du « je » mais il ne se réfère pas tout le temps à la même personne.

Nous allons vous parler des choix qui ont été pris, testés et mis en production, des difficultés rencontrées et des solutions apportés à ces problèmes.

A la fin du rapport nous vous parlerons également de la nouvelle vulnérabilité que l'on a mis en place car celle prévue (Register globals) a été désactivé dans les nouvelles versions de PHP, il était donc important d'en trouver une nouvelle et de vous en faire part.

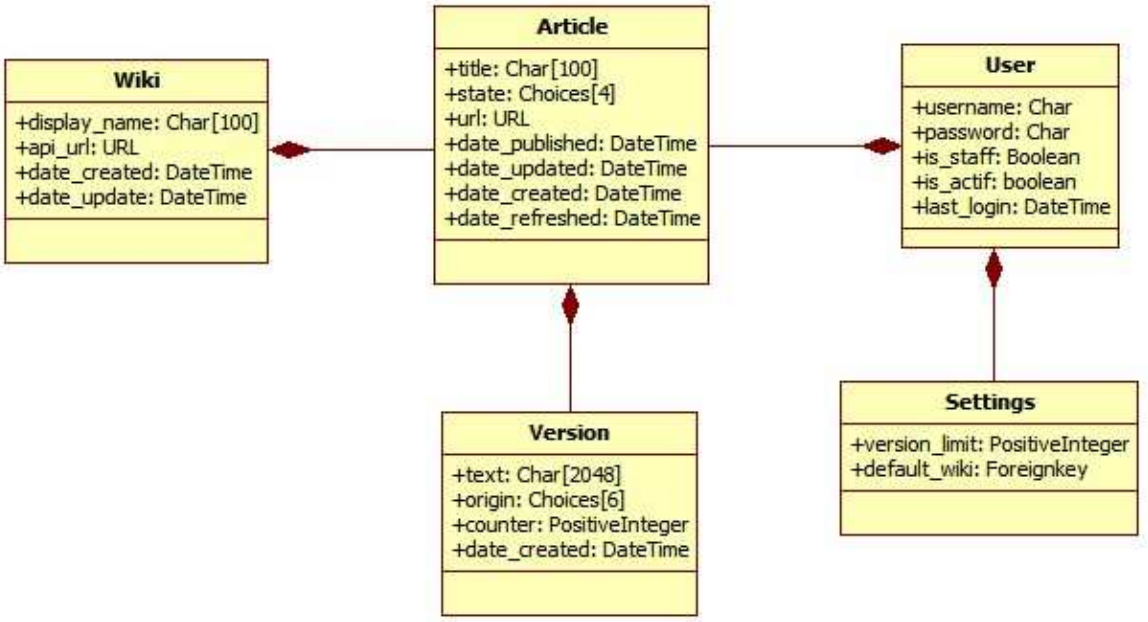
II. Développement de l'appli web

A. Rétro-conception

1) Structure logique

Le modèle de données que nous avons spécifié n'a subi que peu de modifications :

- ajout du champ « origin » dans la classe Version pour sauvegarder la façon dont l'article a été créé (création, résolution modification, rechargement, import,



2) Structure physique

Nous utilisons le cadriciel Web Django pour construire notre application web. Cet outil est très souple en termes de convention, il n'impose aucun nommage particulier sur les fichiers, les répertoires et les classes. Le développeur est invité à configurer ses outils selon ses besoins, mais nous avons tenu à suivre les recommandations du site officiel. Notre projet s'appelle WikiWikiWeb et il est composé d'une seule application : weditor.

Structure physique du projet :

- api/ — contient les APIs du site
  - mediawiki/ — API de Mediawiki
    - client.ini — configuration (authentification)
    - errors.py — exceptions renvoyées
    - lib.py — fonctions bas niveau
    - requests.py — fonctions haut niveau
    - tests.py — tests d'intégration
- database/ — base de données locales
  - db.sqlite — base de données de test
- public/ — dossier accessible depuis le serveur web
  - wsgi.py — configuration du montage de l'application web

- settings/ — configuration de l'application
  - common.py — commune à tous les environnements
  - dev.py — propre à l'environnement de développement
  - prod.py — propre à l'environnement de production
- weditor/ — application principale : éditeur Mediawiki
  - fixtures/ — données initiales
    - common.yaml — données communes (compte utilisateurs ...)
    - few.yaml — quelques données de test
  - static/ — interface cliente
    - css/ — thème graphique
    - images/ — images
    - js/ — fonctions d'interaction
  - templates/ — structure d'affichage du site
    - admin/ — propre à l'interface d'administration
    - layout/ — conteneur général (entête, pied de page)
    - pages/ — contenu des pages du site (accueil, connexion)
    - 404.html — page d'erreur HTTP 404
    - 500.html — page d'erreur HTTP 500
  - templatetags/ — fonctionnalités additionnelles aux templates
    - weditor_extras.py — fonctionnalités de l'application weditor
  - addons.py — fonctionnalités additionnelles aux contrôleurs
  - admin.py — paramétrage de l'interface d'administration
  - forms.py — formulaires de l'application
  - models.py — modèle de données (base de données)
  - views.py — traitements sur le site

B. Sécurisation

1) Protection contre les abus

Mise en œuvre d'un module robots :

- Problème : des utilisateurs malveillants peuvent créer des scripts afin d'automatiser la création d'article et saturer les ressources du serveur.
- Solution : nous avons ajouté un compteur de session pour vérifier le nombre de tentative de connexion. Au-delà d'un certain seuil, celui-ci est invité à compléter un captcha. Cette fonction est activée pour l'authentification et la création de compte.

Quotas utilisateurs :

- Problème : les utilisateurs peuvent ajouter un nombre illimité d'article à l'interface. Comme pour les services d'email en ligne (gmail.com, yahoo.com …), il faut soumettre les utilisateurs à des quotas.
- Solution : nous avons fixé une limite arbitraire de 10 articles par utilisateur. Cette limitation peut se justifier dans le cas d'un plan d'affaire et dans les conditions d'utilisations. De plus, un utilisateur doit sélectionner le nombre de version pour ses articles dans ses préférences (entre 2 et 20).

Mise en œuvre d'un module de protection contre les Frames:

- Problème : les attaques XSS exploitent beaucoup les éléments de type fenêtre (iFrames). Il faut éviter ce genre de dérive.
- Solution : notre outil fournit une option qui modifie les entêtes de notre serveur HTTP. Grâce à la directive X-Frame-Options 'Deny', nous avons désactivé l'affichage des pages dans des frames.

Filtrage des noms de domaine :

- Problème : un attaquant peut empoisonner le cache des navigateurs clients s'il arrive à modifier l'entête HTTP Host.
- Solution : nous avons configuré l'option ALLOWED_HOSTS de l'outil, ce qui permet de n'autoriser que les réponses pour un nombre limité d'hôte ou de domaine.

2) Erreurs et surveillance

Désactivation de l'affichage des erreurs :

- Problème : les erreurs en environnement de développement ne doivent pas s'afficher en environnement de production.
- Solution : en changeant la valeur de l'option DEBUG à False, nous avons désactivé entièrement l'affichage des erreurs en environnement de production.

Activation des logs spécifiques à l'application web :

- Problème : les erreurs de l'application doivent être remontées à l'administrateur.
- Solution : nous avons activé l'option LOGGING, et désigné un fichier pour stocker toutes les erreurs de production (/var/log/apache2/wikiwikiweb-django.log).

C. Installation de l'application web

1) Déploiement automatique

Comme pour le Wiki, nous avons mis en place un hôte virtuel avec une instance non sécurisée (port 9667) et une instance sécurisée (port 10000).

Pour automatiser le déploiement, nous avons utilisé le logiciel Fabric. C'est un outil en Python qui permet, à l'instar de Make, d'effectuer des tâches automatiques (déploiement, mise à jour de la base de données, lancement du serveur ...). Son avantage est de pouvoir fonctionner en local, et à distance via SSH. Les tâches de déploiement sont contenues dans le fichier WikiWikiWeb/fabfile.py.

Enfin, nous avons procédé à une revue de sécurisation en nous basant sur la documentation officielle en ligne.

2) Redirections automatiques

Une des contraintes du projet est d'autoriser la connexion sécurisée uniquement pour l'authentification, la création de compte et l'administration. Pour répondre à ce problème, nous avons utilisé le mode Rewrite d'Apache, qui permet de définir des redirections transparentes selon les critères de la requête.

Ainsi, nous avons identifié que pour les URLs contenant les actions connecte, inscris ou adm, il faut passer en HTTPS. Une règle inverse est également en place pour rebasculer vers le site non sécurisé.

Ces règles sont contenues dans les fichiers VirtualHost.

III. Mise en place de la BDD

Comme on le sait, les bases de données sont des éléments primordiaux dans un système. Dans ce chapitre, je vais détailler les méthodes que j'ai utilisé pour sécuriser notre base de données MySQL ainsi que les tables et les comptes dont on va se servir. (J'ai volontairement caché les noms des comptes utilisateurs, mots de passe et nom de DB grâce à des **********, en cas de diffusion non-volontaire du rapport)

A. Installation

Pour installer mysql, j'ai utilisé la commande :

- `apt-get install mysql`

B. Sécurisation

Par défaut, la configuration du service MySQL est loin d'être optimale, aussi bien en termes de performances qu'en termes de sécurité. Quelques réglages sont donc nécessaires.

1) Sécuriser l'identification de l'utilisateur root

La première étape est de se connecter à la base mysql en tant qu'administrateur :

- `$ mysql -u root mysql`

Changeons le mot de passe sur le compte root ou plus exactement définissons-le :

- `mysql> SET PASSWORD FOR root@localhost=PASSWORD('***************');`
- `mysql> RENAME USER root@localhost TO **********;`

2) Suppression des risque

Le second compte administrateur est lui aussi sans mot de passe et local. Il permet une connexion réseau depuis le serveur vers lui-même. Comme il est inutile, supprimons-le après s'être reconnecté :

- `mysql> DELETE FROM user WHERE Host='***************' AND User='root';`

Il faut noter que si on veut que l'administrateur puisse se connecter à distance, il faut que le `Host` du compte indique le nom ou l'adresse IP de son poste client ou bien le caractère `%` pour qu'il puisse se connecter depuis n'importe où. Des connexions anonymes sont possibles, des comptes sans mots de passe et des comptes qui sont autorisés à se connecter depuis n'importe où existent encore. Il nous faut donc les supprimer de sorte que seuls les utilisateurs dûment authentifiés puissent se connecter :

- `mysql> DELETE FROM user WHERE Password='';`
- `mysql> DELETE FROM user WHERE host = '%';`

MySQL a créé une base de test ainsi que des bases supplémentaires, elles nous sont inutiles :

- `mysql> DROP DATABASE test;`
- `mysql> DROP DATABASE .......;`

Par défaut, la base test et toutes les bases commençant par "test_" si elles existent sont accessibles à toute personne connectée :

- `mysql> DELETE FROM db WHERE db='test' OR db='test\_%';`

Quand MySQL démarre, le contenu des tables de privilèges est chargé en mémoire. Seules les modifications effectuées avec GRANT, REVOKE ou SET PASSWORD sont prises en compte immédiatement. Les autres changements comme par exemple un INSERT ou un UPDATE nécessite de recharger les tables de privilèges. Activons donc les modifications :

- `mysql> FLUSH PRIVILEGES;`

3) Création des BD et comptes utilisateurs

Un utilisateur peut avoir des privilèges s'appliquant à la totalité du système de base de données ou bien des privilèges sur certaines bases ou tables de la base. Seuls les comptes administrateurs doivent avoir des droits sur la totalité de la base (ou presque). De manière générale, ces comptes doivent être réservés aux tâches d'administration : création/suppression d'utilisateurs, de bases.

Pour créer un compte utilisateur, pouvant se connecter de n'importe où, avec le droit d'effectuer diverses commandes sur la base. Dans notre cas on effectue 2 fois les commandes car il nous faut 2 BD donc 2 comptes utilisateur. Chaque compte utilisateur a des droits sur sa propre DB et rien d'autre.

- `mysql> CREATE DATABASE **********`
- `mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,ALTER,DROP ON ********.* TO *********@"%" IDENTIFIED BY "**********";`

Une technique aurai été de créer 3 compte par DB :

- un avec le droit SELECT pour les opérations de lecture,
- un autre avec les droits supplémentaires INSERT, UPDATE, DELETE pour réaliser les opérations d'écriture,
- un dernier ayant aussi les privilèges CREATE, ALTER, DROP pour intervenir sur la structure des tables.

Mais pour éviter d'avoir trop de compte j'ai décidé d'en avoir que 3 (en comptant le compte root) pour pallier les risques de hack des comptes.

4) Visibilité des comptes

A l'aide de la commande SHOW DATABASES, les utilisateurs obtiennent la liste de toutes les bases de données du serveur.

- `$mysql -u ********** -p **********`
- `mysql> show databases;`

```
+----------+
| Database |
+----------+
| *****    |
| ******** |
| *********|
+--------------+
```

Vu notre paranoïa, on a autorisé à lister que les bases sur lequel l'utilisateur a des droits. J'ai donc ajouté dans le fichier `/etc/my.cnf`, section `mysqld`, cette ligne `safe-show-database`.

- `$mysql -u ********** -p **********`
- `mysql> show databases;`

```
+----------+
| Database |
+----------+
| *******  |
+----------+
```

5) Limiter les connexions

Le nombre de connexions est limité par le paramètre `max_connections`. Donc pour éviter qu'un utilisateur accapare toutes les connexions, nous limitons le nombre de connexions par utilisateur: `set-variable = max_user_connections = 500`.

III. Développement API cliente de wiki

A. Utilisation de l'API http rest de mediawiki

Le but a été de s'approprier l'utilisation de l'API de mediawiki afin de l'utiliser pour notre application. Pour cela, j'ai lu la documentation officielle disponible à l'adresse suivante :

- http://www.mediawiki.org/wiki/API:Main_page

Durant la lecture de l'api, j'ai remarqué beaucoup de points communs dans les commandes et dans la façon pour les envoyer à mediawiki ; j'ai donc décidé d'adapter mon code, que l'on verra un peu plus loin.

Pour faire simple l'API de mediawiki fonctionne selon ce principe :

$$Endpoint + Commandes$$

« L'endpoint » est simplement l'adresse de l'api du wiki (ex : http://en.wikipedia.org/w/api.php) Les commandes représentent donc ce que l'utilisateur demande, elles sont sous la forme :

$$action = login \,\&\, lgname = name \,\&\, lgpassword = password$$

B. Développement

Après avoir vu comment l'api fonctionnait, je me suis mis au développement de notre propre API de communication entre notre application et les wikis.

Sachant que les wikis fonctionnent tous sur la même base, il m'a été possible de tester directement mon API sur les vrais wikis et donc de gagner du temps sur la mise en place de l'environnement de test. Evidemment, je ne me suis pas amusé à tester toute l'API (notamment tout ce qui touche à la manipulation des articles) en dehors de l'environnement de test.

Les premières fonctions à être développées ont été celles qui permettent d'envoyer les requêtes aux sites distants tout en gérant les cookies (notamment pour le login), ces fonctions sont génériques, elles prennent un dictionnaire de paramètres ([action] = login, […]=…, … par exemple), un « Endpoint » et gèrent le formatage de la requête, l'envoie, la réception, le traitement des cookies, le retour de cette fonction est tout simplement la réponse du wiki.

Une fois la fonction d'envoi mise en place, j'ai commencé à développer les autres fonctions, login/logout, recherche d'articles, importation/exportations d'articles ainsi que toutes les fonctions qui viennent autour (gestion des tokens, demandés par les wikis pour toute modification/édition d'articles ou création de comptes,…).

Au fur et à mesure du développement, des tests unitaires ont été mis en place pour vérifier le bon fonctionnement des fonctions.

Grâce à la méthode de codage mis en place, il est facile de rajouter une fonction à notre API, voici la méthode générale utilisée :

Une classe qui contient l'adresse du wiki (changeable), la fonction d'envoi de la requête (celle vu au-dessus) et les fonctions de manipulation du wiki, sans oublier la classe d'exception, de test.

Chaque fonction propre à la manipulation du wiki as le même Template :

- Un dictionnaire comprenant les paramètres à envoyer (on peut mettre autant de paramètres que nécessaires dans le dictionnaire)
- On envoi au site
- On parse la réponse pour récupérer les informations voulues

Voici un exemple avec la fonction de recherche d'un article :

```
(self en python = this en java)
def search(self, title):
        ''' Retourne des suggestions d'article basé sur leur titre {title: {title:, description:, url:}, ...} '''

        #On remplit le dictionnaire avec les paramètres demandés par mediawiki
        result = []
        parameters = {}
        parameters['action'] = 'opensearch'
        parameters['format'] = 'xml'
        parameters['namespace'] = 0
        parameters['search'] = APIRequest.normalize(title)

        #On prévoit le namespace de la recherche (à cause des plugins des wikis)
        namespace = ' xmlns="http://opensearch.org/searchsuggest2"'

        #On envoie au wiki et on récupère sa réponse en supprimant le namespace pour plus de facilité
        #de traitement
        response = self.__POST(parameters).replace(namespace, '')

        #On parse en XML pour faciliter l'utilisation de la réponse
        xml = ET.fromstring(response)

        #On vérifie les erreurs renvoyées par wikimedia
        APIRequest.wiki_error_handler(xml)

        #On traite la réponse en se promenant dans le XML
        for item in xml.iter('Item'):
                title = item.find('Text').text
                description = item.find('Description').text
                url = item.find('Url').text
                suggestion = {'title': title, 'description': description, 'url': url}
                result.append(suggestion)
        return result
```

C. Sécurisation api

- Pour sécuriser l'API que j'ai écrit, des vérifications sur les paramètres en entrée sont faites (ex : vérifier les valeurs et leur types).
- Des fichiers de configurations (notamment pour les logins / passwords) contenant des paramètres en « dur » comme le login et mot de passe de nos bots permet une plus grande souplesse, facilité d'administration et évite que les mots de passes soient en clair dans le code (rappelons que le python n'est pas un langage compilé, le code est visible directement)
- Des tests unitaires ont été faits pour repérer d'éventuels bugs.
- Une défense en profondeur est mise en place (ex : mot de passe supérieur à 8 caractères vérifié lors de la saisie, mais aussi lors du passage des paramètres à l'API).

IV. Renforcement de la sécurité système

A. Installation du système d'exploitation

1) Partitionnement

Nous avons utilisé un partitionnement physique classique sur notre serveur.

- efi 512 MB, fat32, boot
- / 20 GB, ext4
- /home 20 GB, ext4
- /var 211GB, ext4
- /upload 100 GB, ext4
- /backup 100 GB, ext4
- /log 20 GB, ext4
- swap 8GB, swap

L'installation s'est bien déroulée, mais nous ne sommes pas arrivés à démarrer sur le système. Nous avons compris après plusieurs tentatives que le problème venait de l'EFI, une couche intermédiaire entre le système et le bios et présente sur les nouveaux ordinateurs. Pour contourner cette limitation, nous avons commencé par faire un partitionnement automatique, puis à faire nos modifications manuelles.

2) Configuration de la distribution

Lors de l'installation, nous avons renseigné plusieurs informations :

- la langue
- le mode d'entrée du clavier
- la configuration réseau
- le nom d'hôte
- le fuseau horaire
- l'utilisateur principal
- les paquets à installer par défaut

Nous avons choisi cette distribution car elle est minimaliste. Elle ne comporte pas de paquets et de services superflus après installation (interface graphique, dbus, langage de programmation …). Nous avons vérifié cette propriété, et effectivement, nous n'avons pas eu besoin de fermer des services et de supprimer de paquet. Nous avons simplement installé un système d'accès à distance pour administrer la machine à distance.

B. Sécurité physique

1) Retrait des bannières

Les bannières sont des informations textuelles affichées à l'utilisateur lors de sa connexion à un service (distant) ou à une session (physique). Elles peuvent fournir aux outils de scan des indications sur les composants installés et leurs versions.

Retrait de la bannière Linux :

- Problème : la bannière par défaut de Linux (/etc/issue) est affichée lors de la connexion de l'utilisateur et donne des informations sur le système d'exploitation.
- Solution : nous avons vidé le contenu du fichier.

Modification des entêtes HTTP:

- Problème : les entêtes HTTP sont accessibles depuis des extensions du navigateur et renseigne la version du serveur (Server) et l'outil utilisé (X-Powered-By).
- Solution : nous avons configuré la directive Apache ServerTokens et l'extension mod_security pour changer le contenu de ces entêtes afin d'induire les attaquants en erreur.

2) Restrictions d'accès aux partitions

Le système d'exploitation monte les partitions sur le système de fichier grâce à des points de montage. Ces points sont définis dans le fichier /etc/fstab, et des limitations peuvent être appliquées pour réduire les possibilités d'exploitation (lecture seule, interdiction d'exécution, désactivation de la synchronisation, accès en mémoire …).

Restrictions d'exécution :

- Problème : si un attaquant dispose des accès sur notre système, il peut télécharger ces propres outils et les exécuter librement.
- Solution : nous avons retiré les droits d'exécution sur plusieurs partitions : dossiers utilisateurs (/home), dossier de sauvegarde (/mnt/backup), dossier de téléversement (/mnt/upload), dossier temporaire (/tmp), dossier de surveillance (/var/log).

C. Sécurité du système

1) Renforcement des configurations

Le fichier /etc/sysctl.conf permet de configurer certaines fonctions du système d'exploitation. Il est très important de le consolider, car le système d'exploitation est la fondation de tous nos autres composants.

Directives de configuration système :

- désactivation du forward des paquets (net.ipv4.ip_forward = 0)
- désactivation du routage IP (net.ipv4.conf.all.accept_source_route = 0)
- activation de la protection contre l'IP Spoofing (net.ipv4.conf.all.rp_filter = 1)
- désactivation des redirections ICMP (net.ipv4.conf.all.accept_redirects = 0)
- activation de la surveillance des paquets forgés (net.ipv4.conf.all.log_martians = 0)
- désactivation des touches systèmes magiques (kernel.sysrq = 0)
- activation de la protection par cookies des TCP/SYN (net.ipv4.tcp_syncookies = 1)
- **désactivation de IPV6** (net.ipv6.conf.all.disable_ipv6 = 1)
- diminution du timeout des TCP/FIN (net.ipv4.tcp_fin_timeout = 15)
- diminution du keep-alive TCP (net.ipv4.tcp_keepalive_time = 1800)
- désactivation de la mise à l'échelle des fenêtres TCP (net.ipv4.tcp_window_scaling = 0)
- désactivation de TCP Sack (net.ipv4.tcp_sack = 0)
- désactivation des timestamps TCP (net.ipv4.tcp_timestamps = 0)
- activation du rejet des requêtes ICMP broadcast (net.ipv4.icmp_echo_ignore_all = 1)
- ne renvoie pas de redirections ICMP (net.ipv4.conf.all.send_redirects = 0)
- protection contre les messages malveillants (net.ipv4.icmp_ignore_bogus_error_responses = 1)
- augmente la taille de la queue des sockets (net.ipv4.tcp_max_syn_backlog = 1024)
- augmente le temps tcp-time-wait du la pool TCP (net.ipv4.tcp_max_tw_buckets = 1440000)

Nous avons prévu d'installer un IDS (snort) pour suivre les activités du système. Nous avons prévu d'installer chkrootkit pour surveiller les fichiers système et exécutables. Il permet de détecter les traces d'une attaque et de rechercher la présence d'un rootkit.

2) Restrictions du système de fichier

Tout notre système d'exploitation est représenté sous forme d'un fichier. Il est donc crucial de limiter l'accès au système de fichier et à son contenu.

Modification des permissions sur les répertoires critiques :

- Problème : certains dossiers contiennent des informations sensibles et accessibles à tous (other)
- Solution : nous avons retiré les permissions read, write et execute aux autres utilisateurs (other) sur les répertoires de configuration (/etc), d'exécution (/var), de sauvegarde (/mnt/backup) et aux dossiers utilisateurs (/home).

3) Mise en place du Pare-Feu

Nous avons utilisé le Pare-feu par défaut de Linux : Iptables. C'est un outil très complet, et nous l'avons configuré pour éviter les attaques sur nos services.

Règles du pare-feu :

- rejet par défaut de toutes les entrées (input)
- acceptation par défaut de toutes les sorties (output)
- autorisation pour les services autorisés (accès distant et service web)
- protection contre les paquets TCP forgés (TCP-flags)
- rejet des envois de type broadcast
- conservation des connexions déjà établies (entrée et sortie)
- activation de la surveillance de tous les parquets en entrée)

Rechargement des règles au démarrage :

- Problème : les règles ne sont pas conservées au démarrage de la machine.
- Solution : nous avons installé le service iptables-save et sauvegardés nos règles dans un fichier.

4) Attribution des hôtes

L'attribution des adresses IP sur le réseau est restreint, mais il est possible de contourner ses mesures. Un attaquant peut alors se faire passer pour notre machine.

Protection contre le Spoofing IP :

- Problème : un attaquant peut s'attribuer l'adresse de notre machine.
- Solution : les règles « order bind, hosts », « multi on », « nospoof on» dans le fichier /etc/host.conf permettent de limiter ses effets, en surveillant les machines qui ont la même adresse sur le réseau local.

D. Sécurité des services

1) Sécurité de l'accès à distance

L'accès à distance est un service critique sur notre système car c'est par ce moyen que nous pouvons l'administrer. Nous avons mis en place plusieurs restrictions pour garantir sa sécurité.

Configuration du service :

- changement du port par défaut (port 4812)
- limitation aux connexions IPV4 et avec un client supportant SSHv2
- limitation des utilisateurs autorisés à se connecter (4 membres du groupe)
- limitation du temps d'inactivité des sessions (5 min)
- ne pas tenir compte des préférences utilisateurs (forcer celle du système)
- augmentation de la verbosité des logs (INFO)

Accès par certificat :

- Problème : un mot de passe peut être attaqué par dictionnaire ou par force-brute.
- Solution : utilisation de certificats (clés asymétriques) et désactivation de l'accès uniquement par mot de passe (PasswordAuthentication no). On passe ainsi en **autorisation forte.** Il faut un mot de passe pour déverrouiller le certificat et se connecter au système, et un autre pour devenir super-utilisateur.

Limitation des tentatives de connexions :

- Prévision : nous avions prévu d'installer fail2ban pour limiter les tentatives d'accès et retarder les attaquants.
- Au final : le système d'authentification forte refuse immédiatement toute tentative de connexion frauduleuse. Nous avons préféré ne pas mettre de système de limitation pour ne pas risquer de bloquer nos comptes légitimes.
- A envisager : mettre en place un système de protection si des dénis de service sont détectés.

2) Sécurité du serveur web

Configuration du serveur web :

- adaptation des configurations de sécurité par défaut (apache2/conf-enabled/security.conf)
- adaptations des configurations de performance (apache2/conf-enabled/performance.conf)
- retrait des modules inutiles (Webdav, Userdir, Authbasic ...)
- configurations des logs dans chaque hôte virtuel (VirtualHost)

Activation des connexions sécurisées (HTTPS) :

- écoute des ports concernés (443 et 10 000)
- création d'un certificat SSL auto-signé (RSA:1048)
- configuration des hôtes virtuels (VirtualHost)
- mise en place des redirections HTTP vers HTTPS et vice-versa

Installation des modules de sécurité :

- installation de mod_security et des règles du projet « OWASP ModSecurity Core Rule Set Project ». Ce module peut être vu comme un IPS pour serveur web. Il détecte les actions dangereuses, et les rejettent si elles dépassent un certain seuil d'acceptabilité.
- Installation de mod_evasive pour protéger le serveur contre des actions trop lourdes ou trop fréquentes (DdoS).

Amélioration du module PHP :

- revue des directives de sécurité du module (/etc/php5/apache2/php.ini)
- désactivation des modules inutiles (sqlite, pdo …)
- installation d'accélérateur de site (php5-apc et php5-intl)

E. Gestion des utilisateurs

1) Création des utilisateurs

Notre équipe se compose de 4 membres. Nous avons donc crée un compte utilisateur pour chaque personne grâce aux commandes useradd et passwd. Nous n'avons pas eu besoin de créer d'autres comptes, les utilisateurs systèmes par défaut (www-data, mysql, ntp …) étant satisfaisants.

Étape de création des utilisateurs :

- création des utilisateurs : useradd –home <home_directory> -m <username>
- renseignement du mot de passe : passwd <username>
- sécurité des mots de passe : nous nous sommes engagés à utiliser un mot de passe de plus de 8 caractères, avec au moins une lettre minuscule, majuscule, un chiffre et un signe de ponctuation.
- informations complémentaires : le shell par défaut est bash, et à chaque utilisateur, un groupe de même nom est associé.

Désactivation du compte root :

- Problème : le compte root est un administrateur avec tous les droits. Son nom est le même pour toutes les distributions Linux, il est donc vulnérable à des attaques sur le mot de passe.
- Solution : sur notre distribution, le compte est désactivé par défaut.

2) Mise en place des masques

Un masque défini les permissions par défaut d'un fichier ou d'un répertoire crée par l'utilisateur.

Modification du masque par défaut :

- Problème : le masque par défaut est trop permissif (022). Il permet les écritures des utilisateurs autres (other).
- Solution : nous avons changé le masque par défaut pour tous les utilisateurs interactifs en ajoutant la commande umask 077 dans tous les fichiers ~/.profile.

3) Création des comptes super-utilisateur

Nous avons identifié dans le cahier des charges les raisons pour lesquelles un membre de l'équipe peut être amené à administrer le site :

- Médéric HURIER : configuration du serveur, administration du serveur web et déploiement du site.
- Geoffrey GULDNER : configuration de la base de données
- Mouchtali SOILIHI : renforcement du système
- Valentin LACAVE : configuration du Mediawiki et déploiement de l'API

Ces raisons sont toutes recevables. Nous avons donc attribué à chaque utilisateur les droits de super-utilisateur (sudo). Pour cela, nous avons utilisé la commande visudo.

Restrictions super-utilisateurs :

- Prévision : affiner le niveau de granularité des permissions en fonction besoins administratifs.
- Au final : nous avons donné à chaque membre du groupe les permissions maximales. Cela permet à chaque personne d'administrer la machine et d'être plus réactif en cas d'incident.
- A envisager : si un mot de passe est compromis, nous devrons nécessairement revoir notre attribution des rôles super-utilisateur.

F. Configuration des logs

1) Configuration du serveur de temps

Pour avoir des logs fiables, il est très important que l'heure du système soit réaliste. Un trop grand décalage entre les composants peut fausser les interprétations.

Configuration du démon NTP :

- Problème : synchroniser l'heure du système pour que celle-ci reste pertinente.
- Solution : installation du client ntp et lancement automatique au démarrage. Nous avons également revu le fichier de configuration /etc/ntp.conf.

2) Configuration du serveur de log

Les logs sont des données essentielles pour comprendre les attaques visant un système et améliorer les performances. Nous avons configuré le serveur de log pour centraliser efficacement ces évènements.

Étape de configuration du serveur de log :

- vérification des permissions sur le répertoire de stockage (/var/log)
- désactivation de l'accès réseau (UDP et TCP)
- changement des permissions de création par défaut (640 pour les fichiers et 750 pour les répertoires).
- changement des règles de log par défaut (suppression des règles inutiles et modification des fichiers associés)
- création d'une règle pour dupliquer toutes les erreurs dans un autre fichier (/var/log/error).

Étape de configuration des logs Apache et MySQL :

- Problème : les logs du serveur web et de la base de données sont critiques. Il faut surveiller à ce qu'ils ne nuisent pas aux performances ou qu'ils fuitent.
- Solution : nous avons configuré les services pour qu'ils gèrent eux même les logs plutôt que de les déléguer à MySQL. Sur MySQL, nous avons désactivé les logs génériques très couteux en ressources, et activé la surveillance des requêtes lentes (slow queries).

3) Rotation des logs

Rotation des logs :

- Problème : il est possible de saturer les ressources d'une machine si celle-ci ne fait que d'ajouter des évènements.
- Solution : nous avons utilisé logrotate pour paramétrer la rotation automatique des logs. C'est un démon qui compresse ou supprime les logs les plus anciens. Il est lancé toutes les semaines.

G. Mise en place des sauvegardes logiques

Sauvegarde automatique des configurations :

- Problème : si un incident mineur survient (ex : erreur de configuration), il faut pouvoir revenir à la version précédente.
- Solution : nous avons utilisé les logiciel duplicity et cron pour effectuer une sauvegarde hebdomadaire des dossiers /home et /etc. Les fichiers produits sont envoyés sur le serveur personnel de Médéric (freaxmind.pro) via FTP. Ils sont encryptés avec une clé asymétrique et compressé avant envoie.

V. Logiciel

A. Clonezilla

Au même titre que des programmes commerciaux (et payants) comme NORTON GHOST ou ACRONIS TRUE IMAGE, Clonezilla permet de sauvegarder disques ou partitions pour plus de sécurité et avoir une réinstallation plus facile. Comme tout LiveCd de Linux, ***Clonezilla*** est capable de booter sur un grand nombre de configurations et de reconnaitre beaucoup de périphériques. On peut réaliser une image ISO et la sauvegarder sur une partition de disque ou sur un CD boot ou encore clef USB. L'application supporte les systèmes de fichiers aux formats EXT2, EXT3, Reiserfs, XFS, JFS et LVM2, et effectue des copies de disque bloc à bloc, ce qui améliore la rapidité et les performances du clonage.

Nous n'allons pas nous lancer dans des explications sur l'utilisation du logiciel car un tuto existe et que celui-ci et très complet.

- http://wiki.mandriva.com/fr/Clonezilla_Live#Cloner_un_disque_dur_entier

Pour notre projet, nous avons fait un clone du système après installation du serveur et paramétrage de celui-ci, puis un second juste avant les vacances de noël (pour pallier le risque du « Mr le serveur marche plus on a tout perdu !!!! »). Grace à ce système si notre serveur venait à tomber, que cela soit dû à un problème hardware ou software, il suffira de prendre notre clef USB ou est stocké notre images du système puis prendre la seconde clef USB (bootable) contenant Clonezilla et de les insérer soit dans notre serveur de départ (si problème OS) ou dans un nouveau serveur (si problème matériel) et de faire une restauration.

B. Mediawiki

1) Configuration du logiciel

A notre surprise, Mediawiki était présent dans les dépôts de notre distribution. Nous avons donc opté pour le choix le plus facile et le plus sécurisé en optant pour l'installation via les dépôts. Cela permet de garder la cohérence entre les dépendances, et de recevoir automatiquement des mises à jours de sécurité.

Nous avons ensuite crée un virtualhost pour le site : un pour la version non sécurisée (/etc/apache2/site-available/mediawiki.conf) et un pour la version sécurisée (/etc/apache2/site-available/mediawiki-ssl.conf). Une fois le site en ligne, nous nous sommes rendus à l'adresse spécifiée par la documentation (http://172.24.141.120/mw-config/index.php) pour remplir un fichier de configuration (LocalSettings.php). Nous avons copié ce fichier à la racine du site, et veuillez à contrôler son accès. En effet, il contient l'ensemble des configurations du site !

Une fois le site accessible, nous avons suivi la procédure de sécurisation disponible sur le site officiel, et adaptées nos configurations et nos autres services en conséquence. Nous avons fait très attention aux permissions accordées à l'API (ex: création de compte).

2) Installation des extensions

La seule extension installée sur le site est OpenSearchXml. Elle nous permet d'effectuer des recherches grâce en l'API.

Nous avons étudié la possibilité d'installer des extensions de protection contre le vandalisme, mais ces dernières demandes un gros effort de suivi pour mettre à jour correctement les mots incorrects. Nous avons donc reporté leurs installations à plus tard.

3) Sécurisation

Pour sécuriser mediawiki, je me suis principalement basé sur ses mécanismes de défenses que j'ai fortement durci, voici la liste des paramètres que j'ai retouchés avec leurs nouveaux paramètres:

```
$wgGroupPermissions['*']['createaccount'] = true;
$wgGroupPermissions['*']['edit'] = false;
$wgAPIModules['createaccount'] = 'ApiDisabled';
$wgMaxArticleSize = 2048;
$wgUseAjax = false;
$wgAllowUserCss = false;
$wgAllowUserJs = false;
$wgWellFormatedXml = false;
$wgAccountCreationThrottle = 3;
$wgPasswordAttempsThrottle = array('count' => 3, 'seconds' => 1800);
$wgRestrictionTypes = array('create','edit');
$wgAciveUserDays = 3;
$wgAllowUserCssPrefs = false;
$wgInvalidUsernameCharacters = '&~#"{(|`\^@])°+=}"$£%ù*!§:/;.,?€';
$wgMaxNameChars = 15;
$wgMinimalPasswordLength = 8;
$wgAutocreatePolicy = false;
$wgCookieExpiration = 3600;
$wgCookieHttpOnly = true;
$wgAllDBsAreLocalhost = true;
```

Je pense que chaque nom est asser explicite afin de ne pas avoir à l'expliquer.
Pour augmenter la sécurité de mediawiki, je me suis intéressé aux plugins pour l'authentification en https, malheureusement, je n'en ai pas trouvé qui étaient compatibles avec notre version de mediawiki, car depuis quelques temps, il est possible de tout passer en https et tous les projets concernant la sécurité des pages a été abandonné depuis l'année dernière au profit de la totalité https. Or nous n'avons pas le droit de mettre du https autre que pour l'authentification. Je me penche actuellement sur la configuration de notre serveur web pour faire une redirection uniquement sur la page de connexion.

VI. Audit

Audit de la BDD

Revue des performances :

- But : pour améliorer la disponibilité du serveur, j'ai procédé à une amélioration des performances
- Solution : utilisation du script mysqltuner.pl pour isoler les directives à ajuster
- A envisager : le programme se base sur les précédentes requêtes, nous pourrons donc le relancer une fois le site déployé pour être plus précis.

Prévention des attaques par injection SQL :

- Prévision : nous avions prévu d'installer Green SQL pour nous assurer que les applications web ne laissent passer aucun injection SQL
- Au final : par manque de temps, nous n'avons pas pu déployer cette solution. L'intégration demande de nombreux tests que nous n'avons pas pu organiser.
- A envisager : les applications web (Mediawiki et Django) ont une couche d'abstraction qui filtre bien les entrées SQL. Si des injections sont détectées, nous mettre en place cette protection supplémentaire.

VII. Faille du système

A. Vulnérabilité

Notre API utilise un compte unique pour se connecter au Wiki, ce qui offre plusieurs avantages:

- Le premier est que le mot de passe des autres utilisateurs ne circule pas sur le réseau. Ils ne pourront donc pas être interceptés par des écoutes.
- Le second avantage est de pouvoir facilement renforcer la sécurité pour ce compte précis.

En conséquence, nous avons jugé inutile d'encrypter les communications sur le réseau. Si des attaquants interceptent le mot de passe du robot, ils pourront au pire créer ou modifier un article avec ce compte

Cependant, cela offre aux attaquants une vulnérabilité subtile. En supprimant le compte du robot, ils peuvent créer un déni de service sur notre site en empêchant la communication avec le Wiki.

Le mode opération est détaillé dans la partie suivante.

B. Exploitation

Un attaquant peut intercepter le mot de passe en clair grâce à une écoute sur une interface réseau. Il lui suffit ensuite de se connecter au Wiki pour supprimer le compte et exécuter la menace.

L'exploitation n'est pas difficile en elle-même, mais l'attaquant doit avoir assez d'imagination pour envisager cette attaque. Pour s'en protéger, il suffit de désactiver la suppression de compte. On peut aussi encrypter les canaux de communication.

C. Risques

Le risque de l'attaque est très faible sur notre système, mais une telle attaque pourrait causer des dégâts considérables contre un site marchant. Empêcher l'accès à une API, tel que l'extension de paiement, peut entraîner une baisser significative de la fréquentation ou de la réputation !

Ainsi, nous estimons que c'est une attaque bien adaptée au contexte de ce projet.

VIII. Conclusion

Comme nous l'avons vu, une grosse partie du travail prévu a été fait, testé et approuvé, l'application web développée fonctionne sans problèmes. Cependant, nous n'avons pas pu réaliser toutes les tâches fixées dans notre cahier des charges par manque de temps, ce qui vas surement poser problème : certains composants de sécurité n'ont pas été installés (composant non vitaux mais apportant un plus).