Compte-rendu
projet de synthèse

Valentin LACAVE <valentin.lacave@etu.univ-lorraine.fr>

Médéric HURIER <mederic.hurier@etu.univ-lorraine.fr>

Mouchtali SOILIHI <mouchtali.soilihi@etu.univ-lorraine.fr>

Geoffrey GULDNER <geoffrey.guldner@etu.univ-lorraine.fr>

2013-2014

Sommaire

Que contient ce document ?

Ce document est **le compte-rendu synthétique et final** des activités que nous avons réalisé dans le cadre **du projet de synthèse 2013-2014**. Pour rappel, ces activités ont été:

- le développement et le déploiement d'une application web permettant à des utilisateurs novices en informatique d'éditer le contenu d'un Wiki de façon simple et intuitive.
- l'attaque des sites des groupes concurrents tout assurance la défense du nôtre.

Au terme du semestre, nous avons rédigé et livré plusieurs documents:

- un cahier des charges fonctionnel et une présentation associée
- un cahier des charges technique et une présentation associée
- 10 rapports d'attaque
- 5 rapports de défenses

Une partie des informations de ces documents sera reprise et complétée dans celui-ci. Nous y ajouterons **notre analyse** sur ce qui s'est passé et **les conclusions** à en tirer.

Ce document est confidentiel et réservé uniquement à Mr Yann Lanuel et Mme Francine Herrmann.

Dans la partie 1, nous présentons l'architecture générale de l'application en terme de données (partie 1.1) et de traitements (partie 1.2). Nous donnons également l'infrastructure (partie 1.3).

Dans la partie 2, nous développons les éléments techniques que nous avons choisi pour concevoir notre application (partie 2.1) et les moyens de sécurisation (partie 2.2 et partie 2.3).

Dans la partie 3, nous détaillons les attaques que nous avons réalisé sur les sites concurrents.

Dans la partie 4, nous donnons les attaques que nous avons détecté sur notre site.

Dans la partie 5, nous dressons le bilan de ce projet et ce qu'il nous a apporté.

En cas de problème, n'hésitez pas à nous contacter aux adresses suivantes:

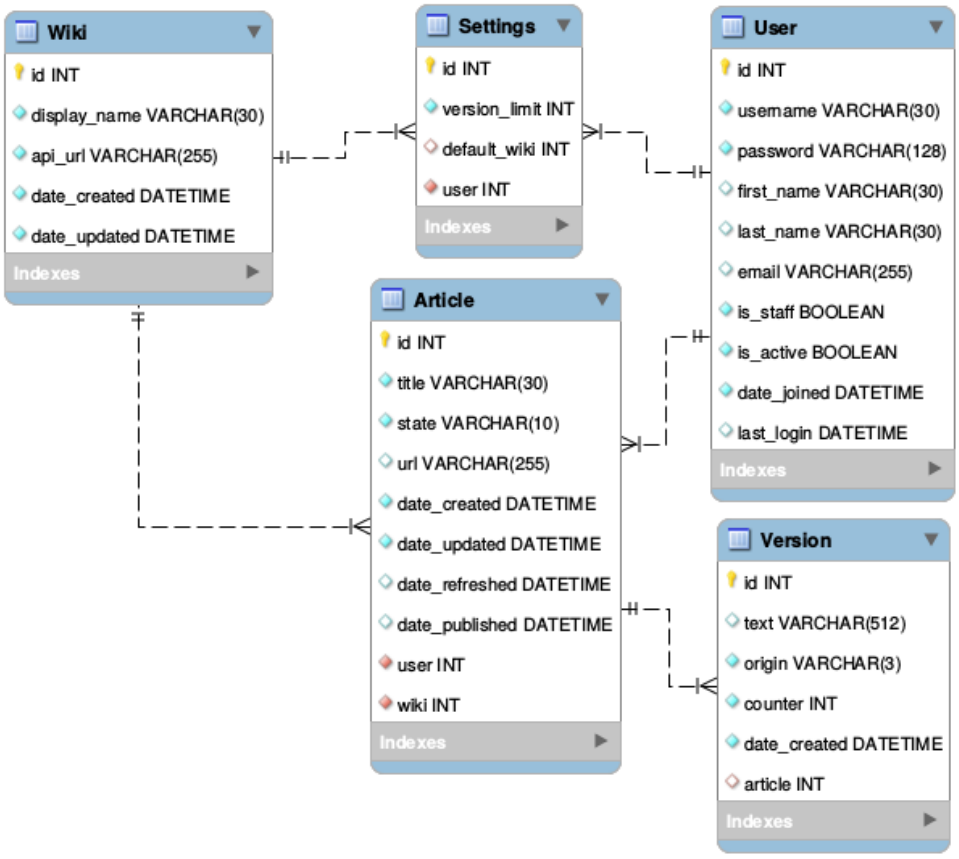
mederic.hurier@etu.univ-lorraine.fr
valentin.lacave@etu.univ-lorraine.fr
geoffrey.guldner@etu.univ-lorraine.fr
mouchtali.soilihi@etu.univ-lorraine.fr

1. Architecture générale

1.1 Modèle de données

1.1.1 Schéma de la base de données

Le schéma suivant représente la structure de la base de données telle qu'elle est en production
Les tables systèmes liés au cadriciel (Django) ne sont pas présentes sur le dessin



Notre système peut communiquer avec plusieurs Wikis enregistrés dans **la table Wiki**. L'entité contient un nom d'affichage destiné à l'utilisateur (display_name), une URL vers le le script de l'api (api_url), une date de création (date_created) et une date de modification (date_updated).

Les utilisateurs et les administrateurs sont enregistrées dans la **table User**. Ils sont identifiés par un nom d'utilisateur (username), reconnus par un mot de passe (password) et décrits par un prénom (first_name), un nom (last_name) et une adresse mail (email). Le drapeau is_staff permet d'indiquer si l'utilisateur est un administrateur, et le drapeau is_active si l'utilisateur est autorisé à se connecter. Enfin, la table possède une date d'inscription (date_joined) et une date de dernière connexion (last_login) pour suivre l'activité de l'utilisateur.

Les configurations utilisateurs sont stockées dans la **table Settings**. Elle est composée d'un champ indiquant le nombre de version d'article que l'utilisateur souhaite conserver (version_limit) et le Wiki par défaut dans lequel seront crées et importées les articles (default_wiki). La clé étrangère permet de relier cette table à la table User (avec une cardinalité 1-1).

Les informations permettant de décrire un article sont stockées dans la **table Article**. Les champs sont: le titre de l'article (title), l'état dans lequel l'article se trouve (ex: nouveau ...) (state), l'URL vers l'article sur le Wiki (url), une date de création (date_created), une date de modification (date_updated), une date de dernière mise à jours (date_refreshed) et une date de dernière publication (date_published). Cette table est associée au Wiki dans lequel l'article est géré (wiki) et à l'utilisateur qui a crée ou importé l'article (user).

Enfin, la **table Version** contient les versions des textes datés et associées aux articles (article). La table se compose du texte de la version (texte), de l'évènement qui a crée la version (ex: import, création …) (origin), d'un compteur interne et auto-incrémenté permettant de donner un numéro cohérent à la version (counter) et d'une date de création (date_created).

Comparé au modèle que nous avons réalisé dans le cahier des charges techniques, environ 80% des informations sont identiques. Les différences sont:

- les URL liées au Wiki (api_url et search_url) ont été combinées en une seule, car il n'y a qu'un point d'entrée pour assurer ces deux fonctions.
- les champs énums donnant le rôle et l'état de l'utilisateur (role et state) ont été transformées en booléen pour simplifier la structure (is_staff et is_active).
- le champ last_version de la table Article a disparu car on peut retrouver aussi rapidement la dernière version associée à un article directement depuis la table Version
- la table Version a été complétée par deux champs:
    - origin: permet de retrouver comment la version a été créée (import, création …)
    - counter: permet d'afficher un numéro interne (1, 2, 3 …) plutôt que l'ID (1, 245 …)

Parmi les tables systèmes créées par notre outils de développement, on peut citer:

- les tables permettant de gérer les permissions selon un modèle RBAC
    - auth_permission, auth_user_groups, auth_user_user_permissions
- les tables servant à assurer le service Single Sign On:
    - social_auth_association, social_auth_code, social_auth_usersocialauth
- la tables de journalisation des activités administrateurs: django_admin_log
- la table permettant de sauvegarder les sessions HTML: django_session
- la table servant à enregistrer les captchas envoyés: captcha_captchastore

Toutes ces tables ont été créées automatiquement par l'outil de génération du cadriciel.

Pour faciliter les tests, nous avons également crée des données initiales (fixtures) rechargeable facilement, automatiquement et à l'infini.

1.1.2 Contraintes d'intégrité

Dans une base de données, une contrainte d'intégrité permet de garantir la **cohérence** des données lors des mises à jour. Les règles suivantes sont celles implémentées en production.

Table	Champ	Type	Contraintes
Wiki	TOUS	/	• édition réservée à l'admin
	id	INT	• clé primaire auto-incrémentée
	display_name	VARCHAR(30)	• ^[0-9a-zA-Z-]*$ • non nul • unique
	api_url	VARCHAR(255)	• est une URL • non nul
	date_created	DATETIME	• non nul • édition manuelle impossible
	date_updated	DATETIME	• non nul • édition manuelle impossible
User	id	INT	• clé primaire auto-incrémentée
	username	VARCHAR(30)	• ^[\w.@+-]+$ • unique • non nul
	password	VARCHAR(128)	• chiffré (algorithme PBKDF2) • non nul
	first_name	VARCHAR(30)	/
	last_name	VARCHAR(30)	/
	email	VARCHAR(255)	• est un email
	is_staff	BOOLEAN	• non nul • par défaul: faux
	is_active	BOOLEAN	• non nul • par défaut: vrai
	date_joined	DATETIME	• non nul • édition manuelle impossible

	last_login	DATETIME	● édition manuelle impossible
Settings	id	INT	● clé primaire auto-incrémentée
	version_limit	UNSIGN. INT	● non nul ● valeur entre 2 et 10 ● par défaut: 10
	default_wiki	INT	● clé étrangère => WIki (1*n)
	user	INT	● clé étrangère => User (1*1) ● non nul ● édition manuelle impossible
Article	id	INT	● clé primaire auto-incrémentée
	title	VARCHAR(30)	● ^[0-9a-zA-Z-]*$ ● non nul
	state	VARCHAR(10)	● non nul ● valeurs parmis: OK, NEW, DRAFT et CONFLICT ● par défaut: OK
	url	VARCHAR(255)	● est une URL ● édition manuelle impossible
	date_created	DATETIME	● non nul ● édition manuelle impossible
	date_updated	DATETIME	● non nul ● édition manuelle impossible
	date_refreshed	DATETIME	● édition manuelle impossible
	date_published	DATETIME	● édition manuelle impossible
	user	INT	● clé étrangère => User (1*n) ● non nul ● édition manuelle impossible
	wiki	INT	● clé étrangère => Wiki (1*n) ● non nul

Version	id	INT	• clé primaire auto-incrémentée
	text	VARCHAR(512)	/
	origin	VARCHAR(3)	• non nul • choix parmis: CREATION, RESOLUTION, IMPORT, MODIFICATION, REVIEW et DOWNLOAD
	counter	UNSIGN. INT	• non nul • par défaut: 1
	date_created	DATETIME	• non nul • édition manuelle impossible
	article	INT	• clé étrangère => Article (1*n) • édition manuelle impossible

Les contraintes d'intégrité suivantes sont communes à toutes les tables de la base de données:

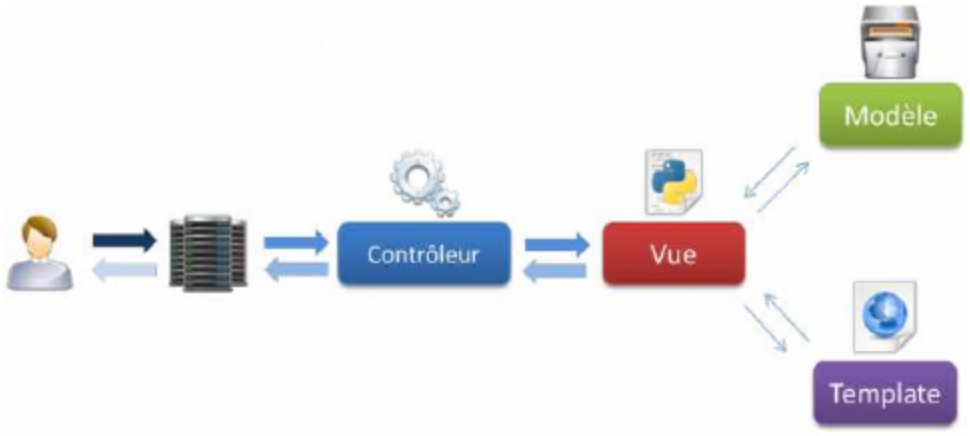
- l'encodage par défaut est **UTF-8**
- les modifications et suppressions sont effectuées en cascade
- le moteur de la base de données MySQL est **InnoDB** car il supporte les clés étrangères

1.2 Traitements

1.2.1 Patrons de conception

Un patron de conception est un arrangement caractéristique de modules, reconnu comme bonne pratique en réponse à un problème de conception d'un logiciel. Il décrit une solution standard, utilisable dans la conception de différents logiciels (tel que les applications web).

Dans le cadre de notre site, nous avons implémenté un patron de conception **MVC** qui peut se décrire de la façon suivante:



- **L'utilisateur** communique avec notre application via son navigateur web. C'est la seule entité capable de déclencher un évènement (clic sur un lien, envoie de formulaire …).
- **Les contrôleurs** contiennent la logique métier de l'application. Suite à un évènement, le dispatcher diffusent la requête au morceau de code concerné. Ensuite, on vérifie les permissions de l'utilisateur et la cohérences des données saisies avant d'exécuter les actions du traitement. C'est aussi ce module qui gère les retours (corrects ou inopinés).
- **Le modèle** est une couche d'abstraction de la BDD (DAL: Data Abstraction Layer). Elle fait office d'interface avec celle-ci, en décrivant de manière plus intégrée la structure. Elle effectue également des contrôles supplémentaires (expression régulière, URL …)
- **les templates** sont des fichiers HTML permettant de décrire la logique de présentation. En plus du langage de base, un système de balise propre au cadriciel permet d'intégrer des structures liées à la programmation (boucle, condition, déclaration …).

1.2.2 Liste des traitements

Les traitements sont des suites d'instruction qui, à partir des données en entrée (requête), génèrent un résultat affichable (modification de la page) et durable (modification de la base).

Le tableau suivant décrit tous les traitements possibles par notre application:

Nom	en entrée	vérifications	en sortie
Index (OK)	/	● si l'utilisateur n'est pas connecté	affichage Index
(OK)	/	● si l'utilisateur est connecté	redirection Home
Home (OK)	/	● l'utilisateur est connecté	affichage Home
(ERREUR)	/	● l'utilisateur n'est pas connecté **(idem pour les pages suivantes)**	redirection Login
Search (OK)	recherche ID wiki	● peut créer un article (<seuil) ● le wiki est référencé	recherche via API enregistre req. affichage Search
(OK)	/	● peut créer un article ● une requête est enregistrée	rejoue req. affichage Search
(ERREUR)	*	● ne peut pas créer d'article	redirection Home affichage erreur
(ERREUR)	/	● pas de reqête enregistrée	redirection Home affichage erreur

(ERREUR)	recherche	● manque ID wiki	redirection Home affichage erreur
(ERREUR)	recherche ID wiki	● le Wiki n'est pas référencé	affichage 404
Link (OK)	ID article	● une requête est enregistrée ● peut créer un article ● l'ID de l'article est référencée ● le titre est unique pour l'util. ● longueur < 512	récupération article via API création Article création Version redirection Home affichage OK
(ERREUR)	*	● aucune requête enregistrée	redirect Home affichage erreur
(ERREUR)	*	● ne peut pas créer d'article	redirect Home affichage erreur
(ERREUR)	*	● ID article non référencé	redirect Search affichage erreur
(ERREUR)	ID article	● titre non unique pour l'util.	redirect Search affichage erreur
(ERREUR)	ID article	● longueur > 512	redirect Search affichage erreur
Create (OK)	formulaire	● peut créer un article ● l'article est valide ● le titre est unique pour l'util. ● longueur < 512	création Article création Version redirect Home
(OK)	/	● peut créer un article	affichage Create
(ERREUR)	formulaire	● ne peut pas créer d'article	redirect Home affichage erreur
(ERREUR)	formulaire	● l'article est invalide	affichage Create affichage erreur
(ERREUR)	formulaire	● l'utilisateur possède un autre article avec ce titre	affichage Create affichage erreur
(ERREUR)	formulaire	● longueur < 512	affichage Create affichage erreur

Edit (OK)	ID article formulaire	● l'article est référencé ● user(article) == user(session) ● longueur(versions) >= 1 ● longueur(texte) > 512	mise à jour article: état, date création Version nettoyage ancienne version redirect Edit affichage OK
(OK)	ID article	● l'article estréférencé ● user(article) == user(session)	affichage Edit
(ERREUR)	*	● l'article n'est pas référencé	erreur 404
(ERREUR)	ID article	● user(article) != user(session)	erreur 404
(ERREUR)	ID article	● longueur(versions) < 1	redirect Home affichage erreur
(ERREUR)	ID article	● longueur(texte) < 512	affichage Edit affichage erreur
Review (OK)	ID article ID version	● l'article et la version existe ● article(user) !! article et version(article) == version ● longueur < 512	création Version mise à jour article conflict => draft redirection Edit affichage OK
(ERREUR)	/	● manque ID article ou ID version	erreur 404
(ERREUR)	ID article ID version	● l'article ou la version n'existe pas	erreur 404
(ERREUR)	ID article ID version	● article(user) != article ou version(article) != version	erreur 404
(ERREUR)	ID article ID version	● longueur texte > 512	redirection Edit affichage erreur
Refresh (OK)	ID article	● l'article existe ● user(article) = user(session) ● article.state IN [draft, conflict] ● longueur < 512	récupération article via API mise à jour article: état, date création Version nettoyage anciens articles redirect Edit affichage OK
(ERREUR)	/	● manque ID article	erreur 404

(ERREUR)	ID article	● user(article) != user(session)	erreur 404
(ERREUR)	ID article	● article.state NOT IN [draft, conflict]	redirect Edit affichage erreur
(ERREUR)		● longueur > 512	redirect Edit affichage erreur
Publish (OK)	ID article	● l'article existe ● user(article) == user(session) ● article.state IN [draft, new]	récupération article via API mise à jour article: état, date redirection Edit affichage OK
(ERREUR)	/	● manque ID article	erreur 404
(ERREUR)	ID article	● artice non référencé OU user(article) != user (session)	erreur 404
(ERREUR)	ID article	● article.state NOT IN [draft, new]	redirect Edit affichage erreur
(ERREUR)	ID article	● conflit lors de MAJ	mise à jour article état=conflit redirection Edit affichage erreur
Delete (OK)	ID article	● l'article existe ● l'article appartient à l'utilisateur	suppression de l'article redirection Home
(ERREUR)	ID article	● l'article n'existe pas ou l'article n'appartient pas à l'util .	erreur 404
Profile (OK)	formulaire	● le formulaire est valide	modification User affichage Profile affichage OK
(OK)	/	/	affichage Profile
(ERREUR)	formulaire	● le formulaire n'est pas valide	affichage Profile affichage erreur
Settings (OK)	formulaire	● le formulaire est valide	modification Set. affichage Set. affichage OK
(OK)	/	/	affichage Set.

(ERREUR)	formulaire	● le formulaire n'est pas valide	affichage Set. affichage erreur
Subscribe (OK)	formulaire	● l'utilisateur n'est pas connecté ● le formulaire est valide	création User redirection Home
(OK)	/	● l'utilisateur n'est pas connecté	affichage Subscribe
(OK)	*	● l'utilisateur est déjà connecté	redirection Home
(ERREUR)	formulaire	● l'utilisateur n'est pas connecté ● le formulaire n'est pas valide	affichage Subscribe affichage erreur
Login (OK)	formulaire	● l'utilisateur n'est pas connecté ● formulaire valide ● l'utilisateur existe ● l'utilisateur est actif ● l'utilisateur s'est authentifié	création session redirection Home
(OK)	/	● l'utilisateur n'est pas connecté	affichage Login
(OK)	/	● l'utilisateur est déjà connecté	redirection Home
(ERREUR)	formulaire	● le formulaire n'est pas valide	affichage Login affichage erreur inc. spam
(ERREUR)	formulaire	● l'utilisateur n'existe pas ou n'est pas actif ou n'est pas authentifié	affichage Login affichage erreur inc. spam
Logout	/	/	déconnexion util. redirection Index

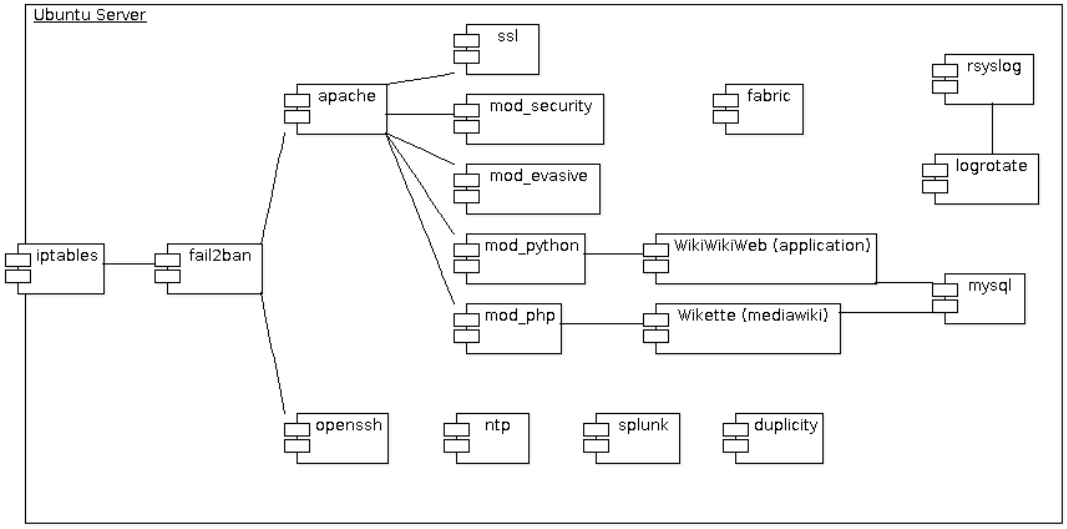
En plus des traitements que nous avons réalisé nous mêmes, 3 sont fournis par le cadriciel:

- **la gestion des captcha** (django-simple-captcha) permet de générer et de valider les captcha ajoutés aux formulaires de connexion et d'inscription.
- **la gestion du Single-Sign On** (django-social-auth) effectue les demandes d'authentification auprès des services tiers concernés (Google, Yahoo, OpenID …).
- **la gestion du site** permet aux administrateurs de gérer facilement le contenu de la base données de l'application (wiki, utilisateurs, permissions configurations, articles, versions).

1.3 Infrastructure

1.3.1 Composants

Pour répondre aux besoins du sujet, nous avons mis en place **une infrastructure très simple** basée sur un système **ouvert** et **open-source** (Ubuntu Server 13.10).



Le schéma suivant donne les composants de cette infrastructure et leurs associations:

Les rôles et les qualités de ces composants sont détaillés dans l**a partie suivante** (voir 2.1). Nous allons vous présenter ici les arguments qui ont motivé ce choix d'architecture.

Le système d'exploitation (Ubuntu Server 13.10) est **une distribution Linux** qui se veut aussi stable que le système Debian sur lequel elle est basée, mais avec des mises à jours de version plus fréquente afin de donner une logithèque plus récente. Nous avons besoin des derniers paquets stables pour notre environnement de programmation et notre Wiki. Aussi, nous avons préféré choisir le système qui nous donne le meilleur compromis entre **nouveauté** et **stabilité**.

Les systèmes Linux ont l'avantage d'être **open-source**, ce qui permet de comprendre leur fonctionnement et de l'adapter à notre avantage. C'est un atout pédagogique et de sécurité majeur. De plus, ils **supportent de nombreuses technologies** et sont **par défaut minimaliste**. Windows nous semble plus plus lourd et avec une plus grande surface d'attaque.

Nous avons choisi **la famille Ubuntu/Debian** notre projet car elle dispose d'une **communauté très actives** (forums, tutoriels, IRC …) et d'une **forte popularité** (voir Distrowatch). Les autres alternatives basées sur Redhat ou Suse semblent plus adaptées dans un contexte corporate, car elles disposent d'un support de qualité et de solutions intégrées (Active Directory, environnement de programmation Java, RADIUS …). Cependant, ces solutions ne nous intéressaient pas dans le cadre du projet. Nous voulions un système **rapide** et **simple** à utiliser.

Ce système a été installé **physiquement** sur la machine, plutôt que virtuellement. En effet, bien que les avantages de la virtualisation sont nombreux, nous n'avions pas assez de retour sur la sécurité et la fiabilité de ces architectures pour nous baser dessus. C'est un choix de **prudence**.

Vu qu'une seule machine était à notre disposition, nous avons concentré tous les services sur celle-ci (base de données, service web, authentification …). Dans un contexte réel, nous aurions réparti ces services sur plusieurs machines (ex: Architecture 3-tiers).

Au final, notre système se compose d'un **pare-feux** (iptables) qui protège notre **service web** (apache) et **d'accès à distance** (openssh). Le service web comporte plusieurs modules pour supporter les **environnements de programmation** (Python, PHP), et des fonctions de sécurité (mod_security, mod_evasive, SSL). Il propulse les **2 applications web** (WikiWikiWeb et Wikette), et ces dernières utilisent pour gérer la persistance de données **un SGBD** (MySQL). Enfin, de nombreux services complémentaires viennent s'ajouter comme un système de **sauvegarde logique** (duplicity), de **journalisation** (rsyslog/logrotate), de **supervision** (splunk), de **synchronisation du temps** (ntp) et de **déploiement** (fabric).

Les différences par rapport à l'infrastructure initiale sont:

- **suppression du composant PHPMyAdmin**: plutôt que d'ajouter un composant supplémentaire (et augmenter la surface d'attaque), nous avons préféré utiliser le client MySQL par défaut (CLI) pour faire nos requêtes. Cela a été rendu possible par la génération automatique des requêtes (wiki et app.).
- **remplacement du service web Nginx par Apache**: le support de Mediawiki pour Nginx étant limité, nous avons préféré utilisé le système le plus support: Apache.
- **ajout du logiciel de supervision Splunk**: les premières attaques que nous avons subis ont révélé un manque de disposition pour surveiller notre système. Nous avons donc installé Splunk pour suivre l'activité de notre système et des sites web.
- **ajout de l'outil de synchronisation du temps client NTP**: suite à nos cours, nous avons compris l'importance d'avoir des rapports cohérents en terme de temps.
- **ajout de l'outil de sauvegarde logique Duplicity**: pour tenir compte des modifications depuis l'installation initiale, nous avons configuré un service de sauvegarde logique en complément de notre outil de sauvegarde physique (CloneZIlla). Nous pouvons ainsi limiter la perte des opérations perdues en cas de panne.
- **Ajout de l'outil de déploiement Fabric**: pour éviter les erreurs de manipulation lors d'une nouvelle version du site, nous avons utilisé Fabric pour automatiser cette tâche.
- **absence des logiciels IDS Snort et de pare-feu SQL GreenSQ**L: par manque de temps, nous n'avons pas installé ces logiciels. Ils n'étaient pas jugés critiques dans notre installation, car leur service sont redondants (mod_security et fonctions cadriciel).

1.3.2 Communication inter-services

Une des contraintes importantes imposée par le projet a été **d'utiliser le réseau de l'université** pour accéder et héberger nos machines. Cela pose plusieurs problèmes:

- **il est possible d'écouter le trafic réseau facilement**: les attaquants ayant accès au réseau local, il leur est possible de procéder à des écoutes simplement grâce à des techniques comme ARP Spoofing.
- **il est possible de connaître l'adressage des machines et de l'usurper**: en changeant simplement l'adresse MAC de sa machine, l'attaquant peut se faire passer pour l'une de nos machines. Le blocage par IP est donc totalement inefficace (et interdit).
- **Il est possible d'usurper l'adresse mail d'un étudiant ou d'un professeur**: le serveur SMTP de l'université ne demandant aucune authentification, toutes les adresses du domaines sont falsifiables.
- **Les attaques de type DoS peuvent être très rapides**: grâce à la bande passante qu'offre le réseau local, il est possible de faire des dénis de service facilement car il n'y a aucun problème de débit similaire à ceux rencontrés sur Internet.

Pour éviter que les attaquants abusent de cette exigence, nous avons fait attention de n'utiliser que des moyens de communication sécurisés:
SSL pour le web, PGP pour les mails et SSH pour l'accès à distance

Au niveau de l'application, deux services sont indispensables pour assurer son fonctionnement:

- **le service Single Sign-On** délivré par des sites comme Google ou Yahoo sert à fournir un système d'authentification externe basé sur OpenID ou Oauth.
    - pour des raisons de simplicité, nous avons choisi de nous focaliser uniquement sur OpenID qui est plus facile à intégrer sur un site.
    - l'application et le wiki supportent le protocole OpenID via des extensions. Elles redirigent l'utilisateur vers le site tiers (Yahoo ou Google) et lui demande ses informations de connexion. Le retour est ensuite envoyé à notre site par HTTP.
    - note: l'API de Mediawiki fournit également ce service, mais sans être interopérable.
- **Le service API de Mediawiki**: fournit par le Wiki que nous avons installé sur le site permet de gérer les articles qu'il héberge depuis un site distant.
    - Les requêtes sont envoyées HTTP, et les paramètres sont contenus dans l'URL.
        - Ex : http://wikette/api.php?format=xml&action=logout
    - l'API est très peu sécurisée par défaut, il est nécessaire d'utiliser SSL pour garantir l'intégrité des requêtes. Cependant, nous avions pour consigne de ne pas le faire
    - nous avons codé une bibliothèque Python, séparée du code du site, et couverte par des tests unitaires. Cela nous permet de la réutiliser pour nos scripts d'attaque.

2. Détails techniques

2.1 Environnement

2.1.1 Système

Nom	Catégorie	Description	Arguments
Ubuntu Server	Système d'exploitation	Fondation logicielle de notre infrastructure	Robuste, minimaliste simple à administrer
Apache2	Serveur Web	Propulse l'application WikiWikiWeb et le Wiki Wikette	Populaire et bien supporté par nos applications (Mediawiki et Django)
MySQL	SGBD	Stocke les informations des applications web	Simple et performant pour des petites bases
OpenSSH	Accès à Distance	Permet d'administrer le site à distance	Sécurisé et fonctionne uniquement en CLI
splunk	Outils de supervision	Donne des statistiques sur l'utilisation du système et la disponibilité des sites	Gratuit (mais limité) très simple à installer repose sur des technologies open-source
ntp	Synchronisation du temps	Permet de retracer plus facilement les évènements passés	Choix par défaut sous Linux
Iptables	Filtrage réseau	Protège les services contre les attaques réseaux	Choix par défaut sous Linux
Fail2ban	Blocage DNS	Bloque les machines qui émettent trop de requêtes	Choix par défaut sous Linux
RSyslog	Journalisation	Garde une trace des opérations effectuées	Choix par défaut sous Linux
Logrotate	Rotation des journaux	Compresse et classe les journaux de log	Choix par défaut sous Linux
Duplicity	Sauvegarde logique	Sauvegarde les fichiers en cas de panne	Supporte l'envoi à distance et le chiffrement
Clonezilla	Sauvegarde physique	Sauvegarde les disques en cas de panne	Choix par défaut sous Linux

2.1.2 Développement

Nom	Catégorie	Description	Arguments
Django	Cadriciel de développement	Fournit de nombreux outils pour faciliter le développement	Très riche, bien conçu et beaucoup utilisé
Mediawiki	Logiciel de Wiki	Imposé par le CDC	/
Python	Langage de programmation	Bien construit et avec de nombreuses lib.	Permet un développement rapide et fiable
PHP	Langage de programmation	Imposé par le CDC (dépendance Mediawiki)	/
Mercurial	SCV	Décentralisé et multi-plateforme	Simple à utiliser pour les novices
Redmine	Forge logiciel	Facilite le travail collaboratif	Déjà installée et utilisée
KDevelop	IDE	Effectue des vérifications de code	Préférence personnelle
Fabric	Outils de déploiement	Évite les mauvaises manipulations	S'intègre très bien avec SSH
SQLite	Base de données	Permet d'utiliser une base de données locale	Fonctionne avec un simple fichier

Par rapport à à l'environnement que nous avions prévu dans le cahier des charges :

Nous sommes restés cohérents en essayant de trouver des technologies **matures**, **fiables** et **qui facilitent grandement la vie des développeurs/administrateurs**. Les opérations manuelles sont toujours sources d'erreur. Aussi, nous avons essayé de limiter leur nécessité **en rendant le plus d'opération possible automatique** (déploiement, mise à jour de la base, tests, partage des sources et de la documentation, sauvegarde, parcours des journaux …).

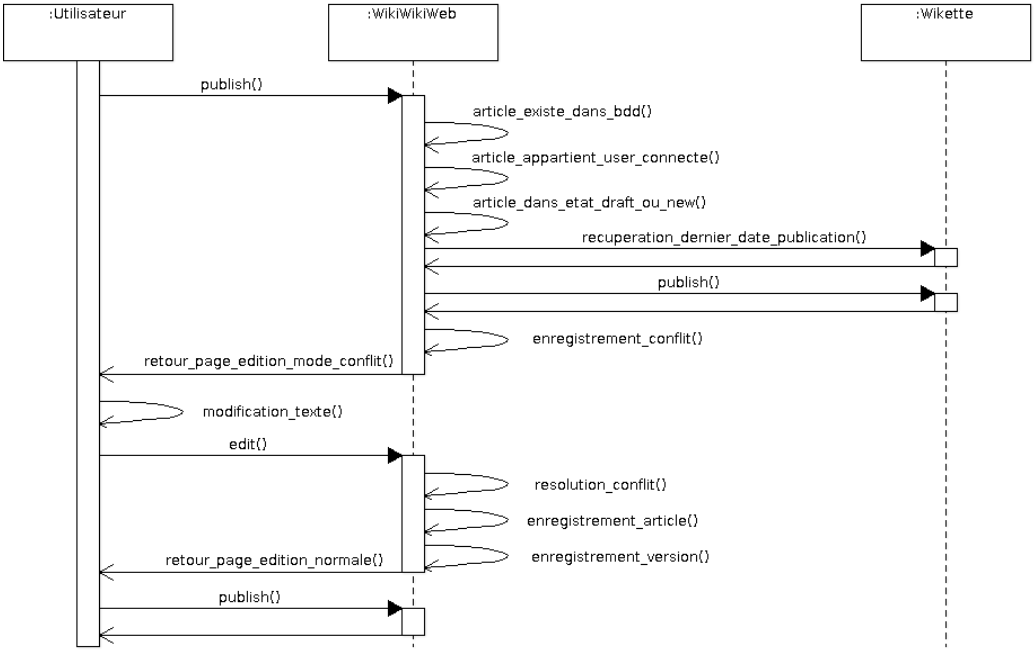
La version finale de l'environnement est **plus élaborée que prévu**, car nous avons pris le temps d'intégrer des outils qui facilitent le travail de maintenance. Nous voulions répondre plus facilement aux changements en prévision des attaques que nous allions subir.

Le changement majeur est l'apparition de **l'outil de supervision Splunk.** Il nous a permis de prendre connaissance plus rapidement de l'état du site et du système. Nous avions à l'origine sous-estimé l'importance de ce type d'outil, mais la pratique nous a montré qu'analyser des logs à la main est long, fastidieux et inefficace.

2.2 Application

2.2.1 Gestion des conflits

Le point le plus technique du site est la gestion des conflits entre les articles du site et du Wiki. Le diagramme suivant montre la gestion de l'application en cas de conflit.



Au départ, nous comptions utiliser des **robots** pour récupérer automatiquement les dernières versions des articles. Nous voulions ainsi **synchroniser** les articles locaux avec ceux du site et éviter à l'utilisateur de faire des manipulations. Cependant, cette méthode s'est révélée **compliquée à réaliser**, surtout en cas de conflit dû à un rafraichissement automatique.

Pour simplifier le traitement et l'utilisation, nous avons décidé d'adopter **une gestion paresseuse** (Lazy) dans la détection des conflits. Lorsque l'utilisateur veut publier un article, des vérifications sont effectuées sur le site avant de procéder à l'envoi. Nous nous basons sur la date de création ou de dernière récupération selon que l'article a déjà été récupéré/publié ou non. Les conflits sont détectés par le Wiki qui compare les dates des versions locales et distantes. Dans ce cas, l'article est marqué comme en conflit et l'utilisateur est invité à l'éditer.

Une fois le texte modifié, il est envoyé, vérifié et sauvegardé comme résolu. L'utilisateur peut tenter de le renvoyer, et l'étape se répète tant que des conflits sont détectés. Si tout se passe bien, l'état de l'article est modifié sur notre application et la nouvelle version apparait sur le Wiki.

Cette version nous parait la plus efficace en temps et en ressource.

2.2.3 Organisation du code

Pour que vous puissiez apprécier l'implémentation de l'application, nous vous donnons dans le tableau suivant avec la structure physiques des répertoires et des fichiers.

● api/mediawiki/ ○ client.ini ○ lib.py ○ requests.py ○ errors.py ○ tests.py	**code source de l'API Mediawiki** fichier de configuration (informations d'authentification) bibliothèque bas-niveau (utilisée pour construire des apps.) bibliothèque haut-niveau (utilisé dans notre app. Web) exceptions lancées par l'API tests unitaires basée sur unittest
● public/ ○ static/ ○ wsgi.py	**point de montage du serveur web** (data non-confidentielle) contient les fichiers publiques (JS, feuille de style, images …) point d'entrée de l'application (lien entre le serveur et l'app.)
● settings/ ○ common.py ○ prod.py ○ dev.py	**dossiers contenant les fichiers de configurations** configurations communes à tous les environnement configurations dédiées à l'environnement de production configurations dédiées à l'environnement de test
● weditor/ ○ fixtures/ ○ static/ ○ templates/ ○ templatetags/ ○ views.py ○ addons.py ○ admin.py ○ forms.py ○ models.py	**code source de l'application principale** (weditor) données initiales à insérer dans la base (format YAML) fichiers publiques de l'application (JS, CSS, images …) code HTML du site = logique de présentation (couche vue) fonctions permettant d'étendre le langage des templates fonctions de traitements = logique métier (couche contrôleur) fonctions communes aux traitements déclaration de l'interface d'administration du site déclaration des formulaires interface avec la base de données (couche modèle)
● url.py	dispatcher : association entre les requêtes et les fonction
● manage.py	script permettant de manipuler le code source

Plusieurs points intéressants concernant cette organisation :

- **le point de montage du serveur est séparé du code principal** : cela permet de cloisonner ce que le serveur web peut exposer et ce qui doit rester confidentiel.
- **l'api est modulaire (bibliothèque bas niveau/haut niveau)** : de cette façon, il est possible de la réutiliser pour d'autres applications (ex : scripts d'attaque).
- **plusieurs environnements pour chaque instance applicative** : nous pouvons ainsi faire des vérifications dans un environnement confiné avant de mettre en production.
- **La structure de l'application respecte le modèle MVC** : (voir partie 1.2.1)

2.3 Mécanismes de protection

Dans cette partie, nous allons résumer les actions de sécurisation effectuées
Pour plus de détails, vous pouvez consulter directement les fichiers de configuration.

2.3.1 Défense système et périphérique

- **Retrait de la bannière de connexion par défaut:** un message de bienvenue est souvent affiché aux utilisateurs souhaitant se connecter à un service réseau (SSH, MySQL, SMTP …). Ce message peut révéler des informations secrètes que les détecteurs de vulnérabilités peuvent repérer.
    - pour résoudre se problème, nous avons supprimé le contenu du fichier /etc/issue

- **Changement dans /etc/fstab**: pour que les attaquants ne puissent pas exécuter leurs propres scripts, nous avons ajouté la clause no-exec à certains volumes montés.
    - cette option ignore le droit d'exécution (+x) sur les fichiers
    - cette option concerne /home, /mnt/backup, /mnt/upload, /tmp et /var/log.

- **Configuration avancée du système**: pour renforcer la sécurité du système, nous avons ajouté des règles supplémentaires dans le fichier /etc/sysclt.conf
    - ces commandes sont par exemple : désactiver la réponse au ping, la fonction de routage, IPV6, surveiller les paquets mal formés, supprimer les clés magiques …

- **Configuration du pare-feux**: pour protéger les services, nous avons configuré iptables pour restreindre les ports accessibles depuis le réseau et filtrer les paquets mal formé.
    - les paquets broadcast et TCP avec des drapeaux incorrects sont ignorés
    - les connexions établies persistent lors du rechargement des règles iptables pour que nous ne perdions pas la main à distance en cas de mauvaise manipulation
    - par défaut, les paquets entrantes sont bloquées, les paquets sortants sont quand à eux autorisés à sortir pour simplifier la maintenance (bénéfice < coût)
    - les requêtes sont par défaut surveillées par le démon rsyslog
    - les règles sont rechargées à chaque redémarrage par le service iptable-save

- **Changement des règles de création des utilisateurs**: nous avons édité le fichier /etc/login.defs pour renforcer la sécurité des comptes utilisateurs
    - les paramètres de sécurité modifiés sont entre autres le délai entre 2 tentatives, la valeur par défaut du masque (umask), la taille minimum du mot de passe …
    - notez que la plupart de ces paramètres sont maintenant gérés par le module PAM. Lorsque c'était nécessaire, nous les avons édité à ce niveau.

- **Changement du masque par défaut**: pour être sûr que les masques soient bien à jours pour les utilisateurs déjà crées, nous avons éditer le fichier ~/.profile
    - nous avons été le plus restrictif possible avec le masque 077. Seul l'utilisateur peut consulter ses fichiers.

- **Configuration de l'accès à distance :** nous avons installé et configuré le service SSH pour pouvoir administrer le serveur à distance
    - les configurations ont été éditées dans le fichier /etc/ssh/sshd_config
    - le plus grand changement est le passage vers une authentification par clé asymétrique plutôt que par phrase de passe. Nousvoulions ainsi décourager les attaquants de faire du brute-force (l'erreur indique clairement que c'est inutile).
    - nous avons également changé le port par défaut (de 22 à 4812) pour brouiller les détections et laisser le temps au logicel de sécurité de les identifier.

- **Configuration de l'attribution des hôtes** : pour limiter les tentatives d'usurpation des IP, nous avons édité le fichier /etc/host.conf qui gère l'attribution des hôtes.

- **Configuration de la base de données** : les bases MySQL ne sont pas sécurisées par défaut. Il se peut que des comptes et des informations secrètes soient accessibles.
    - pour sécuriser l'instance, nous avons utilisé le script mysql_secure_installation
    - nous avons ensuite fait des opérations manuelles : création des comptes applicatifs (1 pour le Wiki et 1 pour le site), suppression des accès réseau, changement du port par défaut (vers 20012).
        - Ces opérations ont été faites directement depuis le client MySQL ou bien par le fichier de configuration /etc/mysql/my.conf
    - pour être sûr que rien a été oublié, nous avons procédé à un audit sur la base.
    - par la suite, nous avons utilisé le script mysqltuner.pl pour améliorer les performances de la base après avoir récolté assez de données en entrée.

- **Configuration du serveur web** : le serveur web est au cœur de notre dispositif, car il traite les requêtes extérieures tout en ayant accès aux composants internes du système (le système de fichier, le code source …). C'est donc un élément critique !
    - nous avons procédé au déploiement avant la sécurisation pour être sûr que les éléments de sécurité ne rentrent pas en conflit avec nos applications
    - nous avons ensuite modifié les paramètres de sécurité par défaut contenu dans le fichier /etc/apache2/security.conf pour garder lesinformations secrètes.
    - nous avons installé mod_security qui est un IPS (donc actif) pour Apache
        - ce module nous a posé beaucoup de problème car il a bloqué de nombreuses requêtes légitimes. Nous l'avons souvent désactivé.
    - nous avons ajouté le module mod_evasive pour bloquer les tentatives de DoS
        - ce module est difficile a bien paramétrer car il faut être capable de déterminer les seuils liés aux traffics légitimes. Il ne nous a pas empêché de bloquer la première attaque DoS que nous avons subit.

    - pour brouiller la détection de notre serveur web, nous avons changé l'entête «Serve » de la réponse HTTP de Apache vers Microosft-IIS/8.5.

- **Configuration de la journalisation** : pour suivre les évènements effectués sur le système, tant ceux des utilisateurs que des administrateurs, il est nécessaire de garder des journaux de log avec suffisamment d'information pour comprendre ce qui se passe.
    - Pour gérer cette fonction, nous avons utilisé rsyslog qui est fourni par défaut sur le système d'exploitation que nous utilisons (Ubuntu Server)
    - nous avons changé le niveau de verbosité et le fichier de destination quand cela était nécessaire pour l'adapter à nos besoins. Ceci était particulièrement important pour le service web Apache et le SGBD MySQL
    - Pour optimiser les performances, nous avons rendu la plupart des écritures asynchrones, sauf pour les authentifications et les erreurs.

- **Configuration de la rotation des journaux** : en complément de rsyslog, logrotate permet de changer de fichier au bout d'un certain temps pour les traiter plus facilement et plus rapidement.
    - au départ, nous l'avons principalement utilisé pour archiver les anciens logs
    - par la suite, il s'est révélé très utile pour classer les logs par jours

- **Configuration de la sauvegarde logique** : grâce au logiciel duplicity, nous avons pu effectuer des sauvegardes logiques et les exporter vers un site Internet distant.
    - la sauvegarde logique fonctionne sur un système en cours de fonctionnement et permet de conserver les fichiers tel que le système les voit
    - ce logiciel supporte le chiffrement asymétrique (avec pgp) et symétrique (avec une phrase de passe). Nous avons préféré la seconde méthode, car elle est suffisante pour notre usage et facile à utiliser.
    - l'export se fait par un compte FTP isolé sur un système distant (hébergeur)
    - la sauvegarde est planifiée toutes les semaines par un démon cron
    - la sauvegarde comprend les fichiers de configuration, des dossiers personnels, l'export de la base de données et des paquets installés.

- **Configuration de la sauvegarde physique** : nous avons sauvegardé le disque entier du système pour pouvoir le restaurer complètement en cas d'anomalie.
-
    - la sauvegarde physique avec Clonezilla nécessite que le système soit éteint, ce qui pose un inconvénient pour un système non redondant comme le nôtre
    - en comparaison, une solution de virtualisation peut effectuer ses sauvegardes sans interrompre le système. Celui ci sera également plus facile à redémarrer.
    - Nous avons procédé à une sauvegarde complète avant la mise en production.

2.3.2 Défense applicative

La défense applicative est par nature très différente de celle du système. Elle est construite **tout au long du processus de développement**. Il faut s'assurer que chaque fonction, chaque accès et chaque affichage soit contrôlé pour ne pas qu'il soit abusé. Heureusement, notre outil de développement Django et le logiciel Mediawiki nous ont permis de nous appuyer sur des méthodes approuvées pour sécuriser tous ces éléments.

Pour le déploiement, nous avons crée des hôtes virtuels afin d'isoler le contexte d'exécution de chaque application. Au total, nous comptons 4 environnements :

- port 80 : l'instance non sécurisée de Mediawiki (Wikette)
- port 443 : l'instance sécurisée de Mediawiki (Wikette-SSL)
- port 9667 : l'instance non sécurisée de l'application (WikiWikiWeb)
- port 10000 : l'instance sécurisée de l'application (WikiWikiWeb-SSL)

Pour respecter les consignes, nous avons veillé grâce à des règles de redirection que seules les pages d'authentification et d'inscription soient sécurisées **par une connexion SSL**.

WikiWikiWeb est déployé automatiquement par un script utilisant le logiciel Fabric (fabfile.py).

Mediawiki a été installé depuis les dépôts par défaut de Ubuntu (en version 1.19).

Ces deux sites ont été protégés en suivant les **tutoriels** respectifs présents sur les sites web :
https://docs.djangoproject.com/en/dev/topics/security/
http://www.mediawiki.org/wiki/Manual:Security

Grâce aux fonctions de Django et à la qualité de Mediawiki, nous avons à notre disposition de nombreuses défenses contre les attaques populaires : injection SQL, CSRF, XSS ...

Comparé à notre planning initial, nous avons réussi à réaliser toutes les fonctions, aussi bien en terme utilisabilité que de sécurité. Nous avons cependant du améliorer la sécurité à plusieurs reprises en fonction des attaques des autres groupes et des nôtres quand c'était possible :

- **correction du bug sur la fonction « voir » de l'historique des articles** : le texte de l'article ne s'affichait pas dans la partie prévue à cette effet, mais il était toujours possible de relancer l'historique.
- **correction des bugs liés à mod_evasive** : le module Apache détectait un grand nombre de faux négatifs, ce qui engendrait des problèmes d'utilisation. Nous avons dû adapter finement la configuration, en passant beaucoup de temps dessus.
- **prévention des attaques automatiques liées à l'API du site** : grâce à des configurations dans le fichier LocalSettings.php, nous avons pu prévenir certaines attaques comme la suppression en masse des articles ou la création en masse de compte.

3. Présentation des attaques

3.1 Aspiration de site web grâce à un index Git

Nom du rapport	Rapport d'A001
Groupe ciblé	Groupe 2
Nom Attaque	Aspiration site web
Category d'attaque	Vol d'information
Techniques	Script Python alimenté par un fichier index Git.
CID	Atteinte à la confidentialité du code source et de la base de données.
Preuve de l'attaque	Vous pourrez trouver en pièce-jointe le code source de leur applicatio et le script SQL.

3.2 Suppression automatique des articles d'un Wiki

Nom du rapport	Rapport d'A002 & A003
Groupe ciblé	Groupe 2 & 3
Nom Attaque	Flush de tous les articles du wiki cible
Category d'attaque	Attaque par bot python
Techniques	Bot python threadé utilisant l'API écrite
CID	La disponibilité des article est touchée (les articles sont remplacés) L'intégrité est touchée car l'historique est rempli
Preuve de l'attaque	Défiguration totale des articles du wiki, tous remplacés par une "pierre tombale" ou "tête de mort"

3.3 Remplissage automatique d’un Wiki

Nom du rapport	Rapport d’A004
Groupe ciblé	Groupe 2
Nom Attaque	Flood du wiki
Category d’attaque	Attaque par bot python
Techniques	Bot python threadé utilisant l'API écrite
CID	Le bot va toucher la disponibilité du wiki voir du serveur en remplissar la base de donnée d’articles. On ne pourra plus créer d’articles et le serveur risque de planter.
Preuve de l’attaque	Déni de service

Nom du rapport	Rapport d’A005
Groupe ciblé	Groupe 3
Nom Attaque	DOS du wiki
Category d’attaque	Attaque par bot python
Techniques	Bot python threadé utilisant l'API écrite
CID	Le bot va toucher la disponibilité du wiki voir du serveur en remplissar la base de donnée d’articles. On ne pourra plus créer d’articles et le serveur risque de planter.
Preuve de l’attaque	Déni de service

3.4 Déni de service par requêtes escargots

Nom du rapport	Rapport d'A006 & A007
Groupe ciblé	Groupe 2 & 3
Nom Attaque	slowloris
Category d'attaque	Déni de Service
Techniques	En envoyant des requêtes octet par octet, on peut simuler un client web très lent (c'est le cas par exemple pour un téléphone mobile su bande GSM). Ces requêtes sont très lourdes à gérer pour le serveur, car les délais de transferts occupent les processus dédiés à traiter les autres demandes. Si on envoie suffisamment ce type de requête, or peut saturer le serveur en quelques secondes.
CID	Site indisponible à 95%
Preuve de l'attaque	Les captures suivantes montrent le script d'attaque en cours de fonctionnement et le résultat sur le navigateur. Les requêtes restent en cours de chargement, sans jamais aboutir.

3.5 Usurpation d'adresse email

Nom du rapport	Rapport d'A008 & A009
Groupe ciblé	Groupe 2 & 3
Nom Attaque	Le mail spoofing de la mort qui tue :p
Category d'attaque	Spoofing
Techniques	Connexion au server SMTP de l'université via Telnet
CID	100% : la confidentialité
Preuve de l'attaque	Le mail a bien été reçu par l'ensemble de la promo et tous on dut faire un rapport sur leur faille qu'ils doivent envoyer vendredi avant 12h.

3.6 Injection SQL sur un champ de recherche

Nom du rapport	Rapport d'A010
Groupe ciblé	Groupe 3
Nom Attaque	Le mail spoofing de la mort qui tue :p
Category d'attaque	Injection SQL
Techniques	Première étape : recherche de champs vulnérables à l'injection sql (on met ' dans un champ et on regarde) Deuxième étape : une fois un champ trouvé on essaye différentes injections (ici la mienne est très basique : '#).
CID	La confidentialité est touchée.
Preuve de l'attaque	Affichage "Erreur requête sql", preuve qu'une injection est faisable

3.7 Failles des autres groupes

Nom du rapport	Rapport d'A011
Groupe ciblé	Groupe 3
Nom Attaque	Exploitation de la faille
Catégorie d'attaque	Remplissage de la base de données
Techniques	Première étape : recherche de champs vulnérables à l'injection sql (on met ' dans un champ et on regarde) Deuxième étape : une fois un champ trouvé on essaye différentes injections (ici la mienne est très basique : '#).
CID	Integrité
Preuve de l'attaque	Aucune preuve n'est disponible car leur serveur bug énormément et malgré les tentatives de repartions de Gorlt et Charlie rien n'as pu être fait. Gorlt me la fait testé sur le serveur de teste.

Nom du rapport	Rapport d'A012
Groupe ciblé	Groupe 2
Nom Attaque	Exploitation de la faille
Catégorie d'attaque	Mot de passe
Techniques	Création d'un script en PHP permettant de testée des mots
CID	Integrité
Preuve de l'attaque	Récupération du mot de passe

4. Présentation des défenses

4.1 Déni de service par grand nombre de requête

Nom du rapport	Rapport de D001
Groupe Attaquant	Groupe 2
Categorie d'attaque	Attaque de type Déni de Service
Techniques	L'attaquant a envoyé un grand nombre de requête HTTP sur une courte période (3341 requêtes en 2 minutes). Les requêtes n'ont rien de particulier, c'est leur quantité qui a posé problème.
CID	L'application WikiWikiWeb était rendue complétement indisponible (disponibilité).
Description des dégâts	L'attaque a porté atteinte à la disponibilité de notre application. Celle-n'était plus capable de répondre aux requêtes légitimes des autres utilisateurs (temps d'expiration dépassé).

4.2 Tentative de découverte de vulnérabilités par requêtes web forgées

Nom du rapport	Rapport de D002
Groupe Attaquant	Groupe 2
Catégorie d'attaque	Attaque automatisée (logiciel) sur notre application Web (WikiWikiWeb)
Techniques	L'attaquant a émi un grand nombre de requête typique d'un logiciel d'attaque.
CID	L'application est restée accessible et aucune information n'a fuité. Nous avons pu vérifier cette dernière affirmation en filtrant les requêtes réussies (HTTP Code 200) et en analysant leur contenu. Elles ne contenaient rien de particulier.
Description des dégâts	Les journaux montrent des URL avec des caractères encodées, du code SQL et des injections de chemins d'accès. Plus de 41 000 requêtes ont été émises en 1H.

4.3 Tentative d’attaque par remplissage automatique du Wiki

Nom du rapport	Rapport de D004
Groupe Attaquant	Groupe 3
Catégorie d’attaque	Flood qui mène à du dos
Techniques	Script php qui créé des articles au hasard avec des noms randomisé
CID	La disponibilité est touchée une fois que la base de données est remplie
Description des dégâts	Aucun dégât subit, juste des articles inutiles sur le wiki

Nom du rapport	Rapport de D005
Groupe Attaquant	Aucune idée
Catégorie d’attaque	Dos par changement de mot de passe
Techniques	Aucune idée, il peut y avoir plusieurs manières : -bruteforce (impossible, il y a une sécurité mais on ne sait jamais) -social engeenering -Tout simplement notre faille qui a été trouvée
CID	La disponibilité est touchée une fois que la base de données est remplie
Description des dégâts	Les 3 critères ont été touchés.

5. Conclusion

5.1 Bilan du projet

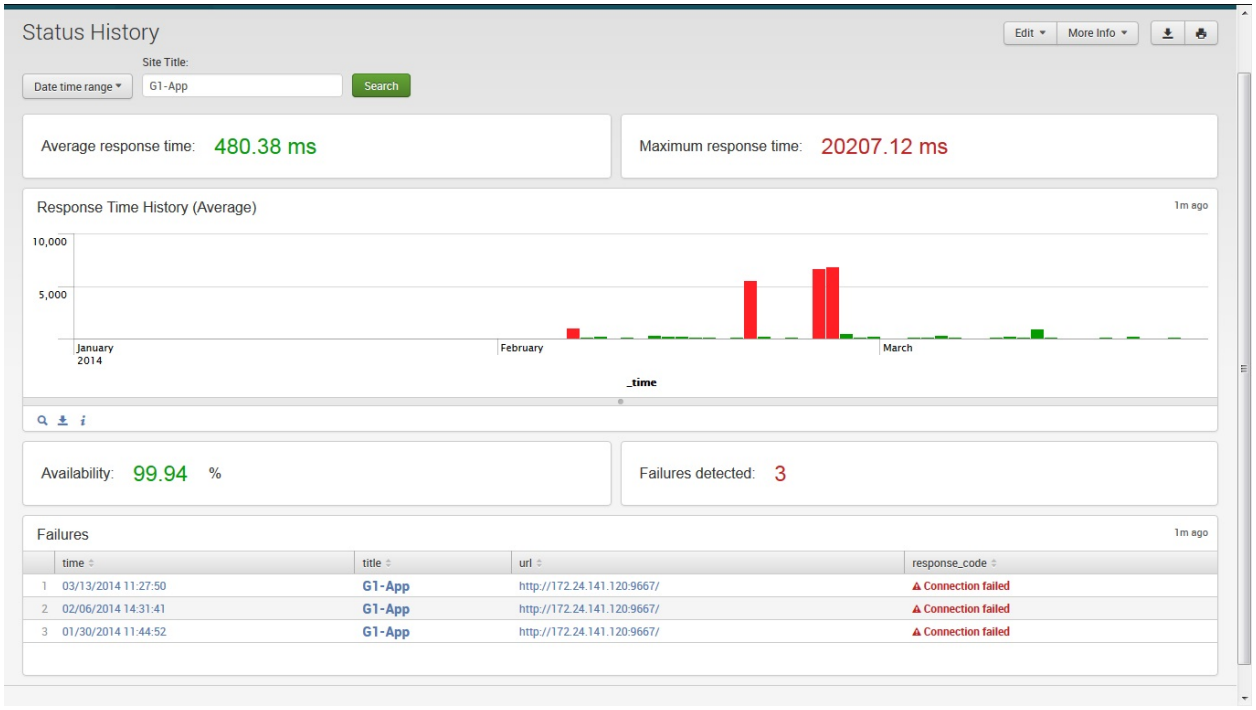
Nous pouvons difficilement avancer un bilan sachant que les attaques des autres groupes sont restées secrètes pendant toute la seconde phase du projet. Cependant, nous pensons que notre installation a bien tenu malgré les perturbations que nous avons subi.

Ces résultats sont à prendre avec beaucoup de recul ! En effet, nous les avons pris sur le système qui sert à héberger les sites. Nous espérons toutefois que vous pourrez les comparer avec ceux des autres groupes.

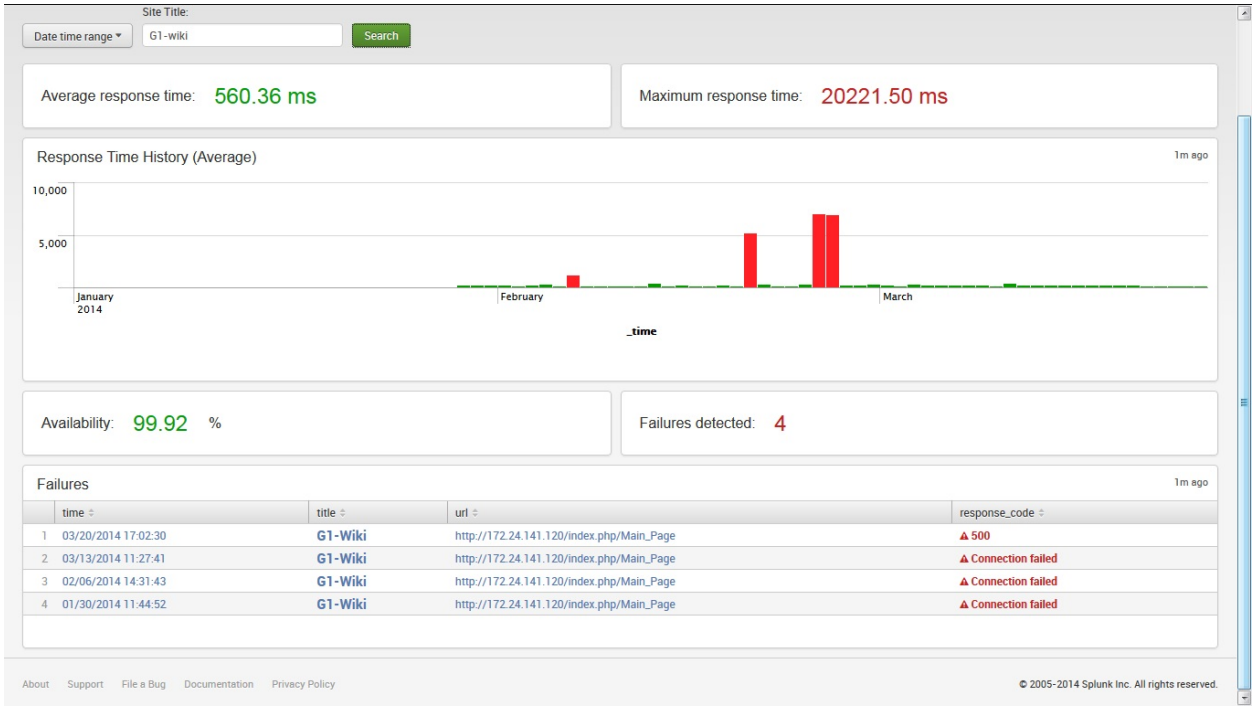
Nous sommes également satisfait du rapport utilisabilité/sécurité que propose notre système. En effet, un système doit rester utilisable pour remplir sa fonction et pour que les utilisateurs ne cherchent pas à contourner les moyens de sécurisation mis en place. Grâce à des tests poussés et l'automatisation des tâches, nous avons pu consacrer du temps à la revue globale.

Enfin concernant nos attaques, nous pensons qu'elles n'ont pas été aussi dévastatrices que prévues. Nous avons réussi à récupérer de nombreuses informations secrètes, tel que le code source et la structure de la base de données du groupe 2 ou à faire tomber les systèmes en saturant la base du Wiki par des robots automatiques. Nous n'avons cependant pas réussir à obtenir un accès root, les mots de passe étant difficile à craquer vu les moyens disponibles.

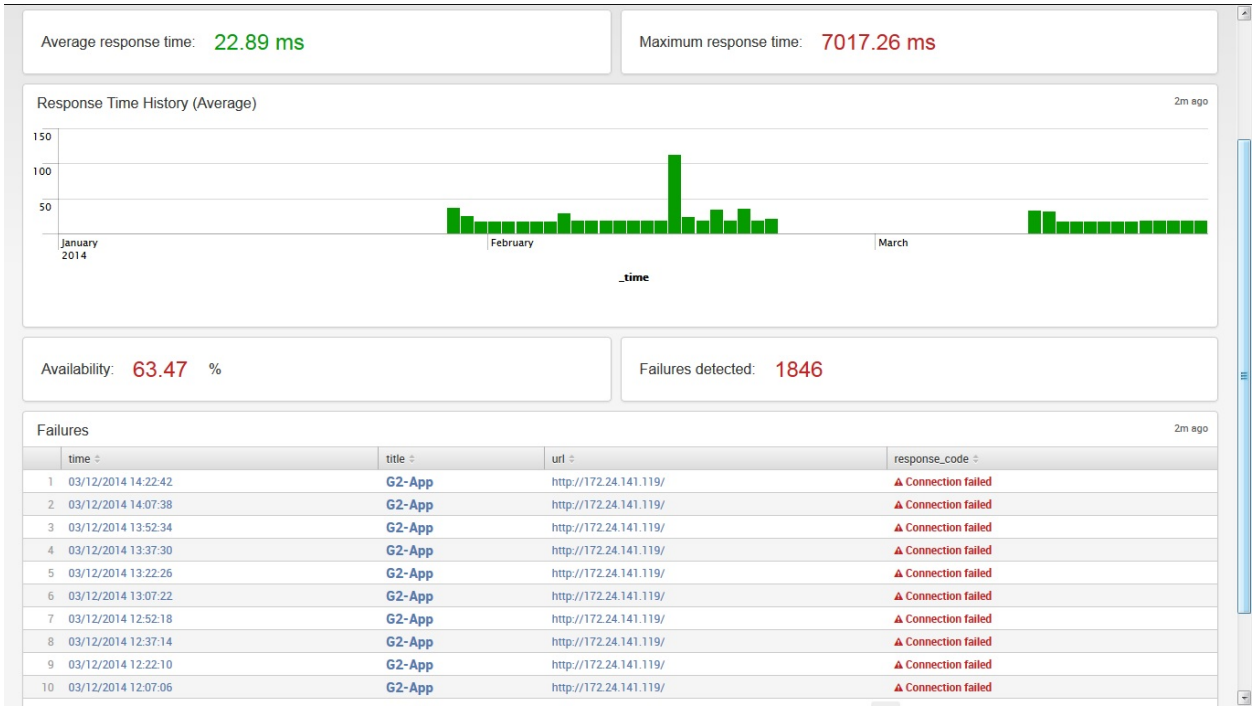
Disponibilité de l'application du groupe 1



Disponibilité de Wiki du groupe 1



Disponibilité de l'application du groupe 2



Average response time: 22.89 ms

Maximum response time: 7017.26 ms

Response Time History (Average) — 2m ago

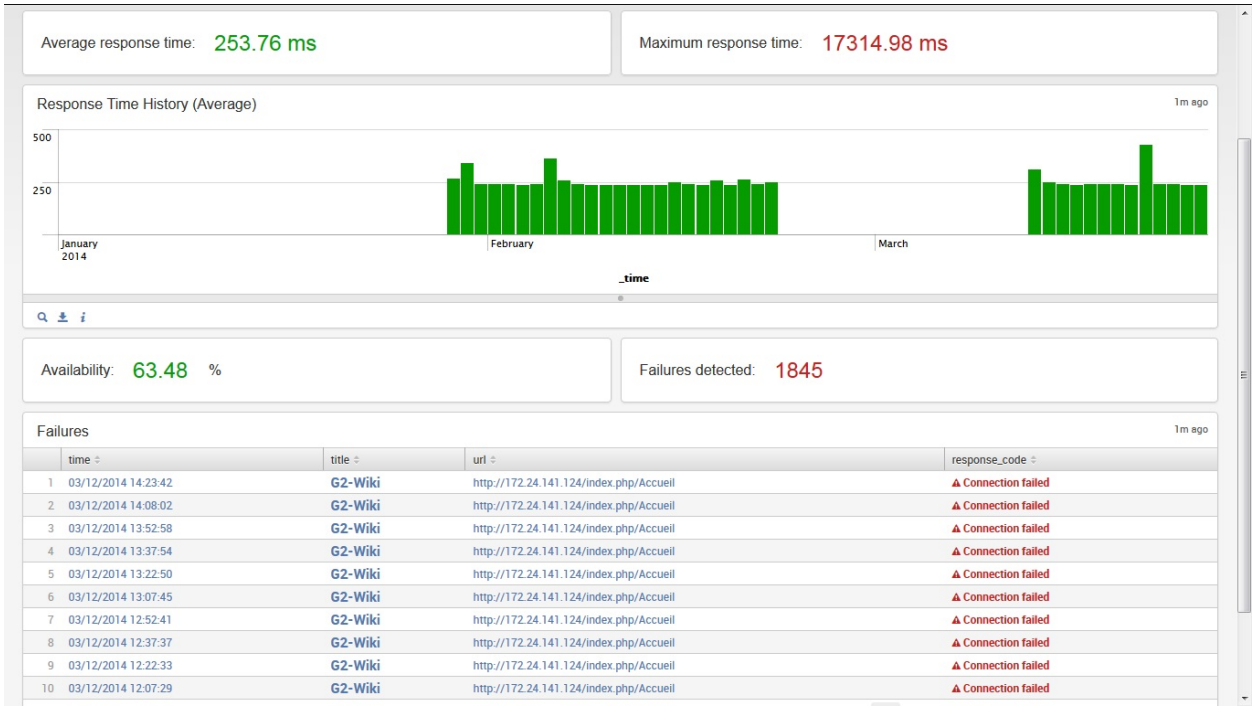
Availability: 63.47 %

Failures detected: 1846

Failures — 2m ago

	time	title	url	response_code
1	03/12/2014 14:22:42	G2-App	http://172.24.141.119/	Connection failed
2	03/12/2014 14:07:38	G2-App	http://172.24.141.119/	Connection failed
3	03/12/2014 13:52:34	G2-App	http://172.24.141.119/	Connection failed
4	03/12/2014 13:37:30	G2-App	http://172.24.141.119/	Connection failed
5	03/12/2014 13:22:26	G2-App	http://172.24.141.119/	Connection failed
6	03/12/2014 13:07:22	G2-App	http://172.24.141.119/	Connection failed
7	03/12/2014 12:52:18	G2-App	http://172.24.141.119/	Connection failed
8	03/12/2014 12:37:14	G2-App	http://172.24.141.119/	Connection failed
9	03/12/2014 12:22:10	G2-App	http://172.24.141.119/	Connection failed
10	03/12/2014 12:07:06	G2-App	http://172.24.141.119/	Connection failed

Disponibilité de l'application du groupe 2



Average response time: 253.76 ms

Maximum response time: 17314.98 ms

Response Time History (Average) — 1m ago

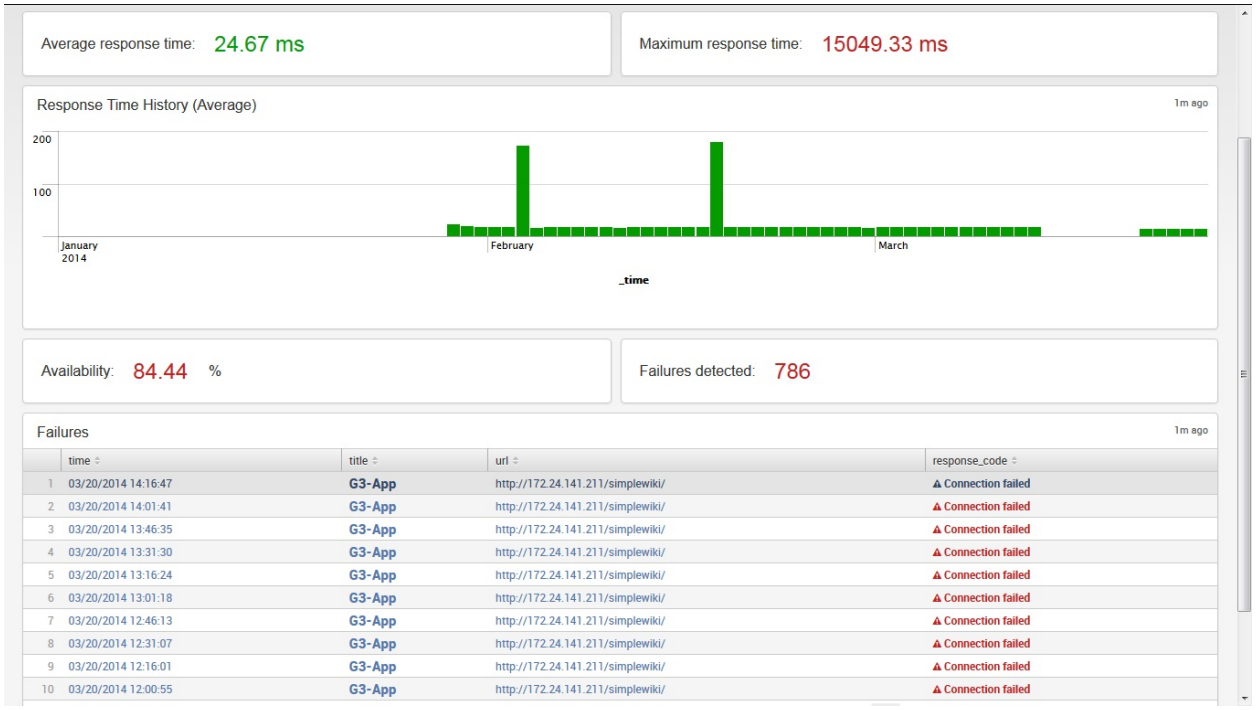
Availability: 63.48 %

Failures detected: 1845

Failures — 1m ago

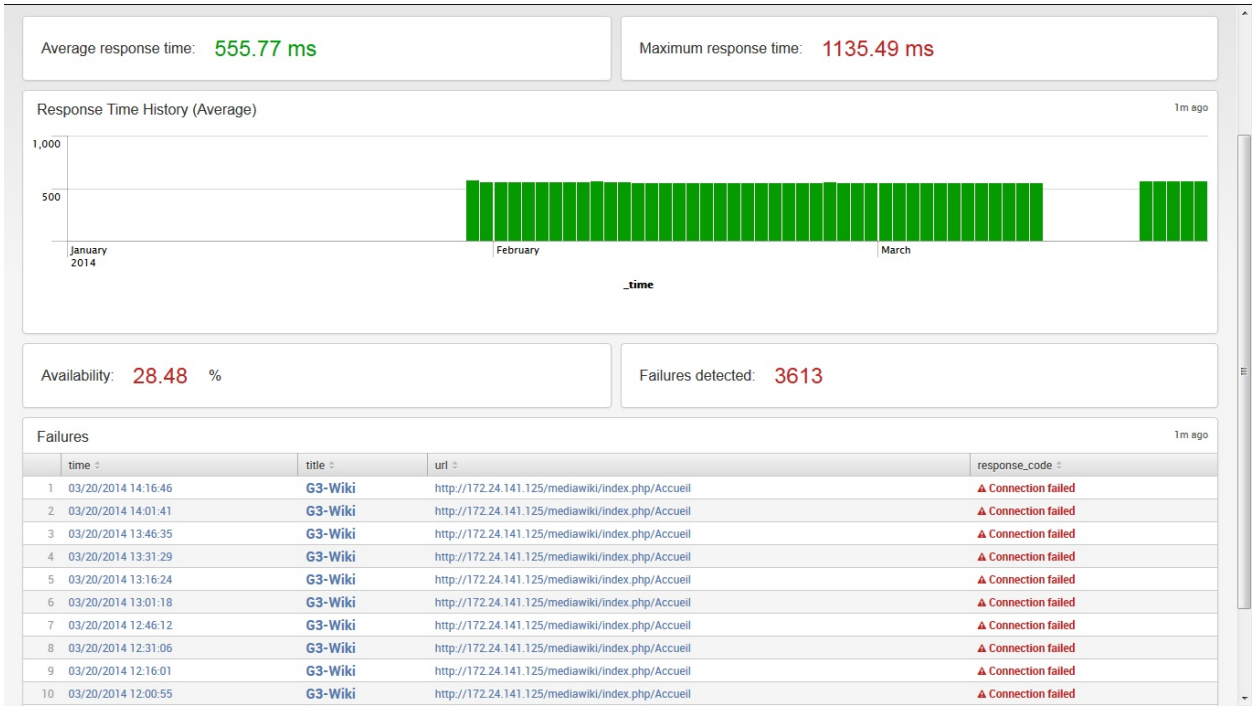
	time	title	url	response_code
1	03/12/2014 14:23:42	G2-Wiki	http://172.24.141.124/index.php/Accueil	Connection failed
2	03/12/2014 14:08:02	G2-Wiki	http://172.24.141.124/index.php/Accueil	Connection failed
3	03/12/2014 13:52:58	G2-Wiki	http://172.24.141.124/index.php/Accueil	Connection failed
4	03/12/2014 13:37:54	G2-Wiki	http://172.24.141.124/index.php/Accueil	Connection failed
5	03/12/2014 13:22:50	G2-Wiki	http://172.24.141.124/index.php/Accueil	Connection failed
6	03/12/2014 13:07:45	G2-Wiki	http://172.24.141.124/index.php/Accueil	Connection failed
7	03/12/2014 12:52:41	G2-Wiki	http://172.24.141.124/index.php/Accueil	Connection failed
8	03/12/2014 12:37:37	G2-Wiki	http://172.24.141.124/index.php/Accueil	Connection failed
9	03/12/2014 12:22:33	G2-Wiki	http://172.24.141.124/index.php/Accueil	Connection failed
10	03/12/2014 12:07:29	G2-Wiki	http://172.24.141.124/index.php/Accueil	Connection failed

Disponibilité de l'application du groupe 3



	time	title	url	response_code
1	03/20/2014 14:16:47	G3-App	http://172.24.141.211/simplewiki/	Connection failed
2	03/20/2014 14:01:41	G3-App	http://172.24.141.211/simplewiki/	Connection failed
3	03/20/2014 13:46:35	G3-App	http://172.24.141.211/simplewiki/	Connection failed
4	03/20/2014 13:31:30	G3-App	http://172.24.141.211/simplewiki/	Connection failed
5	03/20/2014 13:16:24	G3-App	http://172.24.141.211/simplewiki/	Connection failed
6	03/20/2014 13:01:18	G3-App	http://172.24.141.211/simplewiki/	Connection failed
7	03/20/2014 12:46:13	G3-App	http://172.24.141.211/simplewiki/	Connection failed
8	03/20/2014 12:31:07	G3-App	http://172.24.141.211/simplewiki/	Connection failed
9	03/20/2014 12:16:01	G3-App	http://172.24.141.211/simplewiki/	Connection failed
10	03/20/2014 12:00:55	G3-App	http://172.24.141.211/simplewiki/	Connection failed

Disponibilité du Wiki groupe 3



	time	title	url	response_code
1	03/20/2014 14:16:46	G3-Wiki	http://172.24.141.125/mediawiki/index.php/Accueil	Connection failed
2	03/20/2014 14:01:41	G3-Wiki	http://172.24.141.125/mediawiki/index.php/Accueil	Connection failed
3	03/20/2014 13:46:35	G3-Wiki	http://172.24.141.125/mediawiki/index.php/Accueil	Connection failed
4	03/20/2014 13:31:29	G3-Wiki	http://172.24.141.125/mediawiki/index.php/Accueil	Connection failed
5	03/20/2014 13:16:24	G3-Wiki	http://172.24.141.125/mediawiki/index.php/Accueil	Connection failed
6	03/20/2014 13:01:18	G3-Wiki	http://172.24.141.125/mediawiki/index.php/Accueil	Connection failed
7	03/20/2014 12:46:12	G3-Wiki	http://172.24.141.125/mediawiki/index.php/Accueil	Connection failed
8	03/20/2014 12:31:06	G3-Wiki	http://172.24.141.125/mediawiki/index.php/Accueil	Connection failed
9	03/20/2014 12:16:01	G3-Wiki	http://172.24.141.125/mediawiki/index.php/Accueil	Connection failed
10	03/20/2014 12:00:55	G3-Wiki	http://172.24.141.125/mediawiki/index.php/Accueil	Connection failed

5.2 Rétrospective

Au niveau de phase de conception, nous regrettons de ne pas avoir utilisé la virtualisation comme l'ont fait les autres groupes. Cela ne nous a pas pénalisé au final, mais nous savions qu'il serait difficile de restaurer notre système en cas de problème majeur. Notre choix de prudence ne nous a pas permis d'apprendre les techniques qu'offrent ces outils, et nous sommes très curieux de voir comment cet aspect a été géré par les autres groupes.

Au niveau de la phase d'attaque, les attaques de type DoS ont été pour nous un cauchemar. Il est très difficile de s'en protéger, car les outils demandent une connaissance pointue de l'utilisation du site et des méthodes d'attaque. Nos lignes de défenses (mod_evasive, fail2ban, mod_security) se sont révélées inefficaces, mais nous savons qu'il s'agit là d'un manque de connaissance plus qu'un manque de fiabilité des outils. C'était la principale faille de notre système, et les autres groupes en ont justement abusé.

Enfin, nous avons grandement sous-estimé l'importance de la supervision dans le cadre d'un projet comme celui-ci. Rien n'indique qu'un site est attaqué si il n'y a pas d'indicateur, et le fait qu'il soit indisponible est une observation trop tardive. Nous aurions dû mettre en place un outil complet avant la phase d'attaque. Nous nous sommes cependant rattrapés par la suite en mettant en œuvre une plateforme efficace et complète basée sur Splunk.

5.3 Nos avis

La liberté qu'offre ce projet a été une expérience enrichissante qui nous a permis de laisser s'exprimer toute notre créativité. C'est une autonomie qu'il est nécessaire de connaitre avant de rejoindre le monde de l'entreprise, et ce projet nous a donné un cadre parfait pour cela.

Cette expérience n'a cependant pas été pleinement vécue, car nous manquions cruellement de temps pour réaliser tout ce que nous avions prévu (aussi bien pour la conception que l'attaque). Les nombreux projets et TD à faire en parallèle au cours du semestre étaient trop prenant pour que nous puissions faire les recherches et le travail nécessaire. De plus, nous n'avions pas eu de cours de pentest concret pour bien commencer cette phase (ex : utilisation de metasploit).

Au niveau du sujet lui même, nous regrettons que ce projet ne soit pas plus représentatif d'un cas réel. Idéalement, nous aurions du déployer ce système sur plusieurs machines (ou imposer la virtualisation) et avoir pleinement accès à Internet pour gérer les services externes (ex : mail).

De plus, le fait qu'aucun critère objectif ne soit fixé quant à l’utilisabilité du site incite les groupes à se concentrer sur la sécurité en délaissant complètement le coté fonctionnel. Cela réduit considérablement la surface d'attaque. Vu les exigences du Master 2, il pourrait s'avérer profitable de faire des projets communs entre IHM et SSI, chacun étant maître de sa partie.