RÉPUBLIQUE DU BÉNIN
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ D'ABOMEY-CALAVI

INSTITUT DE FORMATION ET DE RECHERCHE EN INFORMATIQUE

BP 526 Cotonou Tel : +229 21 14 19 88
http://www.ifri-uac.net Courriel : contact@ifri.uac.bj

MÉMOIRE

pour l'obtention du

Diplôme de Licence en Informatique

Option : Sécurité Informatique

Présenté par :
Jospy **GOUDALO**

Proposition d'un système de paiement électronique des factures d'électricité de la SBEE.

Sous la supervision :
Professeur Eugène C. EZIN
Maître de Conférences en Informatique
Université d'Abomey-Calavi

Année Académique : **2017-2018**

Table des matières

Table des figures

Liste des tableaux

Dédicace

A

mon Père **Noll GOUDALO**

ma Mère **Nadine SODEDJI**

et mon Frère **Brice GOUDALO**

Remerciements

Nous exprimons notre vive gratitude aux personnes physiques et institutions qui ont contribué à rendre meilleur notre travail, notamment :

- M. Eugène C. EZIN, notre Maître de mémoire et Directeur de l'Institut de Formation et de Recherche en Informatique (IFRI) ;
- M. Lionel METONGNON, doctorant en sécurité informatique et enseignant à l'IFRI ;
- M. Armand, ... et enseignant à l'IFRI ;
- Tous les enseignants de l'IFRI.

Que tous ceux qui nous ont aidé, de près ou de loin, trouvent ici l'expression de nos sentiments les meilleurs.

Résumé

Resume en francais

Mots clés : sécurité ,paiement, factures, sbee,

Abstract

Resume en anglais

Key words: ....,....,....

Introduction

L'informatique, science du traitement rationnel et automatisé de l'information, est devenue aujourd'hui un outil indispensable pour l'évolution de toute nation, de toute société. Elle est passée d'un statut luxueux à un statut nécessaire. La preuve : toute structure de renom investit un budget non négligeable pour l'informatisation de ses opérations. Cependant, certains sont ne sont toujours pas en phase avec l'évolution de la technologie en rapport avec les services qu'ils offrent et la sécurité de ces dernières. Dans le même temps, des individus profitent de ce phénomène pour piéger les autres ignorant le fonctionnement de l'ordinateur ou non dans le but de voler ou altérer des informations précieuses et confidentielles pour des fins diverses : c'est la cybercriminalité. Vu l'ampleur que prend ces différents crimes, il devient nécessaire de voir autrement l'importance de la sécurité dans tous les systèmes d'informations.

Contexte et justification

Le Bénin s'inscrit depuis quelques années dans une politique de restructuration du secteur numérique pour y insuffler une nouvelle dynamique. Plusieurs projets et programmes sont donc nés de cette volonté et constituent une feuille de route pour les acteurs au cœur de cette restructuration.
Dans l'optique de contribuer au développement de ce secteur et surtout de faciliter la vie à la population béninoise, nous avons pensé à mettre en place une plateforme permettant le paiement à distance des factures d'électricité de la SBEE. Cependant, il s'agit d'un système assez critique lorsque nous considérons la quantité et la criticité/le caractère critique du flux d'informations que nous aurons à traiter. La sécurité de l'information revêtant une importance capitale pour la survie d'un peuple, il faudra aussi mettre en oeuvre un système de défense face aux menaces pour réduire l'impact des attaques et essayer, au maximum, d'éliminer les risques. Un aucun aspect ne doit être pris à la légère.

Problématique

Il n'est plus à démontrer que l'électricité est à la base de tout développement. De la disponibilité de l'énergie dépend la satisfaction de tous les besoins humains fondamentaux : l'eau, l'alimentation, la santé, l'éducation. L'option la plus accessible est de souscrire à un abonnement post-payé avec la SBEE. Néanmoins il n'est pas rare de constater qu'après de longues heures d'attente au guichet de la SBEE, l'on vous dise: "Désolé monsieur nous avons un problème de connexion. Veuillez repasser demain.". Il faudra donc repasser plus tard pour solder la facture alors que nous n'avons pas forcément assez de temps pour cela.
Nous sommes aussi parfois confrontés à l'oubli des factures non payées. Cependant quand vous n'êtes pas à jour après un délai d'environ un mois, un agent peut passer à tout moment couper le courant et vous devez payer des pénalités. Notre travail consistera donc à mettre en place une application web permettant le

paiement des factures depuis un téléphone portable ou un ordinateur connecté à Internet. Cependant les données gérees par cette application sont sensibles et critiques. A titre d'exemple les informations de votre carte VISA[1] pourraient être volées. La carte sera donc utilisée à votre insu et votre argent ne vous appartiendra plus. Il sera donc nécessaire de prendre en compte toutes les menaces possibles et d'y apporter des solutions efficientes. Ainsi nous pourrions avoir une application sécurisée permettant le paiement des factures électriques depuis un appareil connecté à Internet.

Objectifs

Notre projet a pour objectif principal de faciliter le paiement des factures électriques de la SBEE et de mettre en place les garde four nécessaire pour réduire les attaques informatiques.

Plus précisément, il s'agira de :

- Rappeler aux personnes utilisant la plateforme qu'ils ont des impayés (via des SMS et/ou des emails)
- Assurer la disponibilité complète de la plateforme afin que les consultations et paiements puissent se faire à n'importe quel moment.
- Garantir la sécurité de toutes les transactions financières effectuées via la plateforme
- Construire un historique afin d'avoir une trace des factures payées et impayées ainsi que des dépenses.

Environnement de stage

Ce travail à été réalisé dans les locaux de la Direction des Systèmes Informatiques (DSI) de la Société Béninoise d'Energie Electrique (SBEE) - Direction Générale. La SBEE a pour mission de produire, de transporter et de distribuer l'énergie électrique sur l'ensemble du territoire national. Elle a son siège social à Cotonou - Ganhi et couvre le territoire national à travers huit (8) Directions Régionales et trente-neuf (39) agences géographiquement réparties dans tous les départements du pays.

Organisation du mémoire

Le présent travail se présente en trois (03) chapitres. Le premier concerne la revue de littérature sur le système existant (actuel) permettant le paiement des factures et quelques notions par rapport à la sécurité des applications Web. Le deuxième chapitre aborde les choix organisationnels (procédures) et techniques opérés en vue de la conception et de la réalisation des solutions proposées. Le troisième chapitre, quant à lui, fait une analyse critique des résultats issus de nos tests après les avoir exposés.

[1] Carte de paiement émise par l'établissement bancaire de son titulaire

Chapitre 1

Revue de littérature

Introduction

Pour bien réaliser un projet, un état des lieux permettant de faire un point sur le sujet du projet est requis. Ainsi, ce chapitre présente un état des lieux concernant les intrusions informatiques. Dans un premier temps, nous présenterons quelques solutions existantes, puis nous aborderons les généralités sur les intrusions informatiques.

G D'OR

Informations receuillies par rapport à Gd'or.

- Gestion Clientele Electricité et/ou Eau
- Comptabilité Générale, Analytique, Budgétaire
- Paie et G estion des Ressources Humaines
- Gestion des stocks
- Gestion des Approvisionnements
- Immobilisations
- Monétique

Sécurité des applications Web

Au siècle de l'information, la cyber-sécurité est un défi omniprésent. Les attaques peuvent cibler n'importe quel site internet pour un nombre croissant de raisons. Un manque de sécurité dans une application peut causer des dysfonctionnements, des arrêts complets de service ainsi que des pertes de données dommageables pour les utilisateurs et l'image de l'entreprise. Pour s'en prémunir, les entreprises doivent mettre en place des mesures de sécurité strictes dans le processus de développement d'applications Web.
Cette section expose quelques failles de sécurité liées aux applications web et quelques bonnes pratiques pour remédier.

Notion de hacker et de cracker

Un hacker est une personne qui, par jeu, goût du défi ou souci de notoriété, cherche à contourner les protections d'un logiciel, à s'introduire frauduleusement dans un système ou un réseau informatique. [**?**] [1].
Un cracker, quant à lui, s'introduit tout aussi frauduleusement dans un système informatique pour en entraver ou en fausser le fonctionnement. Son action est souvent plus dévastatrice.

Mode opératoire d'une attaque informatique

Une attaque informatique peut être décomposé en une suite d'étapes ou phases. Lorsqu'elles sont réunies, ces étapes forment une méthodologie complète pour mener à bien une attaque informatique. L'établissement d'une méthodologie permet de décomposer une procédure complexe en une suite de tâches gérables de taille plus réduite. Ainsi nous regroupons cette méthodologie en quatre étapes qui sont **la reconnaissance**, **les scans**, **l'exploitation**, **la post-exploitation et le maintien d'accès** [2][**?**].

La reconnaissance

La reconnaissance, ou recueil d'informations, est probablement la plus importante des quatre phases. Plus le hacker passe du temps à collecter des informations sur sa cible, plus les phases suivantes auront une chance de réussir[**?**]. En effet la reconnaissance permet de connaitre la cible dans les détails, de connaître les points forts et surtout les points faibles afin de notifier les prochaines possibilités d'attaque.

Les scans

Les scans sont des procédés ayant pour objectif d'identifier les systèmes actifs et les services qui existent sur les systèmes scannés. Dans ce cadre, l'hacker prend le soin de vérifier l'activité d'un système, de trouver les portes ouvertes (les ports), de vérifier les processus tournant sur le système et d'aller à la recherche des vulnérabilités. Ce stade requiert une compréhension plus avancée des systèmes informatiques pour mieux comprendre les résultats recueillis [3][**?**].

L'exploitation

En termes simples, l'exploitation consiste à obtenir un contrôle sur un système. Toutefois, il est à notifier que tout exploit ne conduit pas à la compromission intégrale d'un système. Un hacker peut se servir donc d'un exploit pour télécharger des contenus dont il ne détient pas la propriété pendant qu'un autre utilise un exploit pour crypter les fichiers du système. L'utilisation de l'un des exploits [4] dépend donc de l'objectif visé par l'hacker [**?**].

Post exploitation et maintien d'accès

Cette étape consiste à couvrir les traces de l'intrus agissant afin de ne pas se faire repérer [**?**]. Il permet aussi à ce dernier de faciliter ses prochains accès à la machine victime de ses attaques par l'installation de portes dérobées communément appelées "Backdoor". Ainsi, il n'aura plus besoin de reprendre toutes les étapes de son processus pour accéder à la machine dont il prend le contrôle.

[1] Recommandation officielle : fouineur.
[2] La post exploitation et le maintien d'accès forment une étape.
[3] Informations recueillis au cours du scan
[4] Un exploit est le moyen par lequel un attaquant, ou un pentester en l'occurrence, profite d'un défaut dans un système, une application ou un service.

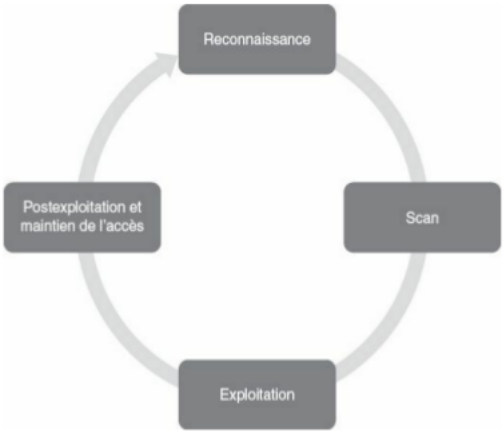

FIGURE 1.1 – Méthodologie ZEH, Patrcick Engebretson, "Les bases du hacking", PEARSON 2013

La méthodologie d'attaque étant cernée, nous allons maintenant présenter les différentes failles auxquelles sont exposées les applications web.

Menaces et risques applicatifs

Failles de Sécurité

Le projet OWASP (Open Web Application Security Project) est une communauté ouverte destinée à permettre aux organisations de développer, d'acheter et de gérer des applications et des API fiables. Tous les outils, documents, vidéos, présentations et chapitres OWASP sont gratuits et ouverts à toute personne intéressée par l'amélioration de la sécurité des applications. L'approche de la sécurité des applications en tant que problème de personnes, de processus et de technologie est préconisée, car les approches les plus efficaces en matière de sécurité des applications nécessitent des améliorations dans ces domaines. OWASP produit aussi de nombreux types de matériaux de manière collaborative, transparente et ouverte. Le projet soutient la recherche innovante en matière de sécurité avec des subventions et des infrastructures.

L'OWASP Top 10 est un document de sensibilisation puissant pour la sécurité des applications Web. Il représente un large consensus sur les risques de sécurité les plus critiques pour les applications Web. Les membres du projet incluent une variété d'experts en sécurité du monde entier qui ont partagé leur expertise pour produire cette liste. Nous recommandons à toutes les entreprises d'adopter ce document de sensibilisation au sein de leur organisation et de commencer à faire en sorte que leurs applications Web minimisent ces risques. L'adoption du Top 10 de l'OWASP est peut-être la première étape la plus efficace pour changer la culture de développement de logiciels au sein de votre organisation en une culture qui produit du code sécurisé.

Cette mise à jour majeure (celle de 2017) ajoute plusieurs nouveaux problèmes, dont deux problèmes sélectionnés par la communauté - **A8 : 2017 - Désérialisation non sécurisée** et **A10 : 2017 - Insuffisance de Logs et de Surveillance**. Les deux principales différences par rapport aux versions précédentes du Top 10 d'OWASP sont les retours substantiels de la communauté et les données complètes rassemblées par des dizaines d'organisations, probablement la plus grande quantité de données jamais rassemblées dans la préparation d'une norme de sécurité applicative. Cela nous permet de croire que le nouveau Top 10 d'OWASP aborde les risques de sécurité applicative les plus importants auxquels sont confrontées les entreprises.

L'un des principaux objectifs du Top 10 d'OWASP est d'éduquer les développeurs, les concepteurs, les architectes, les gestionnaires et les organisations sur les conséquences des faiblesses les plus courantes et les plus importantes de la sécurité des applications Web. Le Top 10 fournit des techniques de base pour se protéger contre ces zones problématiques à haut risque et fournit des indications sur les endroits où aller.

Les attaquants peuvent potentiellement utiliser de nombreux chemins différents à travers votre application pour nuire à votre entreprise ou organisation. Chacun de ces chemins représente un risque qui peut ou non être suffisamment sérieux pour justifier une attention. Parfois, ces chemins sont triviaux à trouver et à exploiter, et parfois ils sont extrêmement difficiles. De même, le préjudice causé peut être sans conséquence, ou vous mettre à la faillite. Pour déterminer le risque pour votre organisation, vous pouvez évaluer la probabilité associée à chaque agent de menace, vecteur d'attaque et faiblesse de la sécurité et le combiner avec une estimation de l'impact technique et commercial sur votre organisation. Ensemble, ces facteurs déterminent votre risque global.

OWASP Top 10 - 2017

Le rapport « OWASP Top 10 » permet ainsi à l'équipe projet de se focaliser sur la protection de l'application Web face aux menaces les plus importantes, ce qui est moins couteux et plus facilement réalisable que d'essayer de se protéger de tous les dangers. L'OWASP établit le classement 2017 ci-dessous, dont chacune des failles est développée dans les chapitres suivants :

1. **Injection**

   Des erreurs d'injection, telles que l'injection SQL, NoSQL, OS et LDAP, se produisent lorsque des données non fiables sont envoyées à un interpréteur dans le cadre d'une commande ou d'une requête. Les données hostiles de l'attaquant peuvent amener l'interpréteur à exécuter des commandes inattendues ou à accéder aux données sans autorisation appropriée.

   L'injection peut parfois conduire à une prise en charge complète de l'hôte. L'impact métier dépend des besoins de l'application et des données.

   La prévention de l'injection nécessite de séparer les données des commandes et des requêtes. L'option préférée consiste à utiliser une API sécurisée, qui évite l'utilisation complète de l'interpréteur ou fournit une interface paramétrée, ou migre pour utiliser les outils ORC (Object Relational Mapping Tools)

2. **Authentification brisée**

   Les fonctions d'application liées à l'authentification et à la gestion de session sont souvent incorrectement implémentées, permettant aux pirates de compromettre les mots de passe, clés ou jetons de session ou d'exploiter d'autres failles d'implémentation pour prendre temporairement ou définitivement les identités des autres utilisateurs. La prévalence de l'authentification brisée est généralisée en raison de la conception et de la mise en œuvre de la plupart des contrôles d'identité et d'accès. La gestion de session est le fondement des contrôles d'authentification et d'accès et est présente dans toutes les applications avec état.

   Les attaquants doivent avoir accès à seulement quelques comptes ou à un seul compte admin pour compromettre le système. Selon le domaine de l'application, cela peut permettre le blanchiment d'argent, la fraude à la sécurité sociale et le vol d'identité, ou divulguer des informations hautement sensibles protégées par la loi.

   Lorsque cela est possible, implémentez l'authentification multi-facteur pour éviter les attaques automatisées, le bourrage des informations d'identification, la force brute et la réutilisation des informations

d'identification volées. Ne pas envoyer ou déployer avec des informations d'identification par défaut, en particulier pour les utilisateurs d'administration.

3. **Exposition de données sensibles**

   De nombreuses applications Web et API ne protègent pas correctement les données sensibles, telles que les données financières, les soins de santé et les informations personnelles. Les attaquants peuvent voler ou modifier de telles données faiblement protégées pour effectuer une fraude par carte de crédit, un vol d'identité ou d'autres crimes. Les données sensibles peuvent être compromises sans protection supplémentaire, telles que le cryptage au repos ou en transit, et nécessitent des précautions spéciales lors d'un échange avec le navigateur.

   Au cours des dernières années, cela a été l'attaque la plus courante. La faille la plus fréquente est simplement de ne pas chiffrer les données sensibles. Lorsque la cryptographie est utilisée, la génération et la gestion de clés faibles et la faible utilisation d'algorithmes, de protocoles et de chiffrements sont courants, en particulier pour les techniques de stockage de hachage avec mot de passe faible. Généralement, ces informations incluent des informations personnelles sensibles telles que des dossiers médicaux, des informations d'identification, des données personnelles et des cartes de crédit, qui nécessitent souvent une protection telle que définie par les lois ou réglementations telles que le GDPR UE ou les lois locales sur la confidentialité.

   Classer les données traitées, stockées ou transmises par une application. Identifiez les données sensibles en fonction des lois sur la confidentialité, des exigences réglementaires ou des besoins de l'entreprise. Appliquer les contrôles selon la classification.

4. **Entités externes XML (XXE)**

   De nombreux processeurs XML anciens ou mal configurés évaluent les références d'entités externes dans les documents XML. Les entités externes peuvent être utilisées pour divulguer des fichiers internes à l'aide du gestionnaire d'URI de fichier, des partages de fichiers internes, de l'analyse de port interne, de l'exécution de code à distance et des attaques par déni de service.

   Par défaut, de nombreux processeurs XML plus anciens permettent la spécification d'une entité externe, un URI qui est déréférencé et évalué pendant le traitement XML. Ces failles peuvent être utilisées pour extraire des données, exécuter une requête à distance à partir du serveur, analyser des systèmes internes, effectuer une attaque par déni de service et exécuter d'autres attaques.

   Autant que possible, utilisez des formats de données moins complexes tels que JSON et évitez la sérialisation des données sensibles. Corrigez ou mettez à niveau tous les processeurs et bibliothèques XML utilisés par l'application ou sur le système d'exploitation sous-jacent.

5. **Contrôle d'accès brisé**

   Les restrictions sur ce que les utilisateurs authentifiés sont autorisés à faire ne sont souvent pas correctement appliquées. Les attaquants peuvent exploiter ces failles pour accéder à des fonctionnalités et / ou données non autorisées, telles que l'accès aux comptes d'autres utilisateurs, l'affichage de fichiers sensibles, la modification des données d'autres utilisateurs, la modification des droits d'accès, etc.

Les faiblesses du contrôle d'accès sont courantes en raison du manque de détection automatisée et du manque de tests fonctionnels efficaces par les développeurs d'applications. L'impact technique est celui des attaquants agissant en tant qu'utilisateurs ou administrateurs, ou des utilisateurs utilisant des fonctions privilégiées, ou créant, accédant, mettant à jour ou supprimant chaque enregistrement.

Le contrôle d'accès n'est efficace que s'il est appliqué dans un code côté serveur approuvé ou dans une API sans serveur, où l'attaquant ne peut pas modifier la vérification du contrôle d'accès ou les métadonnées. À l'exception des ressources publiques, refus par défaut. Implémentez les mécanismes de contrôle d'accès une fois et réutilisez-les dans toute l'application, y compris en minimisant l'utilisation de CORS.

6. **Mauvaise configuration de la sécurité**

La mauvaise configuration de la sécurité est le problème le plus souvent rencontré. Ceci est généralement le résultat de configurations par défaut non sécurisées, de configurations incomplètes ou ad hoc, d'un stockage cloud ouvert, d'en-têtes HTTP mal configurés et de messages d'erreur détaillés contenant des informations sensibles. Non seulement tous les systèmes d'exploitation, cadres, bibliothèques et applications doivent être configurés de manière sécurisée, mais ils doivent également être corrigés / mis à niveau en temps opportun.

Une mauvaise configuration de sécurité peut se produire à n'importe quel niveau d'une pile d'application, notamment les services réseau, la plateforme, le serveur Web, le serveur d'applications, la base de données, les frameworks, le code personnalisé et les machines virtuelles pré-installées. De telles failles donnent souvent aux attaquants un accès non autorisé à certaines données ou fonctionnalités du système. Parfois, de tels défauts entraînent un compromis complet du système.

Des processus d'installation sécurisés devraient être mis en œuvre. Par exemple, un processus de renforcement répétable qui permet de déployer rapidement et facilement un autre environnement correctement verrouillé. Les environnements de développement, d'assurance qualité et de production doivent tous être configurés de manière identique, avec des informations d'identification différentes utilisées dans chaque environnement. Ce processus devrait être automatisé pour minimiser les efforts requis pour installer un nouvel environnement sécurisé.

7. **Cross-Site Scripting (XSS)**

Les failles XSS se produisent chaque fois qu'une application inclut des données non fiables dans une nouvelle page Web sans validation ou échappée, ou met à jour une page Web existante avec des données fournies par l'utilisateur en utilisant une API de navigateur pouvant créer du code HTML ou JavaScript. XSS permet aux attaquants d'exécuter des scripts dans le navigateur de la victime, ce qui peut détourner des sessions utilisateur, dégrader des sites Web ou rediriger l'utilisateur vers des sites malveillants. XSS est le deuxième problème le plus répandu dans le Top 10 d'OWASP, et se retrouve dans environ deux tiers de toutes les applications.

L'impact de XSS est modéré pour XSS réfléchi et DOM XSS, et sévère pour XSS stocké, avec l'exécution de code à distance sur le navigateur de la victime, comme voler des informations d'identification, des sessions, ou livrer des logiciels malveillants à la victime.

La prévention de XSS nécessite la séparation des données non fiables du contenu du navigateur actif. Cela peut être réalisé en utilisant des frameworks qui échappent automatiquement à XSS par conception, comme le dernier Ruby on Rails, React JS. Apprenez les limites de la protection XSS de chaque framework et gérez correctement les cas d'utilisation qui ne sont pas couverts. L'échappement de données de requête HTTP non fiables en fonction du contexte de la sortie HTML (corps, attribut, JavaScript, CSS ou URL) résoudra les vulnérabilités XSS Reflected[5] et Stored[6].

8. **Désérialisation non sécurisée**

La désérialisation non sécurisée conduit souvent à l'exécution de code à distance. Même si les failles de désérialisation n'aboutissent pas à l'exécution de code à distance, elles peuvent être utilisées pour effectuer des attaques, y compris des attaques de relecture, d'injection et d'escalade de privilèges.

Certains outils peuvent détecter des défauts de désérialisation, mais une assistance humaine est souvent nécessaire pour valider le problème. L'impact des défauts de désérialisation ne peut pas être surestimé. Ces failles peuvent mener à des attaques d'exécution de code à distance, l'une des attaques les plus graves possibles.

Le seul modèle architectural sûr est de ne pas accepter les objets sérialisés provenant de sources non fiables ou d'utiliser des supports de sérialisation qui autorisent uniquement les types de données primitifs. Si cela n'est pas possible, il faut faire une implémentation de contrôles d'intégrité tels que les signatures numériques sur tous les objets sérialisés pour empêcher la création d'objets hostiles ou la falsification de données.

9. **Utilisation de composants avec des vulnérabilités connues**

Les composants, tels que les bibliothèques, les frameworks et autres modules logiciels, fonctionnent avec les mêmes privilèges que l'application. Si un composant vulnérable est exploité, une telle attaque peut faciliter la perte de données sérieuse ou la prise de contrôle du serveur. Les applications et les API utilisant des composants présentant des vulnérabilités connues peuvent compromettre les défenses de l'application et permettre diverses attaques et impacts.

La prévalence de ce problème est très répandue. Les modèles de développement à forte composante peuvent amener les équipes de développement à ne plus comprendre quels composants ils utilisent dans leur application ou leur API, et encore moins les tenir à jour. Bien que certaines vulnérabilités connues n'entraînent que des impacts mineurs, certaines des violations les plus importantes à ce jour reposent sur l'exploitation de vulnérabilités connues dans les composants. Selon les actifs que vous protégez, ce risque devrait peut-être figurer en tête de liste.

Un processus de gestion des correctifs devrait être en place pour supprimez les dépendances non utilisées, les fonctions inutiles, les composants, les fichiers et la documentation. Abonnez-vous à des alertes par e-mail pour connaître les failles de sécurité liées aux composants que vous utilisez. N'obtenez que des composants de sources officielles sur des liens sécurisés. Préférer les packages signés pour réduire les risques d'inclusion d'un composant malveillant modifié.

10. **Insuffisance de Logs et de Surveillance**

[5] courte definition ici
[6] courte definition ici

Une journalisation et une surveillance insuffisantes, couplées à une intégration manquante ou inefficace avec la réponse aux incidents, permettent aux attaquants d'attaquer davantage les systèmes, de maintenir la persistance, de pivoter vers plus de systèmes et d'altérer, extraire ou détruire des données. La plupart des études de violation montrent que le temps de détection d'une violation dépasse 200 jours, généralement détectés par des parties externes plutôt que par des processus internes ou de surveillance.

Une stratégie pour déterminer si vous avez une surveillance suffisante est d'examiner les journaux après les tests de pénétration. Les actions des testeurs doivent être enregistrées suffisamment pour comprendre les dommages qu'ils ont pu infliger. La plupart des attaques réussies commencent par un sondage de vulnérabilité. Permettre de telles sondes de continuer peut augmenter la probabilité d'exploitation réussie à près de 100%.

En fonction du risque de stockage ou de traitement des données par l'application : Assurez-vous que tous les échecs de connexion, de contrôle d'accès et de validation des entrées côté serveur peuvent être consignés avec un contexte utilisateur suffisant pour identifier les comptes suspects ou malveillants, et conservés suffisamment longtemps pour permettre une analyse légale retardée. Assurez-vous que les journaux sont générés dans un format pouvant être facilement utilisé par des solutions de gestion de journaux centralisées.

Quelques bonnes pratiques

a confirmer Parler des bonnes pratiques lors du developpement. Utiliser le OWASP Top 10 Proactive Controls V3 comme guide.

Conclusion

Les attaques informatiques sont nombreuses et de différentes formes. Dans ce chapitre, nous avons présenté le fonctionnement de quelques-unes d'entre elles et exposer certaines solutions existantes. Dans le chapitre suivant, nous aborderons notre solution à travers sa conception et les outils utilisés pour sa réalisation.
Pour conclure, la sécurité des applications web est un point qu'il ne faut pas négliger. C'est un travail qui peut paraitre fastidieux, coûteux, pas forcément très utile pour des petites structures / sites mais quoi de mieux pour la sérénité et la satisfaction client que de savoir que son application est robuste et ne flanchera pas sous les attaques du premier pirate venu ?

Chapitre 2

Conception et Matériels

Introduction

La programmation d'une application requiert d'abord l'organisation et la documentation de ses idées. La définition des modules induit les différentes étapes de sa réalisation. C'est cette démarche antérieure à l'écriture que l'on appelle modélisation. Ainsi, notre modélisation est axée autour de deux diagrammes : le diagramme des cas d'utilisation recense les différentes fonctionnalités de notre système ; le diagramme de séquence illustre le fonctionnement interne du système dans le temps.

Conception

Méthode de modélisation

Afin de modéliser les fonctionnalités de notre solution, nous avons choisi le langage UML *Unified Modeling Language* [**?**]. Issu d'un large consensus, le langage UML garantit la stabilité et la performance d'un projet grâce à son caractère formel et industrialisé. Aussi facilite-t-il la compréhension du système par l'usage de représentations graphiques appelées diagrammes. Ces diagrammes nous ont permis de modéliser notre solution en utilisant les diagrammes de cas d'utilisation et de séquence.

Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation représente la structure des grandes fonctionnalités nécessaires aux utilisateurs du système. C'est le premier diagramme du modèle UML, celui où s'assure la relation entre l'utilisateur et les objets que le système met en œuvre.

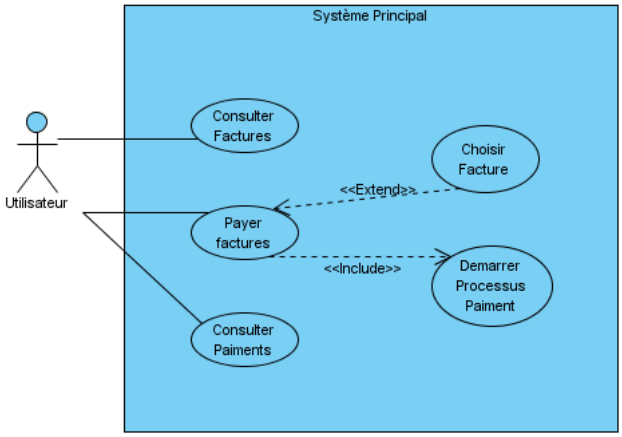

FIGURE 2.1 – Diagramme de cas d'utilisation

Pour une première utilisation, il est nécessaire de créer un compte. Les informations à fournir sont : un nom, un email, un numéro de téléphone, la référence abonnée[1] et le mot de passe pour la protection du compte. Ces informations étant obligatoires. Un mail d'activation est envoyé et l'utilisateur après activation de son compte peut se connecter à l'application. Une authentification est nécessaire afin d'avoir accès aux cas d'utilisations de notre diagramme (ou aux fonctionnalités de l'application).

Diagramme de séquence

Le diagramme de séquence représente la succession chronologique des opérations réalisées par un acteur. Ce mode de représentation effectue la description du fonctionnement dynamique du système. En d'autres termes, il indique les objets que l'acteur va manipuler et les opérations qui font passer d'un objet à l'autre.

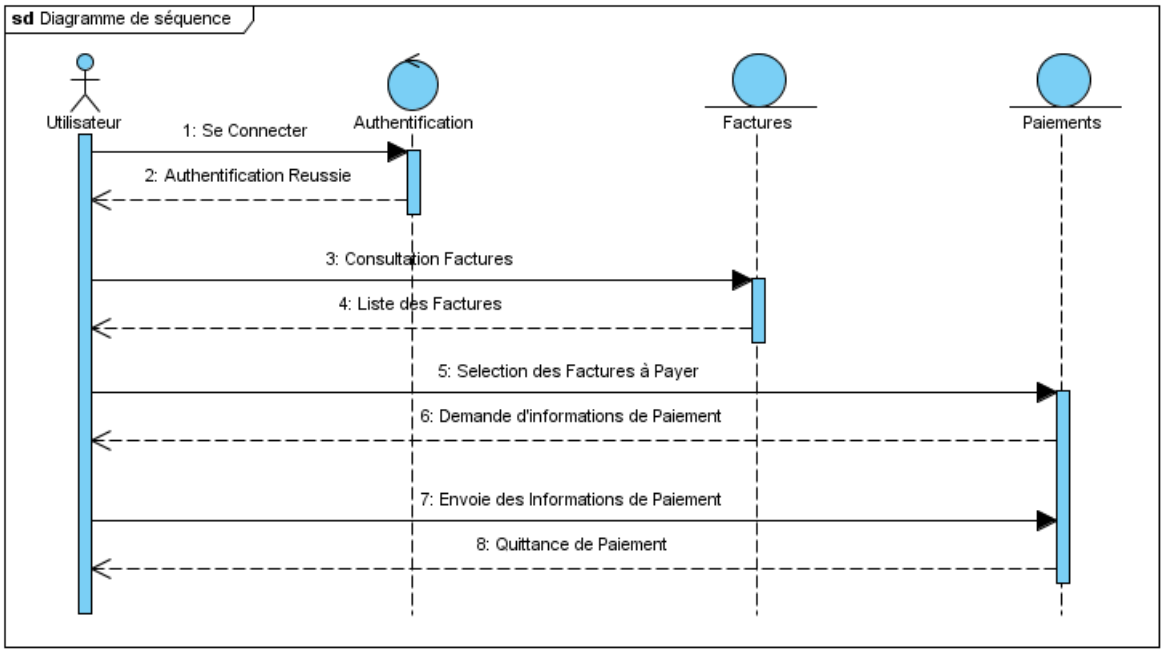

FIGURE 2.2 – Diagramme de séquence

Après que l'utilisateur ait entré ses informations, identifiants pour se connecter, il a accès aux factures. Là il peut ajouter les factures impayées à la liste des factures à payer (même concept que le panier d'un site

[1] trouver le sens exact

d'e-commerce). Il pourra donc procéder au paiement de ses factures via le mode de paiement qui lui convient. Une quittance lui est générée et est ajoutée à son historique de paiements.

Matériels

Principe de fonctionnement de notre solution

Les factures d'un utilisateur sont identifiées grâce à la référence abonnée fournie à l'inscription. Une fois connecté à l'application, le client dispose de la liste de ses factures. Il peut donc sélectionner le ou les factures qu'il désire solder et procéder au paiement. Le progiciel G'd'Or génère tous les jours à une heure précise un fichier contenant les factures impayées de tous les compteurs de la SBEE. Ce fichier est reçu par le serveur d'application et ensuite chargé dans la base de donnée. Ainsi nous disposons des impayées au niveau de tous les numéros de compteurs. A chaque fin de journée, à une heure précise, l'application génère à son tour un fichier contenant les factures qui ont été payées dans la journée et le total encaissé. Nous considérons ce ficher comme un bilan fait par notre application à Gd'OR. Ce fichier permet à Gd'Or de valider les paiements et d'apurer les comptes en fonction des informations fournies par les prestataires de solution de paiement. Tous les échanges de fichiers se feront via un canal sécurisé (SFTP et VPN). Apres utilisations, ils sont archivés de façon automatique afin de garder une trace des échanges et des flux de données. Les erreurs notifiées sont immédiatement signalées aux entités concernées et ils doivent y apporter des solutions le plus tôt possible.

Exemples de fichiers :

- Impayées
- Payés

L'architecture de la solution se présente comme suit :

- les clients
- internet
- le systeme de paiement
- les serveurs
- les routeurs, parefeu

Langages de développements et outils

Notre solution est une application web. Elle est donc accessible depuis tous les systèmes d'exploitations du moment qu'un navigateur et d'une connexion internet sont disponibles. Elle a été réalisé grâce au framework Django.

Pourquoi utiliser un framework ?

Lorsque l'on réalise un site Internet, on en revient toujours aux même étapes :

- réalisation et codage du design ;
- réalisation des modules :
  - réalisation du modèle de données concernant le module,
  - réalisation des formulaires d'ajout, modification et suppression des données :
- réalisation des pages d'affichage du contenu du site ;

- réalisation d'une administration pour gérer les modules ;
- réalisation d'un espace utilisateur avec des droits sur l'accès aux données ;
- mise en place de flux RSS/ATOM ;
- mise en place d'un plan du site ;

Tout cela est relativement répétitif, et si, la première fois, ça peut paraître très amusant, on en arrive rapidement à faire des copier/coller, assez mauvaise méthode car source de nombreuses erreurs. Finalement on regroupe des morceaux de code en fonctions réutilisables. À ce moment, on se rapproche de plus en plus de la notion de framework ci-dessus. L'avantage d'utiliser un framework existant et surtout Open Source tel que Django, c'est que nous ne sommes pas les seuls à l'utiliser, et que les bugs sont donc corrigés plus rapidement, les améliorations sont exécutées par plusieurs personnes et de manière bien mieux réfléchie. C'est d'ailleurs tout l'intérêt d'utiliser un framework. En faire moins, pour en faire plus dans le même temps.

Pourquoi Django ?
Il existe de nombreux framework web, dans différents langages de programmation. Pourquoi utiliser spécifiquement Django et pas un autre ? Nombreuses sont les raisons qui motivent ce choix :

- La simplicité d'apprentissage.
- La qualité des applications réalisées.
- La rapidité de développement.
- La sécurité de l'application.
- La facilité de maintenance des applications sur la durée.

Outres ces avantages on bénéficie de la clarté de Python, qui permet à plusieurs développeurs de travailler sur le même projet. Le style est imposé, donc tout le monde suit les mêmes règles, ce qui facilite les travaux en équipe et la clarté du code.

POSTGRESQL

PostgreSQL est un système de gestion de base de données relationnelle et objet (SGBDRO). C'est un outil libre disponible selon les termes d'une licence de type BSD. Ce système est concurrent d'autres systèmes de gestion de base de données, qu'ils soient libres (comme MariaDB et Firebird), ou propriétaires (comme Oracle, MySQL, Sybase, DB2, Informix et Microsoft SQL Server). Comme les projets libres Apache et Linux, PostgreSQL n'est pas contrôlé par une seule entreprise, mais est fondé sur une communauté mondiale de développeurs et d'entreprises.

Ubuntu 16.04 LTS

Ubuntu 16.04 LTS est la 6ème version LTS du système d'exploitation basé sur Linux, et comme nous en avons discuté ici, elle apporte plusieurs nouvelles fonctionnalités et améliorations. Compte tenu de leur cycle de support, les versions LTS conviennent mieux aux entreprises et aux utilisateurs finaux qui n'aiment pas mettre à jour leur système d'exploitation de temps en temps. Si vous êtes l'un d'entre eux, allez-y et prenez la publication.

Environnement de Production

Debian est un système d'exploitation libre. Il est simple et est constitué de plus de 4000 paquets. Les paquets sont des composants logiciels précompilés conçus pour s'installer facilement sur la machine hôte. L'ouverture de Debian permet à plusieurs programmeurs de pouvoir identifier les failles de sécurité ou de créer plusieurs modules pour faire des tâches qui s'avèrent indispensables. C'est ainsi que la communauté de Debian devient

de plus en plus robuste et flexible. Plusieurs systèmes d'exploitation sont dérivés de Debian dont Kali Linux spécialisé dans les tests d'intrusion.

Conclusion

Dans ce chapitre, nous avons présenté les choix techniques opérés ainsi que notre solution à travers sa modélisation, son principe de fonctionnement et les outils utilisés. Le chapitre suivant exposera les différents résultats et quelques critiques.

Chapitre 3

Résultats et Discussion

Introduction

Dans ce chapitre, nous présentons les résultats des simulations faites pour tester le fonctionnement de notre solution. Dans un premier temps, nous allons présenter quelques fonctionnalités développées dans l'application et nous aborderons ensuite une discussion.

Présentation des résultats des tests

Pour réaliser les simulations, nous avons choisi d'utiliser deux machines toutes linux ; l'une sous kali linux 2.0 Rolling, l'autre sous ubuntu 16.04 LTS.

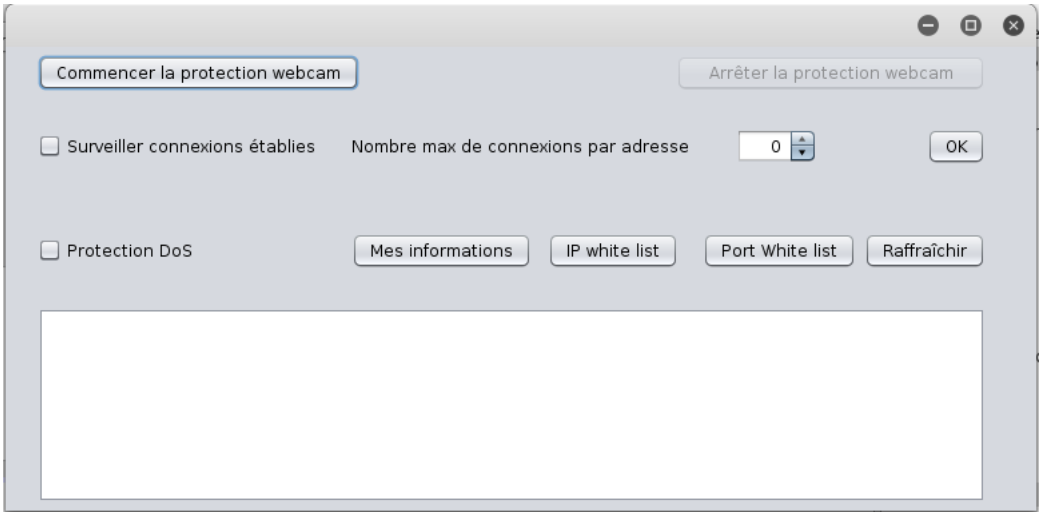

FIGURE 3.1 – Ecran d'accueil de notre solution

Cette figure présente l'écran à l'ouverture de notre logiciel avec les différentes options. Lorsque l'utilisateur clique sur whitelist IP, il est redirigé vers une autre page dont l'image suit.

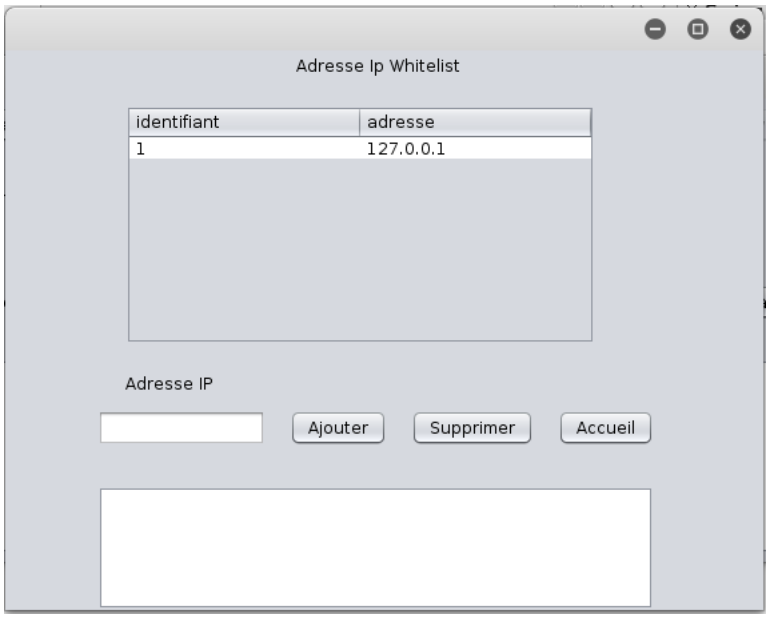

FIGURE 3.2 – Page de la whitelist IP

Lorsqu'il clique sur le bouton réservé à la whitelist des ports, il est renvoyé vers un autre écran dont l'image suit.

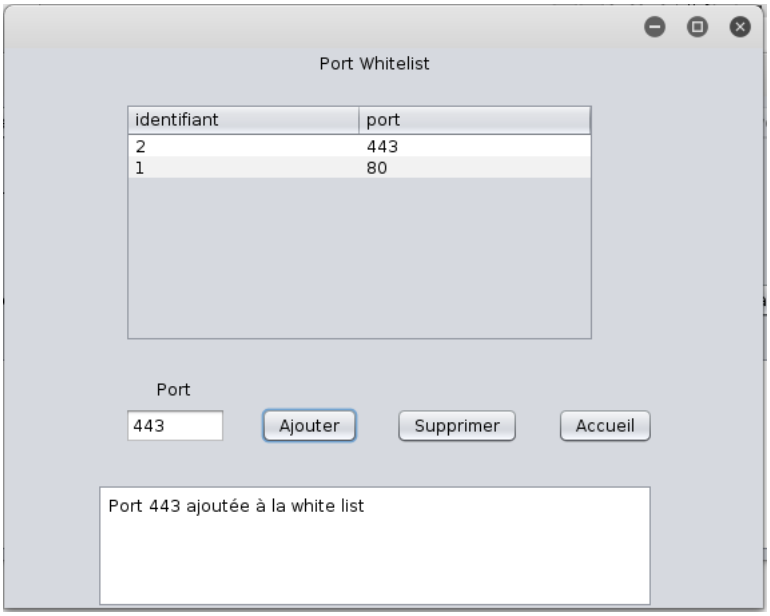

FIGURE 3.3 – Page de la whitelist Port

Les données visualisées sont présentes parce qu'elles ont été entrées préalablement dans la base de données.

Cas pratique

Notre machine cible reçoit un mail comportant un lien pour voir les nouveautés de la communuauté Ubuntu. Après un clic sur le lien, il est redirigé vers un site ressemblant fortement à `community.ubuntu.com`, celui de la communauté Ubuntu, mais qui n'est rien d'autre qu'un site piégé dont voici l'aperçu.

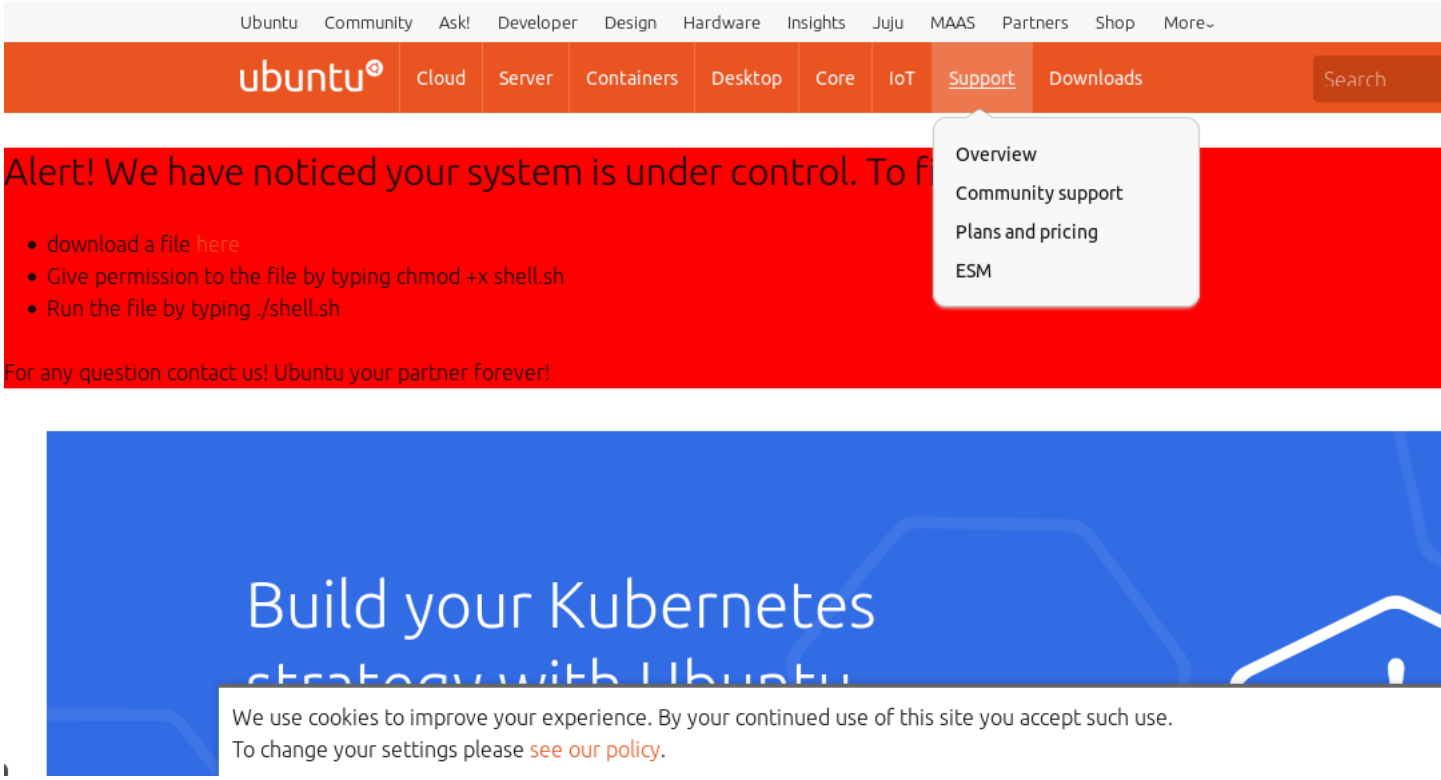

FIGURE 3.4 – Faux site conçu par l'attaquant

L'attaquant ajoute du contenu qui fait croire à la victime que son poste est sous contrôle et qu'il faudrait télécharger et exécuter un fichier contenant un script. Cela met la victime en confiance puisque, selon lui, il est audité depuis la communauté Ubuntu. Cependant, il ne fait que se piéger. L'attaquant utilise le framework Metasploit [**?**] pour lancer une écoute de session (listener[1]) vers toute cible essayant d'exécuter le fichier supposé résoudre le problème notifié sur le faux site. Après exécution du fichier, l'attaquant prend possesion de la machine cible.

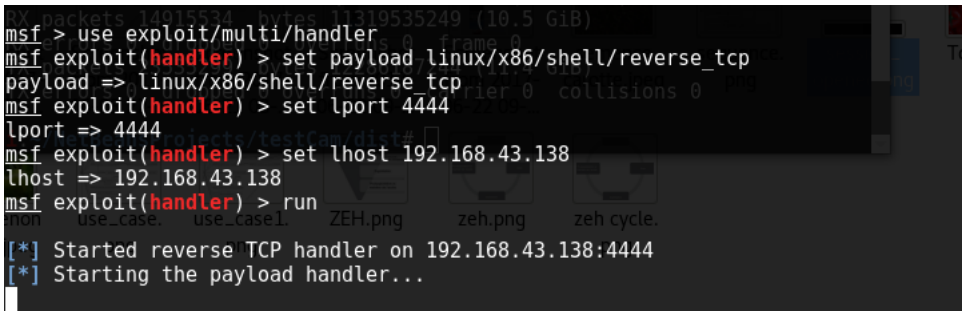

FIGURE 3.5 – Session en attente.

[1]Un listener est un composant de Metasploit qui attend une connexion entrante de tout type.

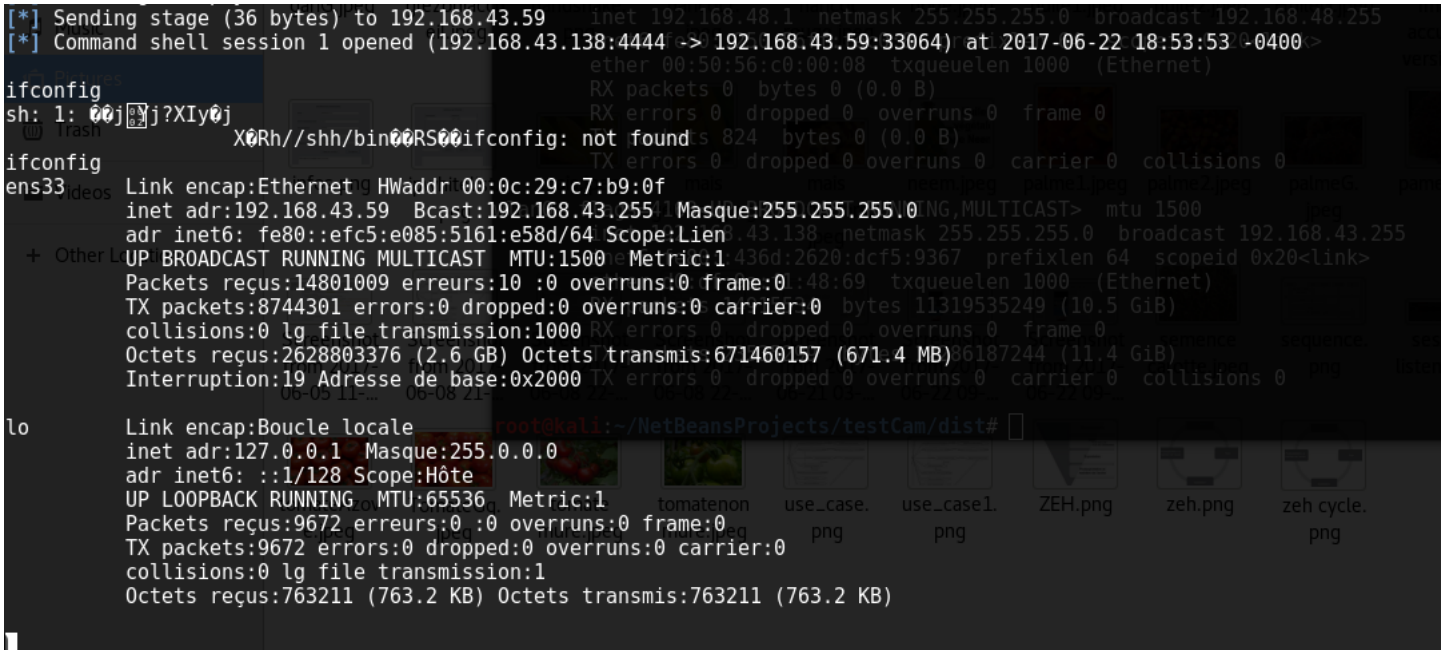

FIGURE 3.6 – Session ouverte vers la machine cible.

Lorsque l'utilisateur de la machine cible active la détection d'intrusions, une alerte lui est envoyée en précisant la connexion établie.

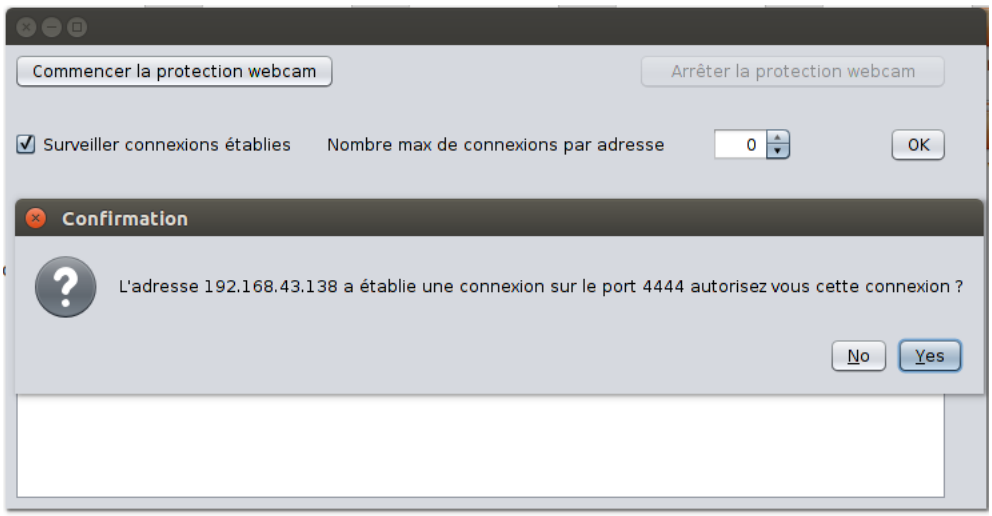

FIGURE 3.7 – Alerte connexion établie

Ne reconnaissant pas cette connexion comme légale, l'utilisateur décide d'arrêter. Cela met fin à la connexion établie et met en liste noire l'adresse source de l'attaquant.

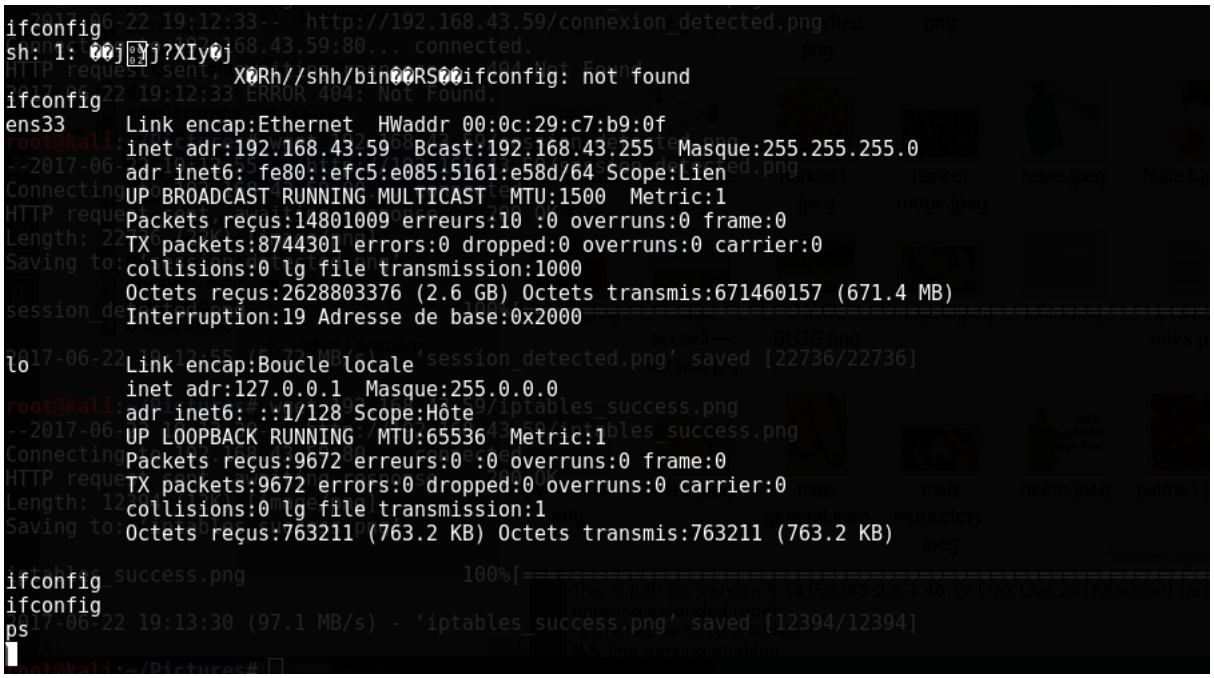

FIGURE 3.8 – Les commandes du shell ne fontionnant plus

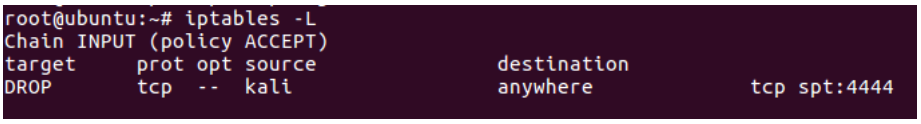

FIGURE 3.9 – Adresse et port en blacklist

Discussion

Notre projet est né de plusieurs constats généraux de l'ordre de la sécurité informatique pour la protection de l'information et de la vie privée.

Les recherches effectuées nous ont permis de constater l'existence de certaines solutions. Cependant, ces solutions, loin de contrer tout type d'intrusion, ne peuvent pas être prises pour seuls moyens de sécurité.

Notre solution apporte un outil de défense contre certaines attaques en réseau informatique parmi lesquelles l'espionnage par webcam, les intrusions clandestines, les dénis de service. Aussi permet-elle de lever le mythe selon lequel on est en sécurité absolue sur les systèmes Linux (un exemple se présente dans le cas pratique).

Ainsi, d'une part, notre solution est simple à utiliser et est accessible à un grand public, celui regroupant les utilisateurs Linux. Elle permet aux utilisateurs d'être informés de quelques processus tournant à leur insu.

D'autre part, notre solution est exclusivement destinée aux systèmes Linux, ce qui représente une limite car, aujourd'hui beaucoup d'ordinateurs tournent sous les sytèmes Windows et Mac, sans compter la multiplicité des versions Linux qui existent.

Conclusion

Dans ce chapitre, nous avons exposé les résultats des tests de notre application et réalisé quelques critiques concernant ses performances et ses insuffisances. Ces insuffisances peuvent être perçues comme des perspectives afin d'améliorer le travail fait pour une utilisation plus éfficiente.

Conclusion

Conclusion

La sécurité informatique passant par la sécurité de l'information, la sécurité de la vie privée n'est plus aujourd'hui un sujet inconnu. Aussi le hacking constitue-t-il aujourd'hui une arme de guerre assez dévastatrice et silencieuse. Dans cette vision, beaucoup d'outils de sécurité se développent de jour en jour afin de restreindre le champ d'attaque des pirates informatiques qui s'arment davantage.

L'objet de ce mémoire a été de mettre en place un système de détection des attaques par webcam et de prévenir par des moyens simples mais efficaces les tentatives d'intrusion. Les solutions proposées proviennent des analyses éffectuées sur les systèmes piratés volontairement. Ainsi les utilisateurs de notre application peuvent être protégés des différentes menaces que nous couvrons.

Dans ce mémoire, nous avons tout d'abord fait une revue de littérature autour des intrusions informatiques. Nous avons ensuite réalisé la conception de la solution que nous avons proposée avant d'exposer les résultats et critiques de l'application que nous avons implémentée.

Bien que notre solution réponde au besoin énoncé, il faut noter certaines insuffisances telles que l'inactivité de l'application dans les intervalles d'attente. Bien que cela soit fait pour éviter l'utilisation abusive des ressources du système, il serait préférable de mettre le système en mode écoute. D'une part, un autre axe de recherche en ce qui concerne les travaux futurs sera d'explorer la possibilité de rendre l'application accessible sur d'autres plateformes telles que Windows, MAC OS et toutes autres distributions Linux. D'autre part, on pourrait envisager de mettre la recherche de symptômes d'attaques en mode écoute en prenant des mesures autonomes. [**?**]

Bibliographie