

## Dédicace

A

A

A

A

A

A

A

A

MES

MES

MES

MES

MES

MES

MES

MES

PARENTS

PARENTS

PARENTS

PARENTS

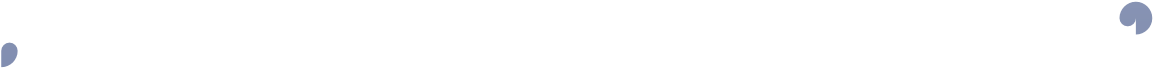
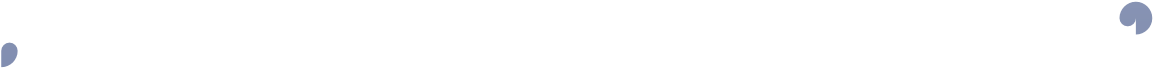
PARENTS

PARENTS

PARENTS

PARENTS

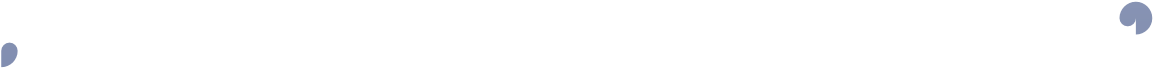
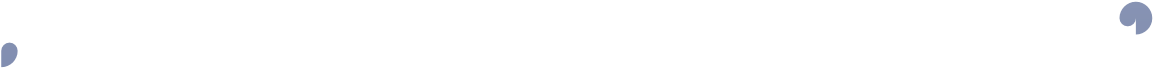
\



## Remerciements

Je tiens à remercier tout d’abord Allah le tout puissant, pour l’intelligence et la force qu’il m’a donné pour atteindre les objectifs de ce mémoire, je tiens également à exprimer ma gratitude et ma reconnaissance aux personnes suivantes :

* **M. GUIMEZAP Paul**, le Président Fondateur de l’IUC pour l’opportunité qu’il offre aux jeunes par le biais de son établissement de recevoir une formation et d’assurer un avenir certain ;
* **Dr. MEUKALEUNI Cyrille**, mon superviseur académique, pour sa disponibilité, ses conseils et son assistance ;
* **M. FOTSO KUATE Victor**, mon encadreur académique, pour son assistance exceptionnelle, et sa disponibilité, qui ont contribué à alimenter ma réflexion ;
* **M. LAGOUE Rodrigue**, mon encadreur professionnel, qui m’a beaucoup appris sur les défis à relever dans le monde des affaires ; les connaissances et expériences dans ce milieu, sa confiance, ses conseils et sa patience ;
* **M. TSOPBENG David**, le Directeur de l’Institut Supérieur des Technologies et du Design Industriel (ISTDI), pour tous ce qu’il a fait en faveur à notre formation ;
* **M. NWOKAM Verlaine,** le Chef de département des masters génie informatique à l’IUC, pour tous ces efforts pendant notre formation ;
* **Dr FOUOKENG**, pour toutes les possibilités qu’il a su mettre à ma disposition pour le compte de ma formation, ses conseils, sa disponibilité, ses encouragements ;
* **M.** **NYAM Aquila**, pour ses conseils, sa disponibilité, ses encouragements ;
* **Ma famille,** pour son soutien, ses conseils qu’elle me donne au quotidien, ainsi que son amour inconditionné ;
* J’adresse mes sincères remerciements à tous les **professeurs**, **intervenants** et toutes les **personnes** ont guidé mes réflexions durant mes recherches ;
* **Mes camarades classes**, amis et collègues, pour leurs aides et leurs disponibilités ;
* Tous ceux qui ont participé de près ou de loin à la rédaction de ce mémoire.



## Sommaire

[Dédicace i](file:///C:\Users\win10\Documents\memo\memo-10.docx#_Toc76232855)

[Remerciements ii](file:///C:\Users\win10\Documents\memo\memo-10.docx#_Toc76232856)

[Sommaire iii](file:///C:\Users\win10\Documents\memo\memo-10.docx#_Toc76232857)

[Abréviations iv](file:///C:\Users\win10\Documents\memo\memo-10.docx#_Toc76232858)

[Liste des figures vi](file:///C:\Users\win10\Documents\memo\memo-10.docx#_Toc76232859)

[Liste des tableaux vii](file:///C:\Users\win10\Documents\memo\memo-10.docx#_Toc76232860)

[Résumé viii](file:///C:\Users\win10\Documents\memo\memo-10.docx#_Toc76232861)

[Abstract ix](file:///C:\Users\win10\Documents\memo\memo-10.docx#_Toc76232862)

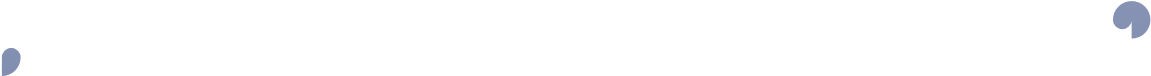
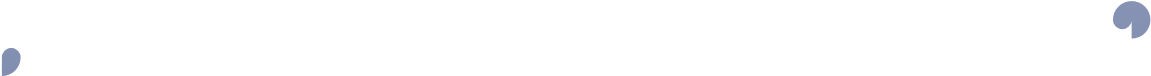
[Partie I : Etat de l’art 1](file:///C:\Users\win10\Documents\memo\memo-10.docx#_Toc76232863)

[Chapitre I : Revue de la littérature 2](#_Toc76232864)

[Section 1. Concepts et définitions 2](#_Toc76232865)

[Section 2. Pentesting 12](#_Toc76232866)

[Section 3. Attaques utilisees pour le pentesting 16](#_Toc76232867)



## Abréviations

**AJAX** : Asynchronious Javascript And Xml

**API :** Application Programming Interface **ASP :** Active Server Pages

**CFTL** : Comité Français de Test Logiciel

**COM:** Component Object Model

**DOM :** Document Object Model

**HTML** : HyperText Markup Language

**HTTP :** HyperText Transfert Protocol

**IBM** : International Business Machines

**IDE** : Integrated Development Environment

**ISTQB** : International Software Testing Qualification Board

**JVM :** Java Virtuel Machine

**MVC :** Modèle Vue Contrôleur

**OLE :** Object Linking and Embeding

**OS :** Operating system

**PC :** Personal Computer

**PDCA** : Plan Do Check Act

**PHP :** Hypertexte PreProcessor

**RAM :** Random Access Memory

**SC Group :** Smart Code Group

**SWT :** Standard Widget Toolkit

**UFT :** Unified Functional Testing

**UI** : User Interface

**UX** : User eXperience

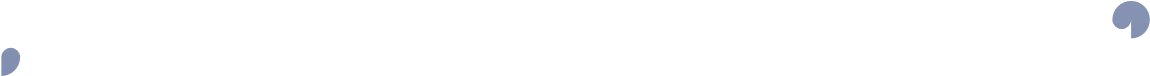
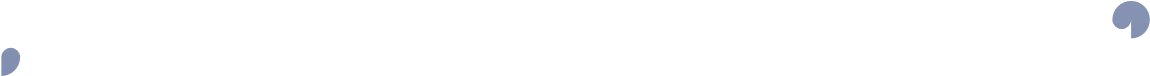
**VB :** Visual Basic

**VS Code** : Visual Studio Code

**VVT** : Validation Vérification et Test Logiciel

**WATIR :** Web Application Testing In Rugy

**BPMN:** Business Process Management and Notation



## Liste des figures

Figure 1. 1. Patch devant réduire des temps de réponse .....................................................7

Figure 2. 1: Processus de test manuel .............................................................................. 16

Figure 2. 2: Succession de tâches avec Gantt .................................................................. 23

Figure 2. 3: Présentation du fonctionnement de Scrum .................................................... 24

Figure 2. 4: Eclipse ......................................................................................................... 25

Figure 2. 5: Logo de Visual Studio Code ........................................................................ 26

Figure 2. 6: Fonctionnement de Protractor ...................................................................... 35

Figure 3.1 : Roue de Deming .......................................................................................... 61

Figure 3.2 : Organigramme de fonctionnement du test après optimisation ....................... 62

Figure 3.3 : Cas d’utilisation du logiciel de test ............................................................... 65

Figure 4.1: Logo de Creately ............................................................................................. 66

Figure 3.4 : Cycle de vie d’un logiciel, cycle en V .......................................................... 67

Figure 3.5 : Architecture de Selenium WebDriver ........................................................... 69

Figure 3.6 : Compatibilité entre navigateurs .................................................................... 71

Figure 4.2: Enregistrement du projet ................................................................................. 76

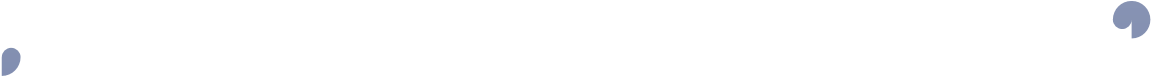
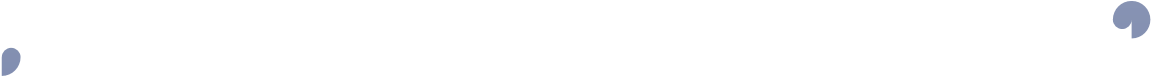
Figure 4.3: Démarrage de test du site smart code group ..................................................... 76

Figure 4.4: Cas de test effectué avec succès ....................................................................... 77

Figure 4.5: Exportation du test dans un IDE ...................................................................... 78

Figure 4.6: Ouverture du test sur Visual Studio Code ........................................................ 79

Figure 4.7: Test effectué avec erreur .................................................................................. 80

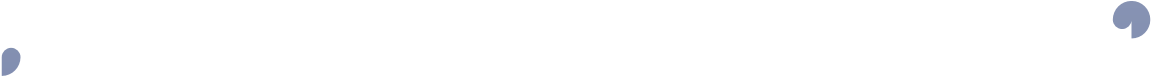
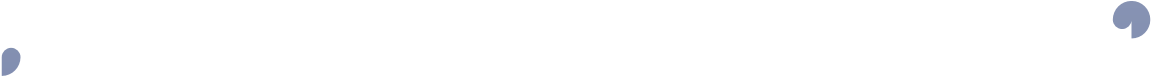


## Liste des tableaux

Tableau 1.1 : Classification des anomalies par niveau de gravité .........................................5

Tableau 1.1 : Caractéristiques principales des solutions de tests existantes ........................ 12

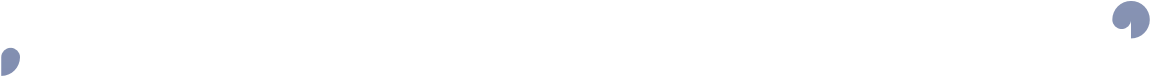
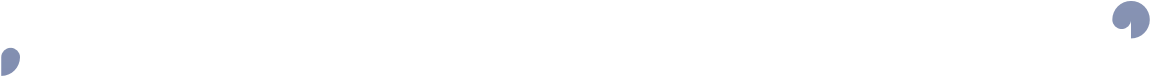
Tableau 2.1: Tableau de répartition des tâches.................................................................. 22



## Résumé

L’environnement actuelle est dominé par l’économie numérique, l’on assiste à un développement exponentiel des équipements et logiciels informatiques, dans ce contexte, les logicielles doivent répondre aux besoins des utilisateurs en garantissant la continuité des activités sans interruptions. Par ailleurs, la principale préoccupation traitée dans ce mémoire de fin d’étude du cycle Master est celle de pouvoir trouver une solution qui donnerait la possibilité de sécuriser les applications SaaS pour qu’elles tendent vers la « sécurité optimale ». Le but de ce travail est de mettre sur pied un processus d’audit à la fois manuel et automatisé permettant d’identifier et réduire au maximum les possibilités d’attaque des logiciels SaaS qui peuvent limiter ou empêcher leur bon fonctionnement. En effet l’on souhaiterait avoir un produit fini exempt de faille ou tout au moins, qui dispose le moins possible, afin de garantir la fiabilité dudit produit et ainsi fidéliser la clientèle. Pour cela, nous avons commencé par étudier le processus d’audit de sécurité des SaaS préalablement utilisé à LAO SARL, ensuite nous avons étudié les d’audits de sécurité existants afin de faire une analyse comparative et d’en ressortir le meilleur, après cette étude, nous avons amélioré la pratique de l’audit de sécurité, grâce à méthode d’audit qui répond mieux au besoin interne de l’entreprise à savoir éviter un accroitre les possibilités de sécurisation de leur SaaS, la génération des rapports de meilleur rapport d’audit, l’ajout de l’outils de capture d’écran de graphe pour une meilleur prise de décision. Ainsi, en s’appuyant sur la méthode « pentesting », nous avons réalisé une un audit qui nous a permis d’aboutir à un niveau optimal de sécurité.

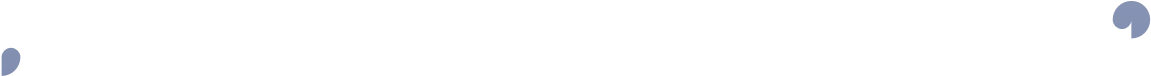
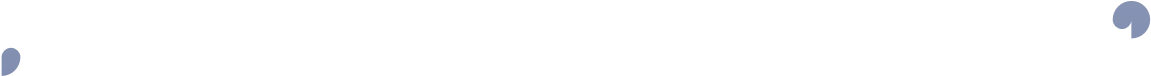
Mots clés : Audit, Pentesting, SaaS, Sécurité



## Abstract

The current environment is determined by the digital economy, assisting in an exponential development of computer equipment and software, in this context, software must meet the needs of users by ensuring business continuity without interruptions. In addition, the main concern dealt with in this master's degree thesis is that of being able to find a solution that would make it possible to secure SaaS applications so that they tend towards “optimal security”. The goal of this work is to set up an audit process that is both manual and automated to identify and minimize the possibilities of using SaaS software that can limit or prevent their proper functioning. In fact, we would like to have a finished product that is free from flaws or at least that has as little as possible, in order to guarantee the reliability of said product and thus build customer loyalty. For this, we started to study the SaaS security audit process used at LAO SARL, then we studied the existing security audits in order to make a comparative analysis and to come out the best, after this study, we have improved the practice of security auditing, thanks to the audit method that better meets the internal need of the company to know to avoid an increase in the possibilities of securing their SaaS, the generation of better reporting reports. audit, adding graph screenshot tool for better decision price. Thus, by relying on the "pentesting" method, we carried out an audit which enabled us to achieve an optimal level of safety.

Keywords: Audit, Pentesting, SaaS, Security



Introduction générale

L’absorption de l’environnement national et international par l’économie numérique bat son plein avec la crise sanitaire du au covid-19. Les logiciels offrant les possibilités de travail étendu (télétravail), tant au niveau national et qu’international, sont très convoitées. A cet égard, les organisations sont en permanence à la recherche de logiciels pouvant au mieux, satisfaire leurs besoins aussi bien pour les petites que les grandes. Le logiciel intervient alors en réponse aux besoins de ces organisations notamment en leur apportant au moins un maintien, sinon une amélioration de leur productivité et du rendement de leurs activités. Et la garantie de ceci relève d’un alignement stratégique avec les organismes, mais aussi de la sécurité de celles-ci. D’où la nécessité des audits de sécurité afin de fiabiliser les produits logiciels en générale et particulièrement de applications SaaS qui se trouve dans jungle de l’internet.

La question principale de ce travail est de savoir comment procéder à des audits de sécurisation des logiciels SaaS. Le pentesting est une méthode d’audit de sécurité, qui va bien au-delà de l’audit conventionnel, en surpassant la prise en compte des dispositions sécuritaires qu’elle propose pour limiter des attaques, pour simuler de véritables attaques afin de prévenir d’éventuelles actions malveillantes. Ceci va offrir une de nombreux avantages et un meilleur rendu sécurité. En effet, les tests d’intrusions renvoient à l’utilisation de scénario programmé ou non pour tester un ou plusieurs aspects de la sécurité d’une application quelle qu’elle soit. Aujourd’hui, les applications SaaS sont fréquemment mises à jour fournissant de nouvelles caractéristiques et fonctions avec une expérience utilisateur améliorée due à une ergonomie de qualité. Cependant, chacun de ces changements, peut faire apparaitre des failles inattendues, c’est pour cela que chaque mise à jour devrait déclencher le test d’intrusion. Dans le cas contraire on prend le risque de lancer une mise à jour qui aboutirait à une ou plusieurs vulnérabilité(s), pouvant offrir des possibilités d’insécurité plus élevées, et donc si exploitées par un tiers, une mauvaise expérience utilisateur et des conséquences catastrophique pour l’entreprise.

En outre, les actions d’un test d’intrusion peuvent se répéter, dans ce cas de figure, un gain d’efficacité est assuré par une automatisation, avec un gain de temps, et une exécution plus rapide, ainsi que des rapports éditer tout aussi automatiquement. Ainsi, à la question principale évoquée ci-dessus, s’ajoute une question secondaire, à savoir comment automatiser le processus de tests d’intrusion logicielle en automatisant les tâches répétitives.

L’objectif principal est de réaliser un audit de sécurité grâce la méthode de penstesting d’une application SaaS avec les outils prévus à cet effet afin de lui assurer une sécurité optimale. Pour atteindre nos objectifs, il sera intéressant de faire un état de l’art et une étude sur la mise en place de la méthode d’audit, rédiger un cahier de charge pour la description du travail à effectuer, l’analyse, l’implémentation et la mise en œuvre du projet et l’évaluation des résultats obtenus.



## Partie I : Etat de l’art



## Chapitre I : Revue de la littérature

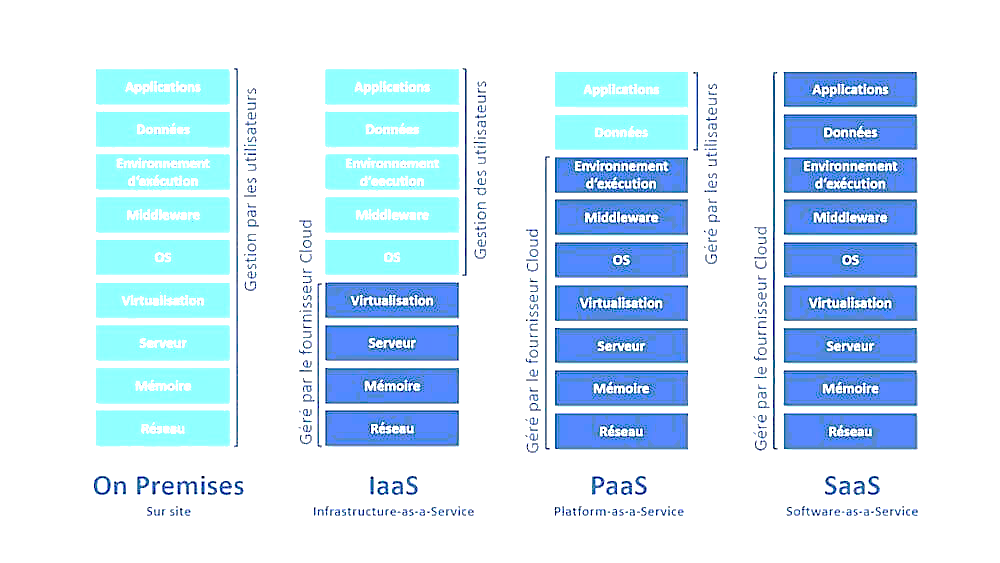
Dans ce chapitre, nous abordons de manière détaillée, les concepts au centre de notre thème, avec entre autres les méthodes d’audit existantes, suivi du pentesting et de son utilité et enfin quelques attaques assez souvent utilisées.

#### Section 1. Concepts et définitions

D’entre la pléthore de problème que peut rencontrer un logiciel SaaS, un problème fondamental est « le problème de la sécurité », en plus de nombreux défauts et anomalies liés aux éléments qui le constituent. Aussi, dans cette partie, nous porterons notre attention sur les concepts (SaaS, audit…) et les failles avec les niveaux de la gravité associes.

1. SaaS

Le cloud computing (*Informatique dans le nuage*) est une technique permettant de gérer des ressources (serveurs, système d’exploitation, des applications…) et d’adapter très rapidement une infrastructure à des variations de charges de manière totalement transparente pour l’administrateur et les utilisateurs. Il a la possibilité de se diversifier de manière typée ainsi qu’il suit : l’IaaS, le PaaS, le S aaS….



Le SaaS est un mode d’utilisation d’une solution logicielle qui se fait en utilisant l’application à distance, elle-même, étant hébergée par l’éditeur. Il se rencontre couramment pour des applications logicielles relatives au CRM ou Web Marketing.

Figure : SaaS

Le Software As A Service (SaaS), ou Logiciel en tant que Service en Français, est **un modèle de distribution de logiciel au sein duquel un fournisseur tiers héberge les applications et les rend disponibles pour ses clients par l’intermédiaire d’internet**. Parmi les principaux fournisseurs d’un logiciel SaaS, on retrouve Salesforce, Oracle, IBM ou encore Microsoft.

La solution logicielle étant utilisée le plus souvent à partir d’un simple navigateur internet, elle permet à l’entreprise d’être dégagée de toutes contraintes d’installation, de mise à jour ou de toutes autres maintenances techniques. Elle permet également d’être utilisée par des collaborateur en situation de mobilité. La mise à disposition d’une solution SaaS peut être facturée par abonnement ou proportionnellement à l’usage. Il peut parfois comporter des frais de personnalisation et de mise à disposition du service. Dans le domaine du web marketing, les plateformes de gestion des campagnes e-mails, les outils d web analytique, et les serveurs publicitaires sont généralement proposés en mode SaaS.

1. **Avantages**

* Grâce à un logiciel SaaS, les entreprises n’ont plus besoin d’installer et de lancer des applications sur leurs propres ordinateurs ou sur leurs Data Centers. Le **coût d’acquisition de matériel est ainsi éliminé, au même titre que les coûts d’approvisionnement et de maintenance, de licence de logiciel, d’installation et de support**. On compte également plusieurs autres avantages ;
* Au lieu d’investir dans un logiciel à installer, et dans un équipement permettant de le prendre en charge, les utilisateurs souscrivent à une offre SaaS. En général, **l’offre se présente sous la forme d’un abonnement mensuel dont le tarif est proportionnel à l’utilisation**. Grâce à cette flexibilité, les entreprises peuvent organiser leur budget avec plus de précision et de facilité. De plus, il est possible de résilier l’abonnement à tout moment pour couper court aux dépenses ;
* Un autre avantage est la haute scalabilité. En fonction de ses besoins, **l’utilisateur peut accéder à plus ou moins de services et à des fonctionnalités à la demande**. Le Logiciel en tant que Service est donc adapté aux besoins propres à chaque business.
* De même, **plutôt que de devoir acheter régulièrement de nouveaux logiciels, les utilisateurs peuvent compter sur le fournisseur SaaS pour effectuer des mises à jour automatiquement et gérer l’ajout de patchs correctifs**. L’entreprise a donc moins besoin d’une équipe d’informaticiens internes.

Enfin, étant donné que les applications SaaS sont délivrées via internet, **les utilisateurs peuvent y accéder depuis n’importe quel appareil connecté et n’importe quelle position géographique**. L’accessibilité est l’un des grands points forts de ce modèle.

Par ailleurs, une**application SaaS peut être utilisée par des milliers, voire des millions d’utilisateurs finaux simultanément puisqu’elle est stockée sur le Cloud**. Par conséquent :

* Plus d’installation ;
* Plus de mise à jour (continuent chez le fournisseur) ;
* Plus de migration de données ;
* Payement à l’usage ;
* Test de nouveaux logiciels avec facilité ;

Même s’il rencontre tout de même quelques limites.

1. **Inconvénients**

* Limitation par définition aux logiciels proposées ;
* Pas de contrôle sur le stockage et la sécurisation des données associées au logiciel ;
* Réactivité des applications web pas toujours idéale ;

En bref, la mise à disposition de telles solutions est sans aucun doute indiquée dans la situation actuelle. Il demeure tout de même important de nous s’intéresser à l’audit sécurité des applications qui y sont hébergées.

2. Application à architecture monolithique / microservice

Aujourd’hui, les entreprises utilisent une multitude d’applications SaaS, 137 en moyenne, d’après le Rapport 2020 sur les tendances SaaS de Blissfully. Ces applications génèrent des téraoctets de données. Souvent, les données présentes sur plusieurs plateformes peuvent être liées, comme l’adresse associée à une carte de crédit utilisée pour un achat en ligne, adresse également exploitée par une plateforme d’expédition, et le suivi de cette transaction en ligne peut être réalisé via la plateforme analytique d’une entreprise.

En matière de déploiement de piles technologiques, les entreprises ont le choix entre deux grands modèles : la mise en œuvre d’une seule plateforme qui combine de nombreuses fonctionnalités ou l’adoption d’une approche « best-of-breed » qui utilise des microsystèmes pour intégrer des services distincts de différents fournisseurs

1. **Architecture monolithique**

Les applications monolithiques ont été conçues pour traiter de multiples tâches connexes. Il s’agit généralement d’applications complexes qui englobent plusieurs fonctions étroitement couplées.

Un monolithe est conçu comme un grand système déployé comme une seule unité derrière un répartiteur de charge. Il dépend généralement d’une base de données unique. Le monolithe se compose de quatre éléments principaux : une interface utilisateur, des logiques métiers, une interface de données et une base de données. Les systèmes monolithiques offrent plusieurs avantages, en particulier en termes de gestion des frais opérationnels. Voici quelques-uns de leurs atouts basiques :

* Simplicité : Les architectures monolithiques sont simples à construire, à tester et à déployer. Les applications qui en dépendent peuvent être mises à l’échelle horizontalement, en exécutant plusieurs copies d’une application derrière un répartiteur de charge.
* Problèmes transversaux : Avec une base de code unique, les applications monolithiques peuvent facilement gérer les problèmes transversaux, tels que l’enregistrement des logs, la configuration et le contrôle des performances.
* Performances : Les composants d’une architecture unifiée partagent la mémoire vive, ce qui est plus rapide que la communication de service à service, dépendante de mécanismes comme l’IPC (Communications inter-processus) ou autres.

Cependant, l’un des principaux inconvénients des architectures monolithes est le couplage étroit. Au fil du temps, les composants deviennent étroitement liés et enchevêtrés. Ce phénomène affecte la gestion, l’évolutivité et le déploiement continu. D’autres désavantages en découlent :

* Fiabilité : Une erreur de programmation dans l’un des modules de l’application peut la faire tomber entièrement.
* Mises à jour : Parce qu’il y a une seule grosse base de code et une dépendance forte entre les composants, toute l’application doit être redéployée à chaque mise à jour.
* Pile technologique : Une application monolithique dépend d’un seul stack technologique. Les modifications de cette dernière s’avèrent coûteuses et longues à effectuer.

Prenons par exemple une application SaaS e-commerce monolithique. Elle peut contenir un serveur web, un équilibreur de charge, un catalogue qui propose des images de produits, un système de commande, une fonction de paiement et une composante d’expédition. Ce qui implique, qu’étant donné leur large portée, les outils monolithiques ont tendance à avoir d’énormes bases de code. Une petite modification à une seule fonction peut obliger à compiler et tester toute la plateforme, ce qui va à l’encontre de l’approche agile aujourd’hui privilégiée par les développeurs.

1. **Architecture microservice**

Contrairement à l’approche monolithique, l’architecture microservice permet de déployer de petites applications indépendantes sous forme de services en couplage, reliées entre elles via l’intégration des applications. De plus, les microservices consistent en des services indépendants des uns des autres. Par essence, ce type d’architecture « divise » les composants en de petits services autonomes qui peuvent être déployés et industrialisés séparément. Avec eux, la logique applicative englobe les multiples plateformes, y compris en mode SaaS, les bases de données et les applications développées en interne destinées à répondre aux besoins non traités par les applications SaaS.

Du point de vue de l’ingénierie logicielle, les microservices peuvent être plus simples à développer. Leur portée est moins large et leur taille est donc plus petite, ce qui permet aux développeurs de les améliorer plus facilement via à une intégration et une livraison en continu (approche CI/CD). Ils peuvent être codés dans n’importe quel langage de programmation. Et ils peuvent communiquer avec d’autres microservices via des API.

Une API (ou interface de programmation d’application) est un ensemble d’appels de programmation qui exposent la fonctionnalité d’une application aux développeurs. Les API simplifient le développement d’applications intégrées en offrant un moyen simple de transmettre des informations d’identification et des données entre les applications.

Et les bénéfices que nous pouvons en dégager sont :

* Passage à l’échelle : Pour industrialiser une application basée sur des microservices, il suffit de passer à l’échelle certains composants, ceux qui optimisent la consommation de ressources.
* Faible couplage : Les composants de microservices ne sont pas interdépendants et peuvent donc être testés individuellement. Cela facilite également les modifications à travers le temps.

Le passage aux microservices demande un partage équitable des coûts. Cela réclame une attention toute particulière en ce qui concerne la surveillance applicative et le fardeau qu’elle représente pour les développeurs. Les entreprises qui adoptent ce type d’architecture devront prendre en compte les facteurs suivants :

* L’expertise de l’équipe : les bénéfices des microservices s’effacent si le personnel n’est pas préparé. Les compétences des membres de l’équipe doivent être évaluées avant d’adopter pleinement ce modèle.
* Tests et surveillance : Une fois les applications devenues des composants, il y aura d’avantage d’éléments mobiles à suivre et à réparer. Sans les bons outils, les choses peuvent rapidement déraper.

Il est à noter qu’il n’y a véritablement de meilleur ente ces deux architectures, tout va dépendre des besoins de chaque organisation. Les entreprises doivent prendre en compte différents critères :

* La facilité de mise en œuvre : On pourrait croire que les systèmes monolithiques sont plus faciles à mettre en œuvre, puisque le logiciel est fourni par un seul vendeur. Ce n’est pas toujours le cas. Les systèmes monolithiques étant généralement complexes, leur déploiement peut être aussi difficile que celui de multiples plateformes individuelles. Le point fort des systèmes monolithiques est leur guichet unique pour l’assistance. Mais c’est un point fort uniquement si le vendeur est réputé pour son assistance de qualité.
* Verrouillage propriétaire : En règle générale, les systèmes monolithiques essaient de couvrir un large éventail de fonctions connexes. Par exemple, une plateforme d’hébergement web monolithique ne comprend pas seulement un serveur web qui gère les requêtes HTTP côté serveur, mais aussi des pare-feux, un équilibreur de charge et un réseau de distribution de contenus. Mais, ayant été conçus pour « tout faire », les systèmes monolithiques ont habituellement du mal à fonctionner avec d’autres systèmes. Ce qui nous amène au point suivant…
* Le contrôle et la propriété de leurs données : Les systèmes monolithiques ne permettent pas aux organisations d’intégrer facilement les données de leurs systèmes. Habituellement, il n’est possible utiliser les données qu’au sein du monolithe. Par exemple, un système d’analyse monolithique qui comprend l’intégration de données, les pipelines de données ETL, un datawarehouse et un logiciel analytique peut ne pas fournir d’outils permettant aux entreprises d’accéder à leurs propres données pour les intégrer à d’autres systèmes, ni réaliser des analyses avec d’autres logiciels.
* Retour sur investissement (RSI) : Il est inutile de déployer une application si son RSI n’est pas positif. Pour les entreprises qui développe leurs propres applications ou déploient des solutions SaaS, les ingénieurs logiciels peuvent créer des micro-services relativement rapidement, les déployer dès qu’ils sont prêts et les mettre à disposition des clients (externes ou internes, selon l’application). Au fur et à mesure qu’ils déploient des services, les délais de mise sur le marché diminuent et le ROI devient positif.

Globalement, le marché semble être en train de passer du monolithique au microservice car il est difficile pour les entreprises de combiner toutes les fonctions nécessaires et souhaitées, complètement en phase avec leurs processus en place, sur une même plateforme. La plupart des entreprises obtiennent une meilleure expérience globale en déployant la solution la plus adaptée à des besoins spécifiques, et en les reliant via l’intégration des applications. Cependant, dans le cadre de l’internet et de son caractère souvent hostile, il apparait judicieux de s’interroger sur la sécurité de SaaS qui ne dépend tout aussi bien de ce que nous venons de voir.

3. Sécurisation d’une application SaaS

De manière globale, la cybernétique renvoie l’étude des systèmes de commande et de communication entre êtres vivants, machines et autres. Et au sein de cet ensemble complexe, apparait la notion de sécurité, que nous pouvons simplement considérer comme un état d’esprit confiant et tranquille d’une personne qui se croit ou se sent à l’abri du danger ; ce qui signifie qu’elle dépend de la personne qui ressent cela, du contexte dans lequel elle se trouve, du type de menaces qu’il peut avoir, et bien d’autre encore. Autrement dit, la sécurité dépend de tout un écosystème et donc il n’y a pas véritablement de sécurité, encore moins de cybersécurité, ce qui complique assez le problème.

Puis nous avons les produits, que les entreprises réalisent pour commercialiser, il peut s’agir de :

* Produit embarque,
* Sous-système,
* Solution,
* Logiciel ou suite de logiciel.

L’idée étant de les fournir à l’utilisateur, pour une utilisation dans un contexte propre à lui. Donc l’entreprise réalise un produit, pensant comprendre de quelle manière le client va s’en servir, d’autant plus que c’est son domaine. Ce qui est vrai dans un contexte fonctionnel, dans la mesure où le produit répond à un besoin du client. Aussi dans le domaine de la cybersécurité, c’est le contexte d’utilisation qui dimensionne le type de risque et d’exposition dont le client peut être victime. Faisant ainsi de la sécurisation des produit un domaine à prendre en compte.

Cependant, nous devons définir ce qu’il y a à protéger. Et en matière de cybersécurité, trois éléments clés sont à retenir :

* La confidentialité : les données ne doivent pas être divulguées ;
* L’intégrité : les données ne doivent pas être falsifiables ou fausses ;
* La disponibilité : l’accès aux données doit être contrôlée.

A ce niveau, la sécurisation se restreint à la fonctionnalité d’un produit. A ceux-là, nous pouvons ajouter les notions de :

* Traçabilité : pour répertorier les transactions d’un système
* Non-répudiation : pour éviter de nier tout participation à une transaction

Et même avec ceux-ci, comme dans le contexte actuel, les choses évoluent de manière exponentielle, dans le sens de rendre rapidement des moyens de sécurité mis en place obsolètes, la validité d’un produit en matière de sécurité, ne tient que le temps de se faire dépasser, ce qui arrive assez rapidement dans bien des cas.

Autrement un produit quel qu’il soit, finira bien par être compromis, s’il n’est pas amélioré de manière permanente. Et donc un jour tout produit SaaS finit par être compromis. Cependant il ne faut pas s’arrêter là, mais pouvoir retrouver ce qui s’est passé, de remonter à la cause, de circonscrire les effets et évidemment d’analyser et de remonter toutes les informations nécessaire la compréhension de ce qui s’est passé. Ceci afin d’améliorer les processus afin que la compromission ne se reproduise pas. Mais aussi qu’en cas d’attaque, de compromission et de dégâts occasionnes, que le produit SaaS permette de remonter les information (accès, nature des dégâts, éléments modifiées) y afférant (grâce aux logs par exemple). Ce qui permettra aux assurances de louer, aux clients de porter plainte au niveau de la justice, et en gros d’avoir tout un écosystème d’éléments permettant de minimiser la portée des effets de la compromission d’un produit SaaS. Aussi, nous avons également le cas des logs, qui dans leur fonctionnement normal, permettent de détecter les comportements qui sont également anormaux ou suspects. Et avec la non-répudiation ci-haut citée, pour se rassurer que pour un élément reçu, son émetteur est bien identifiable et celui-ci ne peut pas nier avoir réalisé cette action.

Nous avons par exemple les ransomwares, qui dans leurs actions touches à peu près tous les piliers principaux ci-dessus citées, à l’instar de l’intégrité car ils vont chiffrer toutes les données, la disponibilité car les données sont désormais rendues indisponibles, et la confidentialité car aujourd’hui les organismes attaqués par ceux-ci se retrouvent menacées de divulgations de leurs données si des conditions ne sont pas respectées. Tout ceci faisant paraître le fait qu’une attaque ne ciblera pas toujours qu’un mais la totalité des principaux piliers de la sécurité.

Pour la sécurisation, la plupart des organisations opterait pour la mise en place d’un ensemble de solutions techniques. Ce qui n’est pas une panacée pour nos problèmes de sécurité applicative, aussi nous opterons pour un processus échelonné comme suit :

* Identification des actifs: Il n’est pas juste question de ce que l’organisme veut protéger bien qu’important, mais aussi et surtout de ce que le client veut protéger ;
* Identifications des cadres réglementaires et normatifs : Il est question de limiter l’impact qui leur est associée, ce qui ajoute des contraintes ou empêche de faire un certain nombre de choses ;
* Identifier toutes les parties prenantes : car même si l’organisme est constitué de plusieurs équipes participant à la réalisation d’un produit SaaS, au même titre que la plateforme d’hébergement, les clients qui ont, eux, probablement des clients, l’état et autres utilisateurs qui sont tous à prendre en compte pour trouver la meilleure approche sécuritaire ;
* L’analyse de risque de cybersécurité (bonne pratique) : elle peut être légère, très lourdes, mais elle permet d’avoir une démarche structurée pour maîtriser le niveau et les actions de sécurisation. Par ailleurs, il n’y en a encore, des sociétés qui investissent énormément dans les risques, les analyses de risque, et les déclinaisons techniques, ceci tant, comme elles sont relativement ignorantes du contexte, elles ne savent pas comment s’arrêter, les exigences ne sont pas forcément suffisantes, le contexte est extrêmement important. Aussi nous avons :
  + La définition des architectures : pour la définition d’une architecture sécurisée, il faut savoir ce qu’on fait, c’est à dire : qu’est-ce qu’on fait? Comment est que le produit SaaS est segmenté? Dans quoi il s’inscrit? Comment faire pour qu’à partir de la manière dont les choses sont organisées, que la sécurité soit plus facile à assurer? Exemple, dans la mesure ou des niveaux d’accès sont définis dans l’utilisation d’un produit SaaS et que pour chacun d’eux le même mot de passe est utilisé, est-ce que celui-ci sera sauvegardée?
  + Les solutions techniques : Ceux-ci doivent être déterminer sur la base de tout ce qui précède.

Nous pouvons ainsi en déduire les grands thèmes d’une sécurisation réussie a l’instar de :

* + Analyse des risques et définition des exigences de sécurité ;
  + Définition des conditions d’utilisation du produit SaaS ;
  + Définition de l’architecture interne ;
  + Implémentation des solutions techniques / défense en profondeur ;
  + Protection de l’environnement (de développement, de production, de maintenance) ;
  + Gestion de fin de vie.

Cependant, il n’y a pas de sécurité absolue. De ce fait, la sécurité des applications doit être vérifier en permanence, afin de s’assurer de la fiabilité de celles-ci. D’où les audits de sécurité que nous présentons dans la suite.

#### Section 2. Audit sécurité d’une application SaaS

L'audit de sécurité d'un système d'information (SI) est une vue à un instant T de tout ou partie du SI, permettant de comparer l'état du SI à un référentiel. Il répertorie les points forts, et surtout les points faibles (vulnérabilités) de tout ou partie du système. Aussi, un auditeur devra dresser également une série de recommandations pour supprimer les vulnérabilités découvertes.

Une application SaaS se veut protégée. Cependant tous les organismes n'ont pas besoin du même niveau de sécurité. Un contrat doit donc mentionner l’étendu des tests. Il se peut, par exemple, que l'entreprise ne soit pas intéressée par un audit utilisant le Social engineering. De même, certaines phases comme la collecte d'informations dépendent directement du type d'audit réalisé, ce qui nous amené à la procédure d’un audit de sécurité.

1. Procédure de l’audit de sécurité

Cette partie peut se découpe en plusieurs étapes, à l’instar de :

* La prise de connaissance de manière extrêmement fine les attentes du client. Il convient de bien comprendre ses besoins et de les reformuler. Cette première étape est particulièrement importante dans la mesure où elle plante le contexte précis dans lequel l’audit va être mené : autant d’informations qui seront incluses dans le rapport d’audit afin d’en faciliter l’interprétation, même plusieurs années après sa réalisation.
* La lettre de mission, pour la procédure à venir avec deux objectifs principaux :
* Elle est le contrat qui lie l’entreprise et l’auditeur ;
* Elle permet d’informer les différentes personnes impliquées de l’arrivée d’un audit dans l’entreprise, et auprès des salariés avec une légitimation de cet audit par la direction.
* Le recueil de toutes les informations nécessaires pour préparer la mission. Il s’agit de récolter les éléments relatifs à la culture de l’entreprise, au contexte général toujours en corrélation avec le système d’information. Des rendez-nous sont donc organisés avec les personnes concernées.
* La réalisation de la mission, l'audition du système d'information de l'entreprise peu commencé.
* Une réunion de synthèse organisée entre l’auditeur et les personnes intéressées. Il s’agit de s’assurer ensemble que :
* Les questions de l’auditeur ont été bien comprises ;
* Les réponses ont été bien interprétées.

Le rapport est ensuite rédigé, de plusieurs manières (concis et plus complet), car il s’adresse en général à plusieurs types de publics. Celui-ci détaillé, expliquera les attentes de départ, le contexte, les limites, les faiblesses constatées, leur importance relative et les solutions. Il doit être clair et didactique. En aucun cas il ne doit être technique. Cependant il existe plusieurs techniques d’audit comme nous le verrons dans ce qui suit.

1. Les techniques d'audit de sécurité

Dans l’implémentation d’un audit de sécurité, plusieurs modes se distinguent. Aussi nous avons les audits dit en mode « boite blanche », « boite grise » et « boite noire ».

**a. Audit en « boite blanche »**

Ce type de test nécessite des connaissances importantes sur la structure interne et l’implantation des systèmes à tester. En plus des spécifications fonctionnelles et d’une description à haut niveau du système, on doit également connaître l’architecture complète, les divers schémas, le code source et toute information nécessaire à la compréhension du système. Le test complet permet de déterminer, avec certitude, si les mécanismes de sécurité en place sont fonctionnels ainsi que de s'assurer qu’aucune erreur évidente n’est présente, que la mesure de sécurité s’exécute comme il se doit, sur une base continue et cohérente, et qu’il existe un soutien pour l’amélioration continue de l’efficacité de la mesure.

**b. Audit en « boite noire »**

Ce type d’audit ne nécessite aucune connaissance sur la structure interne et l’implantation des systèmes à tester. Une description du système à haut niveau ainsi qu’une connaissance de base des spécifications fonctionnelles sont suffisantes pour ce type de test. Le test de base permet de déterminer si les mécanismes de sécurité en place sont fonctionnels et de s'assurer qu’aucune erreur évidente n’est présente.

**c. Audit en « boite grise »**

L’accès de base représente un accès à un système, avec une identité, mais avec peu de privilèges. Exemple, la rigueur des sessions ou la confidentialité de l'information d’un membre connecté pourraient être testées avec ce niveau d’accès.

Et sur la base de ces différents modes, nous pouvons présenter de manière comparative les différents types d’audit dans le tableau ci-après.

*Tableau 1 .1: Tableau comparatif des types d’audit*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Audit site Web**  **Sécurité site Internet** | **Test d'Intrusion Simulation d'attaque pirate** | **Audit de code applicatif et analyse d'applications** | **Audit Sécurité d'Infrastructures et Systèmes** |
| **Objectif principal** | Recherche, identification et analyse et correction des failles et vulnérabilités latentes sur les sites Web Internet. | Détection ex exploitation de vulnérabilités. Tentative d'intrusion dans le SI. Déterminer les axes d'attaques existants sur l'organisation. | Tester la sécurité d'une application. Recherche de bug / failles compromettant la sécurité. | Identifier les failles et vulnérabilités dans le SI au sein de l'organisation. Validation technique (paramétrage) des éléments du SI. |
| **Résultats attendus** | Liste priorisée des vulnérabilités identifiées. Application de correctifs et contre-mesures. Augmentation du niveau de sécurité Web. | Identifier les points de fragilité du SI. Impact réel d'une attaque sur le SI de l'organisation. | Validation de l'étanchéité d'une application à des attaques possibles. | Optimisation du niveau de sécurité interne à l'entreprise (sécurise des postes, serveurs, annuaires d'entreprise, configuration réseaux, PABX, etc.) |
| **Déroulement de l’intervention** | Expertise effectuée à distance | A distance : BlackBox (test d'intrusion en aveugle)  Sur site : Greybox (périmètre), Whitebox (cible précise dans le SI) | A distance (code source, utilisation de VMs) ou sur site in vivo. | Audit technique réalisé sur site |
| **Exemple de resultat** | Audit sécurité de Site Web d'entreprise. Sécurisation d'Extranet ou site eCommerce marchand. Vérification sécurité avant mise en production d'un nouveau site. | Identifier les failles du SI de l'organisation. Test d'intrusion sur la sécurité physique des réseaux WiFi. Valider les comportements via l'ingénierie sociale. | Application métier sensible, application tierce en interaction avec le SI, validation application avant mise en production | Audit d'un annuaire d'entreprise. Identifier le vol d'information possible par un identifiant utilisateur. Corriger les failles du VoIP. Sécurisation du réseau interne à l'entreprise. |

A ce niveau nous pouvons sans conteste admettre que pour la méthode d’audit qui épouse le mieux notre projet est le test d’intrusion ou pentesting, que nous présenterons en détail dans la prochaine section.

## Chapitre 2 : Pentesting

#### Section 1. Généralités

* 1. Présentation

Le pestesting, ou test d'intrusion, ou test de pénétration, ou simplement pentest, est une méthode qui consiste à analyser une cible (une adresse IP, une application, un serveur web, un réseau complet…) en se mettant dans la peau d'un attaquant (connu aussi sous le nom d’hacker malveillant ou pirate, mais nous y reviendrons) [2]. C’est donc une représentation ponctuelle d’une cible., et ne doit donc pas être confondu avec le scan de vulnérabilités.

Car dans le cadre de la sécurité des systèmes d’information, un pentest consiste en la soumission d’un ou plusieurs systèmes à différents contextes d’attaque, en afin d’évaluer la réaction du système et découvrir présence effective ou non de vulnérabilités. Aussi ceux-ci diffèrent bien des tests de vulnérabilités de par leur objectif. En effet, un test de vulnérabilités vise à déceler des vulnérabilités connues sur des systèmes, alors qu’un test d’intrusions a pour but de simuler une attaque, par la découverte et l’exploitation de vulnérabilités. Autrement dit, un test d’intrusions, en plus d’identifier des vulnérabilités, exploite celles-ci dans le but de vérifier les impacts réels des intrusions, permettant ainsi éliminer des faux positifs.

Notons qu’un organisme est d’autant plus sécurise que son système le plus faible l’est. En effet, les hackers (pirates informatiques) utilisent bien souvent les vecteurs d’attaques les plus vulnérables pour réaliser leurs objectifs. Il apparaît donc important de prendre en considération cette règle pour tout ce qui suivra afin de garantir aux tests d’intrusions une réelle valeur ajoutée à l’organisme une nette amélioration de son niveau de sécurité.

**a.** **Historique**

Le besoin en matière de protection l’information, a toujours été une préoccupation importante pour les organismes, et ce qu’importe leur nature (spublique ou privés). Cependant, l’année 2000 a particulièrement marqué le domaine des tests d’intrusions. En effet, en décembre de l’année 2000, une première méthodologie de tests d’intrusions a été publiée par l’Open Source Security Testing Methodology Manual (OSSTMM). Les techniques utilisées pour effectuer les tests de cette nature devenaient ainsi mieux encadrées et plus structurées. C’est à partir de l’année 2004 que le crime informatique s’est orienté davantage vers le développement de code malicieux. Les hackers se sont mis à s'attaquer massivement les postes de travail et aux appareils mobiles. Le monde de la sécurité de l’information est alors entré dans l’ère des réseaux de zombies (botnets).

Aujourd’hui encore, les attaques sont de plus en plus ciblées, c’est-à-dire que le code malicieux développé est personnalisé en fonction de la cible. Les tendances et les techniques utilisées par des attaquants ont grandement évolué et ont influencé, par le fait même, les divers types de tests d’intrusions.

**b.** **Les sources de motivations**

Une bonne protection les systèmes d’'information constitue une source de motivation pertinente des organismes publics pour effectuer des tests d’intrusions. De manière précise, ces tests visent à :

* Répondre à une obligation organisationnelle, comme c’est le cas lorsque le dirigeant d’un organisme doit s’assurer du maintien du niveau de sécurité annuellement ou à la suite d’un changement majeur susceptible d’avoir des conséquences sur la sécurité du système d’information, et en dégager les priorités d’actions et les échéanciers afférents ;
* Evaluer la sécurité de l’infrastructure technologique (postes, serveurs, réseaux, etc.) ;
* Valider la sécurité d’un système avant sa mise en production ou pendant son développement (codification, paramétrage, etc…).

2. Utilité du pentesting

L’étendu des cyberattaques ne cesse de croitre depuis plusieurs années, et les motivations sont d’ordres diverses (économique, politique, ou le challenge). Même si la grande majorité du temps, elle est économique. Plusieurs groupes de hackers existent, et parmi eux des plus ou moins malveillants de par le monde.

La sécurisation des systèmes a souvent été cher et les chances de se faire pirater étaient minces. Cependant la donne a changé depuis car les possibilités piratage sont élevées, surtout dans un contexte ou la maitrise de la sécurité peine à décoller. Il faut donc absolument que chaque entreprise mette en place une politique de sécurisation adaptée, surtout dans le cas de la production d’applications comme les SaaS.

3. Objectifs du pentesting

A ce stade, nous pouvons aisément présenter les objectifs d’un test d’intrusion. Aussi nous avons :

* Identifier les vulnérabilités de son SI ou de son application ;
* Évaluer le degré de risque de chaque faille identifiée ;
* Proposer des correctifs de manière priorisée ;

Ainsi grâce au test d'intrusion, nous pouvons qualifier la sévérité de la vulnérabilité, la complexité de la correction, et l'ordre de priorité qu’il faut donner aux corrections. Aussi, le but n 'est pas malsain, mais il question est de s'assurer que ces vulnérabilités sont bien réelles (éviter de faux positifs). Un exemple probant, est le cas de ces entreprises ou nous se trouvent encore des vulnérabilités dans leurs serveurs, qui permettraient aux rançongiciels (un rançongiciel (ransonmware en anglais), est un logiciel malveillant qui prend en « otage » vos données personnelles et qui les restitue uniquement contre le versement d'une « rançon ») de se propager [2].

Un test d’intrusion permet d’en prendre conscience et de prioriser les corrections. Sinon la vulnérabilité risque de rester dans le système jusqu'à son exploitation malveillante. Une récente étude a établi que dans le monde, une entreprise se fait attaquer, en moyenne, toutes les 40 secondes par un rançongiciel [2].

En outre, Il est à noter qu’afin de sécuriser une application, les tests d’intrusion peuvent être faits à différents moments :

* Lors de la conception du projet, afin d’anticiper les éventuelles attaques ;
* Pendant la phase d’utilisation, à intervalle régulier
* Suite à une cyberattaque pour ne pas que ça se reproduise

Néanmoins, sa conduite demeure est à l’initiative de l’entreprise qui veut se tester. Et bien entendu comme nous l’avons précisé plut-haut (*cf Section 2- : 2- : c- Tableau*), il peut se faire aussi bien de l'extérieur (boite noire) avec n'importe quelle connexion Internet, de l'intérieur de l’infrastructure (boite blanche), sur le LAN (réseau interne de l'entreprise), qu’avec un peu des deux premiers (boite grise). Ceci par un pentesteur, qui est en charge de réaliser le test d’intrusion.

En bref, maintenant que nous avons défini un test d’intrusion et son utilité, nous verrons dans la prochaine section les différents types de vulnérabilité d’un SaaS. Nous pouvons cependant, admettre qu’un test d'intrusion qui intervient dans la vérification et l’identification des types de vulnérabilités ou d’attaques d'un SI et le cas échéant d’un SaaS (infrastructure, application…). Comme audit de sécurité, il est rapide et efficace. Il donne également des résultats plus pertinents.

4. Choix du type pentesting

Pour déterminer le test à effectuer, le choix sera fonction de ce que l’entreprise veut tester. Ce dernier permet de définir la stratégie dont dépendent les résultats. La cible devra dont être minutieusement circonscrite et le type d’attaques dont l’entreprise souhaite se prémunir. Aussi, il faut déterminer le meilleur type de scan et les meilleures conditions de connaissance du pentesteur avant de débuter le un test.

En somme, nous avons dénombré trois types de tests d'intrusion (interne, externe et un peu des deux), D’où se déduisent trois contextes plus Il y a trois conditions de test d'intrusion : boîte blanche, boîte grise, et boîte noire. Ainsi, en fonction de la situation de départ, un pentesteur peut avoir un niveau d'information qui lui permet d’approfondir ou non un test. Il ne nous reste plus qu’à effectuer un tour d’horizon sur les outils de pentest.

#### Section 2. Attaques utilisees pour le pentesting

Après un tour panoramique des outils du pentesting, place aux attaques le plus souvent utilisées lors d’un pentesting.

Dans le cadre de la cybersécurité, une attaque est un acte prémédite, visant une entité (personne, organisme, logiciel…) précise, avec pour objectif la mise à mal de celle-ci. Aussi en présentons-nous quelques-unes.

1. déni de services

Le déni de service (DoS : Denial of Service, en anglais), est une attaque qui a pour but de rendre indisponible un service. Ici, les utilisateurs légitimes ne peuvent plus accéder à une ressource en ligne, et elle concerne : les serveur Web, de messagerie, base de données, ou toutes ressources accessibles en ligne. De plus, son exécution est d’une grande simplicité.

Le principe est uniquement de saturer la connexion cible pour qu'elle ne puisse plus répondre aux requêtes des utilisateurs. Cette attaque peut se présenter sous plusieurs formes :

* La perturbation des connexions entre deux postes, pour empêcher l'accès à un service ;
* La saturation du réseau bloquer son fonctionnement ;
* L'envoyer de plusieurs milliers d'octets à une box internet ;
* L'obstruction d'accès à un service pour une personne en particulier.

2. Attaques web

Elles visent les serveurs Web en générale, avec plusieurs techniques pour outrepasser les protections de ceux-ci à l’instar de ceux dans le tableau qui suit.

*Tableau : Attaques web*

|  |  |
| --- | --- |
| **Attaques connues** | **Description** |
| Faille connue | L’outils Nessus (scanner de vulnérabilités) peut trouver des failles qui seront exploitées avec un logiciel comme Metasploit. |
| MITM | Man-in-the-middle attack ou technique de l'homme du milieu, consiste à se mettre entre un client et un serveur pour intercepter et modifier les paquets échangés entre eux. Ce qui se fait de « Burp suite ». |
| Forger des paquets HTTP | Avec l’écriture manuelle et l’envoi de requêtes HTTP malveillantes au serveur. Le logiciel Wfetch intégré à Microsoft IIS permet de le faire. |
| L'intégration de paramètres | Avec l’ajout ou la modification de paramètres dans les chemins http, pour une fausse interprétation du serveur et par la même son exploitation (pas besoin d’outils). |

Cependant, il en existe bien d'autres attaques Web, mais celles-ci sont les 5 techniques principales.

3. Injection SQL

Selon Wikipédia, une injection SQL est un groupe de méthodes d'exploitation de faille de sécurité d'une application interagissant avec une base de données. Elle permet d'injecter dans la requête SQL en cours un morceau de requête non prévu par le système et pouvant en compromettre la sécurité.

Elle épouse le contexte que la précédente et est basée sur les vulnérabilités des serveurs SQL. En outre, elles s’exécutent par injection de caractères afin de les exploiter, soit en directement, soit via le site ou l’application Web. Sous Kali Linux, plusieurs applications en permettent la réalisation comme BSQL. Quoiqu’elle puisse se faire également manuellement, c’est une activité qui peut devenir vite devenir complexe. Aussi, il en existe plusieurs :

* La méthode blind based, qui permet de détourner une requête SQL en cours sur le système et d'injecter des morceaux qui vont retourner caractère par caractère ce que l'attaquant cherche à extraire de la base de données ;
* La méthode error based, qui permet de détourner une requête SQL en cours sur le système et d'injecter des morceaux qui vont retourner champ par champ ce que l'on cherche à extraire de la base de données ;
* La méthode union based, qui permet de détourner la requête SQL en cours sur le système et d'injecter des morceaux qui vont retourner un ensemble de données directement extraites de la base de données ;
* La méthode Stacked queries, certainement une des plus dangereuses. Elle profite d'une erreur de configuration du serveur de base de données, cette méthode permet d'exécuter n'importe quelle requête SQL sur le système ciblé, ce qui ne se limite pas seulement à récupérer des données comme les 3 précédentes.

4. Les vecteurs d’attaques dans le pentesting

Les vecteurs d’attaques constituent les points d’entrée qu’un attaquant (hacker) pourrait utiliser pour s’introduire dans les systèmes d’une organisation. Les systèmes informatiques, les bâtiments et même les humains sont des vecteurs d’attaques qui peuvent être utilisés afin de compromettre un système.

Nous en dénombrons plusieurs, ce qui justifie pour les systèmes à tester, de les sélectionner en fonction de la criticité et du niveau de confort, ainsi que des mesures de sécurité qui les protègent. La multiplicité des vecteurs d’attaques offre la possibilité de rehausser la sécurité d’un grand nombre de systèmes, en couvrant par la même la sécurité de l’information qu’un organisme détient dans l’exercice de ses fonctions. Le tableau 1 dresse une liste de vecteurs d’attaques communément rencontrés.

*Tableau : Vecteurs d’attaques*

|  |  |
| --- | --- |
| **VECTEUR** | **EXEMPLE** |
| Infrastructure interne / externe l’organisme | Systèmes soutenant les applications, notamment les pares-feux, les routeurs, pour les services accessibles de l’extérieur du périmètre de sécurité |
| Application | Produits applicatifs de type client lourd, Web, mobiles, services Web, qu’ils soient conçus maison ou achetés (logiciels et progiciels). |
| Solution de sécurité | Systèmes ayant pour objectif de protéger les infrastructures ou les applications. |
| Bâtiment  (sécurité physique) | Cartes d’accès, jetons, processus d’accès au bâtiment, processus de demande d’accès, absence de gardiens de sécurité pour surveiller l’accès au bâtiment, etc. |
| Humain | Comprend particulièrement l’ingénierie sociale. Par exemple, une personne malveillante pourrait abuser de la confiance d’un employé ou d’un dirigeant. |

#### Section 3. Les strategies de pentesting

De ce qui précède, pour chacun des vecteurs identifiés, il existe plusieurs stratégies et méthodologies pour effectuer un test d’intrusions. Aussi, les différentes caractéristiques qui peuvent influencer la nature d’un test sont :

* L’information disponible ;
* L’accès utilisateur ;
* L’accès réseau.

1. L’information disponible

Il est possible pour une organisation de réaliser différents types de tests fonction de l’information qu’elle souhaite rendre disponible afin d’en de couvrir les vulnérabilités. Dans le tableau 2, nous décrivons les principaux types de tests proposes par des prestataires de tests d’intrusion, ainsi que le niveau d’information disponible associé.

Ici, la stratégie de test peut être déterminée, tout d’abord, en fonction des motivations de l’organisation et de l’étendu des tests à réaliser. Ainsi, pour une vue globale de la sécurité d’un système d’information, un test de base suffira, alors que pour un extranet ou un changement majeur apporté à une architecture, un test concentre ou complet sera mieux indique.

*Tableau : Type de tests en fonction du niveau d’information (inspiré du NIST)*

|  |  |
| --- | --- |
| TYPE | DESCRIPTION |
| Test de base  (Boîte noire) | Ce type de test ne nécessite aucune connaissance sur la structure interne et l’implantation des systèmes à tester. |
| Test concentré  (Boîte grise) | L’accès de base représente un accès à un système, avec une identité, mais avec peu de privilèges. |
| Test complet  (Boîte blanche) | Ce type de test nécessite des connaissances importantes sur la structure interne et l’implantation des systèmes à tester. |

**a. L’accès utilisateur**

Pour cette caractéristique, nous pouvons avoir un organisme voulant simuler différents niveaux d’accès utilisateur dans sa stratégie de tests. Pour cela, nous pouvons personnifier un utilisateur anonyme, un client accédant à son information ou encore un administrateur du système pour le test du système exposé sur Internet. Nous décrivons dans le tableau (3) trois exemples de niveaux d’accès utilisateur pouvant être retenus dans le cadre d’un test d’intrusion.

Pour définir la stratégie d’un test, la sélection du niveau d’accès utilisateur peut également s’effectue en fonction de la motivation de l’organisation, et tout particulièrement en fonction de la criticité du système à tester. Nous avons par exemple le cas des situations où un organisme suspecte des comportements inhabituels de la part des employés ou des clients, au moyen d'un test, il pourrait, il peut alors vérifier si les mécanismes de contrôle d’accès en place sont suffisamment restrictifs.

*Tableau : Niveau d’accès utilisateur*

|  |  |
| --- | --- |
| **ACCES** | **DESCRIPTION** |
| Anonyme | L’accès anonyme représente un accès à un système, sans identité ou privilèges. |
| Accès  de base | L’accès de base représente un accès à un système, avec une identité, mais avec peu de privilèges. |
| Accès privilégié | L’accès privilégié représente un accès à un système, avec une identité et des privilèges élevés. |

#### 

**b. niveau d’accès réseau**

Selon le niveau d’accès réseau, un organisme peut également vouloir simuler différentes attaques de sa clientèle. Nous récapitulons dans le tableau (4) les niveaux d’accès réseau que peut retenir un organisme dans le cadre d’un test d’intrusion.

Pour le choix du niveau d’accès réseau, il faudra tenir compte des motivations de l’organisation et surtout de l’accessibilité des systèmes à tester. Aussi lors d’un test réaliste simulant une attaque depuis d’internet, un accès à partir de l’extérieur du périmètre de sécurité est sollicité. Aujourd’hui par-contre, les attaques sont le plus souvent initialisées de l’intérieur. Ce qui justifiera, lors d’un changement organisationnel majeur ou fonction de l’exposition aux risques, qu’un test effectué à partir de l’intérieur d’un organisme soit également nécessaire.

*Tableau : Niveau d’accès réseau*

|  |  |
| --- | --- |
| **ACCES** | **DESCRIPTION** |
| Externe | Pour effectuer les tests à l’extérieur du périmètre de sécurité, une ou plusieurs adresses IP (Internet Protocol) publiques sont prédéterminées. |
| Interne | Pour effectuer les tests à l’intérieur du périmètre de sécurité, un organisme public teste les différents systèmes à partir de son réseau interne. |

s

Plusieurs méthodologies nourrissent la réalisation des tests d’intrusions. Les plus utilisées d’entre elles sont sommairement présentées en annexe. De manière globale, nous pouvons regrouper les étapes d’un test en en trois grandes sections, comme illustré dans la figure (1). On y retrouve les étapes de planification, de découverte, d'exploitation et de rapport qui couvrent l’ensemble des éléments à considérer lors de tests d’intrusions



Figure 1 – Étapes de mise en œuvre d’un test d’intrusions

**a.** **Planification**

L’importance de cette étape n’est pas de moindre importance et les éléments qu’elle doit couvrir le sont encore moins. Aussi, avec une bonne planification la détermination des objectifs est garantie et aisance de la réalisation des étapes conséquente. Ainsi pour qu’un organisme qui définisse clairement les objectifs des tests à réaliser, il peut s’inspirer de ses motivations et des vecteurs d’attaques présentés au point 2.3 du chapitre 2. En outre, c’est à cette étape qu’est établie la stratégie du test. Elle intègre également les caractéristiques des tests d’intrusions au point 2.4 du chapitre 2. Par exemple, pour une vue générale de la situation de la sécurité d’un segment de réseau, un test de base pourrait être suffisant. Cependant, pour valider la robustesse des mécanismes de sécurité d’un système critique pour la mission d’une organisation, un test complet reste le meilleur choix.

Par ailleurs, les efforts à consentir sont également influences par les objectifs et les coûts à engager. Pour reprendre ce qui précède, un mécanisme de sécurité peut être testé en 2 jours-personnes ou en 8 jours- personnes, selon le niveau de profondeur recherché et la complexité du système.

**a.1.** **Identification des intervenants**

Lorsque l’entreprise fait appel à un prestataire de services, nous avons un chargé de projet, qui effectue la gestion de l’intervention, mais également une équipe technique de réalisation, spécialisée dans le domaine.

*Tableau : Exemple de rôle des intervenants d’un organisme*

|  |  |
| --- | --- |
| **ROLES** | **IMPLICATION** |
| * Dirigeant d’un organisme public * Dirigeant réseau de l’information * Dirigeant sectoriel de l’information * Responsable organisationnel de la sécurité de l’information * Conseillé organisationnel en sécurité de l’information * Coordonnateur organisationnel de gestion des incidents * Détenteur de l’information * Responsable de la vérification interne * Responsable de l’accès à l’information et de la protection des renseignements personnels * Responsable de l’architecture de sécurité de l’information * Comités de sécurité * Responsable de la continuité des services * Responsable de la sécurité physique * Responsable de la gestion des technologies de l’information * Responsable de la gestion documentaire * Responsable de l’éthique * Responsable du développement ou de l’acquisition de systèmes d’information | * L’implication de ces intervenants à l’étape de planification permet de couvrir l’ensemble des besoins de l’organisation en matière de sécurité. * De par leurs connaissances, ces intervenants ont un rôle important à jouer lors des étapes de planification et de découverte. * Le responsable de la gestion documentaire ou le responsable de l’accès à l’information et de la protection des renseignements personnels, peut être appelés à communiquer avec le prestataire lors de la phase de découverte. * Enfin, le responsable de la gestion des technologies de l’information doit être impliqué dans la phase de découverte, afin de transmettre aux prestataires ses connaissances sur les systèmes. |

**a.2.** **Règles d’engagement**

Des recommandations sont faites aux organismes quant à l’évaluation de la précision les règles d’engagement des parties lors de l’élaboration du document d’appel d’offres, du contrat ou de l’entente. Aussi les indications utiles en fonction de la nature des tests à réaliser sont :

* **Le système :** Les systèmes d’information à tester doivent bien circonscrits. Par exemple, pour lister des segments réseaux, des adresses IP, de noms d’hôte, des adresses URL (Uniform Resource Locator), des personnes et même des bâtiments à tester. Comme nous l'avons énoncé au point 2.3, les systèmes à tester sont les systèmes les plus susceptibles d’être attaqués, particulièrement ceux qui gèrent de l’information sensible. Ainsi, une attention particulière doit être accordée aux systèmes de missions et aux systèmes qui leur sont inter-reliés.
* **La stratégie de tests :** Les caractéristiques décrites au point 2.4 doivent être précisées :
  + Le niveau d’information disponible ;
  + Le niveau d’accès utilisateur ;
  + Le niveau d’accès réseau.

De plus, dans un même projet, différentes stratégies peuvent être appliquées à différents systèmes. Ce qui implique :

* **L’effort :** Les objectifs, des systèmes et des stratégies de tests choisis permettent une estimation du nombre de jours à allouer par système. Exemple, une infrastructure technologique peut être testée aussi bien en 10 jours-personnes qu’en 100 jours-personnes.
* **L’environnement :** Le choix de l’environnement a un impact directement lié à l'intégrité des résultats. Les tests doivent s’effectuer dans l’environnement de production (ou l'équivalent). Cependant, le pentest peut compromettre l’environnement. L’utilisation d’un environnement de test ou de développement est donc nécessaire.
* **La plage horaire :** En cas de nuisances du au test, une restriction des heures est conseillée, quoique ceci limite le réalisme du test.
* **Le changement de prestataire :** Un changement périodique de prestataire permet d’assurer la qualité et l’intégrité des tests. Et donc pour une expertise plus élaborée, l’utilisation de plusieurs prestataires peut s’avérer intéressante.
* **Le mode de facturation :** Les modalités de facturation doivent être judicieusement sélectionnées en fonction des stratégies retenues et précisées.
* **L’exploitation de vulnérabilité :** Pour parfaire le réalisme du test et sans nuire à celui-ci, il est recommandé de ne pas exiger de contraintes à l’exploitation de vulnérabilités. Aussi plutôt que d’interdire l’exploitation de vulnérabilités des systèmes jugés critiques, la réalisation des tests doit être prévue pour une plage horaire moins critique, avec informations des intervenants concernés. Ce qui permet une prise en charge rapide d’éventuels incidents et une restriction des impacts.
* **La relève :** Une capacité de relève du personnel peut être demandée aux prestataires de service. En plus d'assurer une expertise en quantité suffisante, la relève permet le remplacement rapide des intervenants au besoin.
* **L’échéancier :** Pour une vue d’ensemble du test et une bonne organisation de celui-ci, il est nécessaire de réaliser un échéancier qui au besoin, prend en compte périodes critiques pendant lesquelles le système est fonctionnel.
* **Le bien livrable :** Le point 3.3.5 contient plusieurs éléments d'information nécessairement au rapport ou aux documents connexes.Les biens livrables doivent y être bien définis.
* **La stratégie de transfert d’expertise :** Pour garantir une prise en charge adéquate par l’organisme, des solutions proposées.
* **La documentation pour la réalisation :** En fonction de la nature des tests, Le prestataire a besoin d’une documentation qui s’adapte au travail à effectuer, a l’instar d’une liste des adresses IP pour des tests de base ou des documents d’architecture pour des tests concentrés ou complets.
* **Le processus d’escalade :** Un processus doit être mis sur pieds afin que les intervenants concernés par une vulnérabilité critique identifiée ou par un incident rapidement informe.
* **Un moyen de communication :** Laréduction des risques d’ambiguïtés ou d’incompréhension sur les vulnérabilités identifiées, est établie sur des moyens de communication bien définis, a l’instar de la transmission sécurisée par courriel, de l'information sensible ou du suivi par téléphone, ainsi que le chiffrement pour ’assurer l’intégrité et la confidentialité des documents échangés.
* **L’infonuagique (Cloud Computing) :** Pour une infrastructure hébergée chez des tiers, il importe de prendre attache avec les hébergeurs avant de procéder aux tests, pour d’éviter éventuelles interprétations, comme dans le cas où un hébergeur pourrait rendre indisponible un service par souci d’attaque sur son infrastructure au cours du test, s’il n’en a pas été informé.

En somme, des règles d’engagement définies, doivent être prises en compte dans la planification des travaux de test. Cette planification doit être assez flexible et ouverte à des ajustements fonction des résultats obtenus pendant la réalisation du pentest. En effet, tenant compte de l’absence d’information sur le nombre et le type de vulnérabilités en début d’activité, des mesures d’ajustement seront nécessaire pour répondre aux échéances et au budget, dans optimisation de la qualité et la documentation détaillée et complète du test.

**b.** **Découverte**

A ce niveau, nous mettons l’accent sur le fonctionnement du système mis à l’épreuve ainsi que sur la découverte de vulnérabilités.

**b.1.** **Types de vulnérabilités**

Nous avons pu catégoriser les vulnérabilités par type. De cette manière, nous avons pu entrer autres, visualiser rapidement les conséquences d’une vulnérabilité identifiée (cf. tableau 7, inspiré du National Vulnerability Database (NVD 19), qui dresse la liste des principaux types de vulnérabilités, certainement mis à jour depuis).

*Tableau : Type de vulnérabilités (inspiré de NVD19)*

|  |  |
| --- | --- |
| **TPE DE VULNERABILITE** | **EXPLICATION** |
| Mécanisme  d’authentification | Échec à authentifier adéquatement les utilisateurs. |
| Gestion de la session | Échec à créer, stocker, transmettre et protéger adéquatement l'information sensible de sessions telle que les mots de passe. |
| Permissions, privilèges et  contrôle d’accès | Échec à appliquer les permissions et autres restrictions d’accès aux ressources, ou problème de gestion de privilèges. |
| Tampon | Dépassement de tampon, causé par une mauvaise gestion de celui-ci, permettant d’insérer plus d’information que la limite possible et créant ainsi une potentielle injection de code en mémoire. |
| Cross-Site Request  Forgery (CSRF) | Échec à vérifier qu’une requête Web effectuée par un utilisateur provient de lui-même. |
| Cross-Site Scripting (XSS) | Echec d’un site à valider, filtrer ou encoder adéquatement l'information envoyée par un utilisateur avant de la lui retourner. |
| Cryptographie | Utilisation d’un algorithme de chiffrement non sécuritaire ou mauvaise utilisation d’un algorithme. |
| Parcours de chemin  d’accès | Échec à valider adéquatement les chemins d’accès, permettant d’accéder à des fichiers en dehors du ou des répertoires prévus. |
| Injection | Échec à valider les données d’utilisateurs ou les téléchargements de fichiers, permettant l’exécution de code arbitraire sur le système. |
| Configuration | Mauvaise configuration d’un système de l’organisation, permettant une utilisation non sécuritaire de celui-ci. |
| Fuite d’information | Exposition d’information système, sensible ou privée |
| Situation de compétition  (Race Conditions) | Défaut d'un système, caractérisé par un résultat différent selon l’ordre dans lequel agissent les composants et clients du système. |
| Architecture | Défaut de conception qui n’est pas causé par un problème d’implantation ou de configuration. |

Nous pouvons alors en déduire que le potentiel important des vulnérabilités est fonction des systèmes mis à l’épreuve.

Cependant, un pentest peut se restreindre à un type de vulnérabilité, à l’instar du test de vulnérabilité Web dont les essais peuvent être restreints à dix vulnérabilités importantes pouvant être identifiées par un organisme reconnu comme l’OWASP (Open Web Application Security Project).

**b.2.** **Prise de connaissances des systèmes**

Elle consiste en la prise de connaissance du système à tester, ce qui permet de repartir le système en composants. Ici, la détermination des principaux éléments utiles à la détection des vulnérabilités, est réalisée grâce à une analyse des systèmes et de la documentation disponible. Aussi, nous avons :

* Les types de services à l’instar d’un du service Web, du service de courriel entrant/sortant, du service de messagerie instantanée, du service de base de données, etc.
* Les produits utiles à l’offrir le service, comme le service de base de données, où nous retrouvons MySQL, MS SQL Server, PostgreSQL, etc …
* La version des produits utilisés, avec le produit MS SQL Server par exemple, ou nos avons entre autres, les versions 2000, 2005, 2008, 2012, avec pour chacune, les possibles révisions ou ensembles de services (service packs).

Ainsi, pour un test de base, un balayage des adresses peut être utilise pour identifier les services disponibles et d’obtenir le maximum d’information sur eux. Un balayage consiste à tester un certain nombre de ports, sur un ensemble d’adresses IP, pour déceler des différents services et fonctionnalités accessibles et vulnérables.

Par avec un test concentré ou complet, en plus du balayage et de la lecture de la documentation, le soutien des différents échanges avec les intervenants concernés (interviews) est généralement suffisant pour appréhender fonctionnement des services. Toutefois, il incombe à l’organisme de répondre correctement aux questions, en fonction du niveau d’information précédemment rendu disponible dans les temps, que le prestataire ait à disposition, l’essentiel lui permettant de déceler le plus de vulnérabilités possibles. A l’issue de quoi, les composants susceptibles d’être vulnérables sont identifiés, et la véritable quête de vulnérabilités peut débuter. Par exemple, les failles de type XSS et CSRF sont applicables seulement à des sites Web auxquels ont accédé des navigateurs, et non des clients lourds, c’est-à-dire des logiciels qui échangent automatiquement de l'information avec un serveur, mais dont l’intégralité du traitement est assurée par eux-mêmes.

**b.3.** **Identification des vulnérabilités**

Une pléthore de techniques permet déceler des vulnérabilités, et celles-ci sont fonction, entre autres, de l’organisation, des systèmes, de l’expertise disponible et de la stratégie de tests, ainsi que du niveau d’information disponible, du niveau d’accès utilisateur et du niveau d’accès réseau. En outre, ces techniques se regroupent en deux : selon une approche automatisée et selon une approche manuelle.

* **Approche automatisée**

Elle se résume en utilisation des outils, souvent appelés balayeurs de vulnérabilités (scanners), avec la capacité d’effectuer un nombre élevé de tests en un temps réduit. Cette approche se nomme aussi “ balayage “, et permet d’analyser des anomalies dans le but de découvrir des vulnérabilités, comme pour le test d’un site web ou elle servirait à effectuer un balayage rapide du serveur Web. Avec ce type d’outil, nous pourrons aussi découvrir des failles liées à une mauvaise configuration.

Cependant, les outils de cette approche ne sont pas toujours capables d’identifier des vulnérabilités applicatives ou des vulnérabilités propres a un produit utilisé. Un autre outil serait alors nécessaire pour identifier des vulnérabilités à un tel niveau.

En outre, le pentest ne saurait se limiter à des outils automatisés, car ceux-ci ne prennent pas assez compte du contexte du système, ce qui augmenterait considérablement le nombre de faux positifs. De plus, les filtres de sécurité tels que les pares-feux applicatifs peuvent influencer le fonctionnement de tels outils. Une situation décrivant ce fait est proposée à l’Annexe V, avec un pare-feu applicatif protégeant une application contre l’injection de certains caractères spéciaux utiles pour la réalisation de certaines attaques et rendant difficile la découverte de vulnérabilités de manière automatique. Par contre, une approche manuelle permettrait de pousser plus loin l’identification de vulnérabilités et leur exploitation.

* **Approche manuelle**

Pour l’approche manuelle, chaque composante du système est minutieusement analysée grâce à des moyens spécifique à leur environnement. Pour reprendre l’exemple précédent, l’authentification peut faire intervenir un serveur mandataire (proxy), pour intercepter des requêtes et réponses afin de les lire ou les modifier avant leur envoi. Ce qui va offrir par la même occasion la possibilité d’effectuer des injections directement dans les champs d’une page Web en via un navigateur web.

Nous pouvons noter ici, l’exigence de cette approche en matière de connaissances tant en Web, en programmation et en base de données, qu’en des compétences en sécurité de l’information.

Figure : Sous-étapes de l’exploitation

Ainsi en dépit de sa complexité, l’approche manuelle est de loin, la plus efficiente des deux. Surtout que nous avons noté que généralement, il est alloué 20 % du temps à l’approche automatisée et les 80 % restant sont dédié à l’approche manuelle.

**c. Exploitation**

L’étape d’exploitation relative aux tests d’intrusions a pour objectif d’infiltrer le système mis à l’épreuve jusqu’à ce que le niveau d’accès souhaité soit acquis, en exploitant les vulnérabilités précédemment décelées et en utilisant des techniques utilisées par les attaquants malveillants. Il est important de rappeler que cette étape doit être soigneusement réalisée et que la règle d’engagement concernant l’exploitation de vulnérabilités doit être rigoureusement respectée. La figure 3 présente le découpage de l’étape d’exploitation en quatre sous-étapes.

**c.1 Gain d’accès**

Le gain d’accès consiste à obtenir frauduleusement un accès à une ressource (information) ou à un système, après une ou de plusieurs exploitations de vulnérabilité. Alors, autant il est possible d’exploiter une vulnérabilité de type « dépassement de tampon » et d'obtenir un accès direct à un service, Il est possible d’effectuer une simulation de campagne d’hameçonnage (phishing) ou encore une attaque par ingénierie sociale, afin d’inciter un utilisateur à cliquer sur un lien malveillant ou à ouvrir un document permettant l’exécution de code malveillant et la prise de contrôle d’un poste.

**c.2. Élévation de privilèges**

Après avoir accéder à un système par exemple, le pentesteur se sert de techniques d’élévation de privilèges pour passer d’un compte utilisateur à un compte administrateur. Dans le précédent exemple, il est possible que l’accès acquis ne suffise pas pour accéder à des données sensibles, comme le rechercherait un attaquant, il faudra alors qu’il puisse trouver le moyen de s’octroyer les droits lui permettant d’y accéder.

**c.3.** Découverte de nouveaux systèmes

À la suite d'une élévation de privilèges positive, une nouvelle découverte est nécessaire. Aussi, le poste sous contrôle peut servir de pivot pour s’attaquer à d’autres systèmes ciblés ou non initialement par des tests. Si ces systèmes n’étaient pas compris dans la portée préalablement identifiée, il est conseillé de voir dans quelle mesure la poursuite des tests peut se faire de manière approfondie, et même le cas échéant, de revoir le plan de travail.

**c.4. Installation d’outils additionnels**

A l’issue des actions précédentes, le pentesteur tel un véritable attaquant examinera la possibilité d’installer une « porte dérobée » (backdoor), c’est-à-dire des outils additionnels pour s’assurer de pouvoir retourner dans le système en temps voulu. Si le poste de travail était mis à jour ou redémarré par exemple au cours du test, l’accès au poste peut être perdu. Seule l’installation d’outils additionnels pourra garantir la constante possibilité d’accès, afin de poursuivre d’éventuelles attaques. Mais surtout, le testeur doit désinstaller ceux-ci une fois l’intervention terminée. Cependant, en plus de posséder des compétences très techniques, le pentesteur doit également être capable de rédiger et **mettre en** **forme** le résultat de recherches effectuées.

**d. Rapport**

Cette partie ne doit pas être négligée, car même le meilleur des travaux est inutile s’il n'est pas restitué. Nous allons donc voir ici comment effectuer la restitution.

Le **rapport de test d'intrusion** est un élément essentiel pour le client (cible du pentest) car ce sera un guide pour la correction des vulnérabilités trouvées.

Il est le principal bien livrable produit à la suite des tests d’intrusions et de vulnérabilités. Ce rapport doit être rédigé avec une grande rigueur, puisque ce document permet d’appuyer les décisions relatives à l’acceptation des solutions recommandées et de remédier aux vulnérabilités décelées. Bien que le niveau de détails varie en fonction de l’envergure et de la profondeur des tests, un rapport de tests d’intrusions doit minimalement contenir les éléments d’information suivants :

* Les objectifs du mandat ;
* La portée du mandat ;
* Le contexte du mandat ;
* La démarche d’évaluation du risque ;
* La démarche d’identification des vulnérabilités ;
* Une description détaillée de chacun des tests effectués, concluants ou non, et les résultats obtenus ;
* Une documentation détaillée de chaque vulnérabilité identifiée l’Annexe V en donne un exemple complet) ;
* Des recommandations générales.

Il est également intéressant pour l’organisme :

* De déterminer et de documenter les forces et les faiblesses des mesures de sécurité examinées, afin d’appuyer les recommandations et d’éclairer l’organisme public dans le cadre de sa prise de décision ;
* De documenter les vulnérabilités potentiellement présentes qui n’ont pu être exploitées, tel un potentiel de déni de services (denial of services ou DoS) dans un environnement de production ;
* De produire un rapport synthèse, de type sommaire de gestion, complet et d’interprétation facile par la haute direction ;
* De préparer une présentation à l’intention de la haute direction ;
* De rédiger un journal des activités dans le cadre de tests sur un environnement de production, en précisant la date et l'heure ainsi que la nature de chaque activité effectuée par le prestataire, notamment les balayages et les attaques.

Ajoutons que, généralement le contenu d’un rapport doit être en adéquation avec le **niveau technique** et les **compétences** du destinataire. Dans le cas du rapport de test d'intrusion, il peut s'adresser aux personnes suivantes de l’entreprise :

* Le **DSI,** directeur des systèmes d'information
* Le **responsable système et réseau**
* L'équipe **d'informaticiens** en charge de l'administration du système d’information
* Les **développeurs** d'une application interne à l'entrepris
* L'entreprise **prestataire** de service s'occupant de la maintenance du système informatique
* Le **gérant** ou le chef d'entreprise pour les entreprises de plus petite taille

**NB** : Dans certains de ses rôles, il n'est pas nécessaire d'avoir de compétences en sécurité informatique. Le rapport doit donc être rédigé de telle façon qu'une personne (même non-informaticienne puisse le lire, tout en comportant des éléments techniques pointus.

Le rapport se présente de façon diverse suivant le responsable en charge de la réalisation du pentest (prestataire ou autre). Cependant, il est souvent formaté de la manière suivante :

* **Format** : Format **PDF** non modifiable, de quelques dizaines de pages.
* **Langue** : En **français** ou **anglais** en général.
* **Confidentiel** : Le rapport contenant des informations permettant de s'introduire dans l'entreprise, il est impératif de signaler que c’est comme un document confidentiel. Il est possible par exemple d’insérer un **filigrane** sur chaque page.

Il contient également toujours tous ces éléments dans l’ordre suivant :

* Le **sommaire** : comme dans tout document assez long, il est conseillé d'avoir un sommaire pour une meilleure appréhension de la structure et faciliter sa lecture.
* Le **contexte** et le **périmètre** : il est bon de rappeler la raison pour laquelle le test a été effectué, et le périmètre du test d'intrusion, par exemple les adresses IP qui ont été testées.
* Les **conditions du test** : il est bien expliqué qu'il existe plusieurs types de tests d'intrusion (boîte noire, boîte grise, boîte blanche). Il est important de préciser lequel de ces tests a été effectué.
* La **méthodologie** : il existe plein de méthodologie de test d'intrusion (OWASP, PCI, Penetration Testing Execution Standard, OSSTMM, etc…). Il peut être intéressant pour les lecteurs de savoir quelle méthodologie a été choisie. À défaut, il faut indiquer le processus de test qui a été suivi.
* Les **axes d’évaluation** : souvent les 3 axes d'évaluation utilisés pour qualifier les vulnérabilités ne sont pas forcément intuitifs. Il est bon de les expliquer pour que le lecteur comprenne plus facilement.
* Les **résultats du test** : La plupart du temps sous forme de tableau, il est présenté comme un listing des vulnérabilités trouvées, avec les différentes informations les qualifiant.
* Une **synthèse** : pour les lecteurs qui veulent juste avoir un aperçu de la sécurité sans lire l'intégralité du rapport, il est intéressant de mettre une partie de synthèse récapitulant le nombre de vulnérabilités trouvées en fonction :
  + De la sévérité
  + Du niveau de priorité de correction
  + De la difficulté de correction

Ainsi il peut être intéressant de résumer le nombre de failles trouvées dans des petits tableaux de synthèse par sévérité, priorité, et difficulté de correction. Par exemple, noter qu’il existe 2 failles de sévérité intolérables, 5 modérées et 3 acceptables.

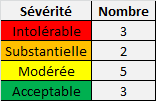
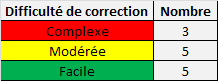
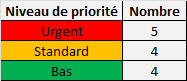


Figure :

En bref, Le rapport de test d'intrusion est un **document technique** qui doit également pouvoir être lu par des responsables sans compétences informatiques poussées. Il doit être **confidentiel,** car il contient des informations sensibles qui peuvent permettre à une personne malveillante de s'introduire dans le système informatique. Et enfin, Il n’y a pas de structure unique de rapport d’intrusion, mais certains éléments sont indispensables.

En somme, il existe plusieurs types d'attaques, avec des niveaux de complexité différents, et qui permettent d'exploiter les vulnérabilités des systèmes. Aussi, nous nous sommes plus appesantis sur les attaques web pouvant impacter des SaaS. Les attaques par déni de service quant à elles sont plutôt basique basiques. Il existe différentes techniques pour attaquer un SaaS compte tenu de sa constitution : l'utilisation d’une faille logicielle, l'attaque par paquets forgés, l'intégration de paramètres dans l’adresse, ou encore le cross site scripting. Aussi, un serveur SQL peut se faire attaquer soit directement, soit au travers du serveur Web pour lequel il héberge la base de données.



## Partie II : Analyse et implémentation



## 

## Chapitre 3 : Cadrage et analyse du projet

Ce chapitre est réservé au cadrage du projet qui consiste à définir le contexte du projet et étudier le processus de sécurisation existant au sein de l’entreprise ensuite décrire le cahier de charge dans lequel l’on expliquera les besoins, les délais et les coûts du projet, et enfin présenter la méthodologie et les outils que nous envisageons d’utiliser.

#### Section 1. Contexte du projet et étude de l’existant

1.Contexte du projet

L’entreprise LAO SARL est une société de services et d’ingénierie logicielle. Elle a été le cadre qui a favorisé la naissance de cette idée de projet parce qu’elle confrontée au problème de sécurisation des SaaS qu’elle compte mettre à la disposition des clients. En effet, malgré les dispositions prises lors de la réalisation de ceux-ci, des limites de sécurité ont été constatées au niveau de certaines applications qu’ils abritent.

Comme tout entreprise digne de ce nom, des audits ont été réalisés, sauf que ceux-ci ne visaient alors que l’aspect normatif de ses applications du point de vue des vulnérabilités avec un scanner de vulnérabilités. Ceux-ci permettent d’énumérer les vulnérabilités, sans tenter de les qualifier ou de vérifier si elles sont exploitables. Aussi **LAO SARL** utilise en son sein le logiciel **NESSUS** qui est un scanner de vulnérabilités à adapté à cet effet.

2.Processus d’audit sécurité existant

Les audits de sécurités des applications SaaS sont faits à **LAO SARL** de façon automatique c’est-à-dire qu’il n’est pas besoin d’une personne spécialement recrutée pour effectuer cet audit ; en effet, le processus se faisait après la réalisation et la validation complète de l’application. Ici., on effectue un scan de vulnérabilités sur les applications, les vulnérabilités relevées, sont étudiées et corriges ; par contre s’il n’y en a aucune, elle est jugée sécurisée et mise en production.

Dans le cadre de ce travail, pour des raisons de confidentialités, nous ne présenterons pas les applications de l’entreprise, mais un environnement qui s’en rapproche au mieux avec les différentes actions menées dans le cadre de notre projet. Il s’agira donc de l’application Juice shop de l’OWASP, à laquelle nous soumettrons le pentesting.



Figure: Juice shop

1. Constat

Le scan de vulnérabilité est une composante du test d’intrusion, c’est-à-dire une sous-partie. C’est plus précisément un scan (comme son nom l’indique) de la cible qui permet d’énumérer les vulnérabilités, sans tenter de les qualifier ou de vérifier si elles sont exploitables.

Pendant l’opération de scan de vulnérabilités, lorsqu’une vulnérabilité est de couverte, le responsable fait un rapport avec une proposition de solution sans vérification de l’effectivité d’une quelconque menace. Il crée une notification et afin que les développeurs se charges des corrections nécessaires à l’inhibition de la faille.

Ce qui va engager des heures de travaux supplémentaires et dont d’avantages de dépenses en termes de solutions de sécurisation.

A partir de ce constat, l’on peut penser à la possibilité d’envisager un audit de type pentesting qui permettra de limiter au maximum des cas de faux positifs.

#### SECTION 2 : Cahier de charge

1. Justification de l’étude

Dans un contexte ainsi dominé par l’informatisation dans les différents secteurs d’activités, l’un des soucis majeurs des éditeurs ou encore des sociétés de services et d’ingénierie informatique est de mettre au point des solutions logicielles, exempt de failles de sécurité. Mais dans beaucoup d’entreprise éditrice de logiciels comme LAO SARL c’est au moment de la mise en œuvre ou de l’utilisation d’un produit logiciel que l’on se rend souvent compet des vulnérabilités existantes et qui pourrons permettre à un éventuel attaquant de le détourner de son but originel. Ceci peut entrainer des désagréments graves à l’endroit de l’entreprise (dépenses supplémentaires) et des clients c’est-à-dire des organisations et entreprises qui ont bien voulus se doter de ces logiciels SaaS. Avec pour effet, des préjudices graves autant pour les clients que pour un fournisseur de services tel LAO SARL.

Le pentesting est une solution qui permet de s’assurer de l’effectivité des menaces et de limites ici les engagements en termes de dépenses et autres. Les audits de sécurités doivent aller beaucoup plus loin que de simples scans de vulnérabilités.

En plus, de faire gagner du temps et l’argent, le pentesting offre également l’avantage de réduire les limites sécuritaires, quoique la mise en œuvre peut s'avérer être un vrai défi. En effet, ce qui fonctionne aujourd’hui peut ne plus fonctionner demain, et le problème s’aggrave en présence d’un manque de compétences adaptées.

1. **Problématique**

Au regard de ces situations, nous nous sommes posé la question centrale suivante : Comment procéder à un pentesting pouvant garantir la fiabilité d’une application SaaS avant sa mise en production ? Cette question centrale nous amène à nous poser deux autres questions auxquelles les réponses contribueraient certainement à résoudre le problème central posé à savoir : quelles sont les étapes qui doivent impérativement se faire lors de l’audit de sécurisations d’une application SaaS et comment venir à bout de vulnérabilités de manière efficace afin de prévenir ou éradiquer toute formes de failles du produit logiciel final.

1. **Intérêt et objectifs**

L’intérêt de notre travail se perçoit clairement en ce qu’un bon service rendu, des applications SaaS qui résistent aux attaques sur internet, avec une satisfaction des clients et une bonne continuité des activités, ainsi qu’une garantie de sécurité optimale. Tout ceci pour contribuer à la fidélisation des clients, à la notoriété de la société éditrice ou du fournisseur et donc du gain de chacune des parties

L’objectif principal de notre travail est de pratiquer un pentesting sur un environnement applicatif capable de d’identifier et évaluer les vulnérabilités, puis trouver les correctifs pour chacune d’elle. Cet objectif principal se décline dans les objectifs spécifiques suivantes :

* Définir les différentes étapes du pentesting de notre environnement
* Tester ces vulnérabilités
* Produire un rapport de test qui envisage des solutions et mesures préventives
  1. Spécifications du projet

1. **Choix de la méthodologie**

Pour la méthodologie à adopter, nous avons oriente notre choix sur la base des éléments cites la base des éléments ci-haut cités. En effet, elle doit être adaptée, dans sa mise en œuvre, à la réalité (métier et taille des entités auditées) et permettre d'aboutir à l'élaboration d’un bilan, de recommandations et de solutions pragmatiques et pertinentes, qui tiennent compte, pour les plus urgentes, de la réalité de l'entreprise. En plus elle est en corrélation avec à la gravité des failles décelées et l'efficacité, l’urgence et faisabilité des actions à mener.

Ainsi, il apparait clairement que la méthodologie d'audit que nous envisageons mettre en œuvre, tout en fournissant des références sur son adéquation avec le référentiel ISO/CEI 27002 est le pentesting. Cependant, l’entreprise tient compte dans son évaluation de la consistance de la méthodologie proposée, ou des parties de cette méthodologie et de chacune des phases avec leur adéquation à sa réalité et au temps imparti.

Nous devrons alors spécifier dans la rubrique « Planning prévisionnel de la mission », au minimum, et pour chaque composant du SaaS:

* Le type de la méthodologie à mettre en œuvre pour l’applicatif et les structures recensées utiles à interviewer, ainsi que les outils logiciels accompagnant la mise en œuvre de cette méthodologie (traitement automatisé des interviews et calcul des risques associés...),
* La méthode de mise en œuvre du volet technique, en spécifiant les types de tests techniques à effectuer et leur objectif, ainsi que les outils utilisés,
* La séquence des actions à mener (interviews, tests techniques, synthèse, rédaction de rapports...)
* Une estimation de la charge homme/jour de chaque action, incluant un résumé des corrections de volumétrie proposées par rapport à l'estimation préliminaire proposée dans le cahier des charges,
* La liste nominative des équipes qui interviendront pour chaque composant de la structure du SaaS avec la référence de l'expérience dans la mise en œuvre de la méthodologie et outils consignés.

1. **Conduite et déroulement de la mission**

La mission d'audit sécurité, objet du présent travail, couvre les aspects essentiellement l’aspect logiciel relatif à la sécurité de l'ensemble des entités et moyens suivants :

* Structure,
* Outils logiciels,
* Éléments de sécurité
* …

Tous en relation avec les fonctions de traitement de l'information au niveau l’application.

En outre, la conduite de la mission et notamment les différentes réunions préparatoires de lancement ainsi que les séances de validation des rapports de chaque phase sont obligatoires tant pour l’entreprise que pour nous.

Notre mission se décompose en cinq phases, comme présenté ci-après.

**b.1 Phase 1 : Préparation de la mission**

**b.1.1 Accompagnement Pré-audit**

Cette phase consiste à assister l'entreprise à définir les besoins de sécurité d’un SaaS par rapport aux objectifs principaux de sécurité (Confidentialité, Intégrité et Disponibilité). Cette définition des besoins nous permettra de proposer les exigences minimales de sécurité et de choisir les contrôles de sécurité appropriés à appliquer.

Le processus de sélection de ces contrôles impliquer la direction et le personnel opérationnel au sein de l'organisme.

**b.1.2 Actions de lancement de la mission**

Au lancement de la mission, il est nécessaire de demander à avoir de l’entreprise tout détail, information ou document nécessaire pour l'exercice de la mission.

Des réunions préparatoires de la mission sont organisées en début de mission, dont l'objet est de finaliser, sur la base des besoins et documents préparés par l’entreprise, les détails de mise en œuvre de la mission. Elles concerneront, sans s'y limiter, la finalisation des détails suivants :

* Désignation des chefs de projets et des interlocuteurs, du côté de l’entreprise
* Fourniture par l'entreprise des détails complémentaires, relatifs au périmètre de l'audit de sécurité,
* Validation du périmètre de l'audit,
* Fourniture par l'entreprise des documents requis pour l'audit (Manuels d'exploitation, Cartographie procédurale...),
* Fourniture du document de définition des besoins de sécurité. Si ce document n’existe pas, le maitre d'ouvrage tâchera à le préparer et le fournira au l’entreprise avant le démarrage de l'audit sur site,
* Détermination de la conformité des documents existants aux exigences de la norme ISO/IEC 27002, arrêt de la liste des documents manquants exigés par cette norme et examen des problèmes éventuels, relatifs à la mise à jour de la documentation,
* Examen des détails des listes des interviews à réaliser par le soumissionnaire et fourniture par l'entreprise de la liste nominative des personnes à interviewer,
* Affinement des plannings d'exécution (planning des actions/site, plannings des réunions de coordination et de synthèse…),
* Examen des détails logistiques nécessaires au déroulement de la mission (octroi des autorisations d'accès aux lieux où l'audit devrait être élaboré sur la base d'études de terrain, octroi de locaux de travail au soumissionnaire retenu...).

Ainsi tous les détails de mise en œuvre doivent être examinés et validés. Ces réunions déboucheront, entre autres, sur la finalisation des plannings précis et détaillés de mise en œuvre de la mission. Quant aux résultats de cette réunion, ils seront consignés dans PV, qui sera annexé au rapport final d'audit.

En cas de difficultés notoires rencontrées lors de cette phase, le titulaire devra faire recours au Maître d'Ouvrage par écrit, pour lui permettre d'intervenir efficacement et dans les délais.

**b.1.3 Actions de sensibilisation pré-audit**

Dans l'objectif de sensibiliser les responsables et acteurs des SaaS de l’entreprise, il faut assurer des actions de sensibilisation préliminaires sont assurées. Ceux-ci ont pour premier objectif une sensibilisation générale sur les dangers cybernétiques et les risques cachés encourus, incluant entre autres la présentation pratique d'attaques cybernétiques. Elles recherchent aussi l’octroi de la transparence et collaboration des utilisateurs, en rappelant les objectifs de l'audit de sécurité, l'urgence et les bienfaits attendus, ainsi que l'assurance sur la confidentialité des données reçues.

A la fin de cette opération un PV sera dressé et signé conjointement par le titulaire et le maitre d'ouvrage et des fiches d'évaluation de ces sessions seront remplies par les participants. Des copies de ce PV et de ces fiches seront jointes au rapport d'audit.

**b.2 Phase 2 : Audit sur terrain**

C'est la phase d'audit proprement dite. Ici, l’équipe intervenante du titulaire de la commande doit obligatoirement participer à l’audit des entités en en présentiel. Aussi, l’équipe intervenante du titulaire de la commande doit effectuer l’audit moyennant des outils et des tests et convoquer le personnel, au siège de l’entreprise, pour réaliser les interviews.

De plus, cette phase couvre principalement trois (03) volets :

* Un volet d'audit organisationnel,
* Un volet d'audit technique
* Un volet d'appréciation des risques.

**b.2.1 Volet audit organisationnel et physique**

Il s'agit, pour ce volet, d'évaluer les aspects organisationnels de gestion de la sécurité de la structure objet de l'audit, d'estimer les risques et de proposer les recommandations adéquates pour la mise en place des mesures organisationnelles et d'une politique sécuritaire adéquate. Ce volet doit couvrir les aspects de gestion et d'organisation de la sécurité, sur les plans organisationnels, humains et physiques.

Au cours de cette étape, nous devons adopter une approche méthodologique, basée sur des batteries de questionnaires préétablis et adaptés à la réalité des entités auditées, permettant d'aboutir à une évaluation pragmatique des failles et des risques encourus, ainsi qu'à l'identification et classification des ressources relativement à leur criticité. Cet audit doit prendre comme référentiel tous les chapitres de la dernière version de la norme ISO/CEI 27002.

**b.2.2 Volet audit technique**

**b.2.2.1 Objectifs de l’audit technique**

Ce volet concerne l'audit technique (objet principale de notre travail) de l'architecture de sécurité. Il s'agit de procéder à une analyse très fine de l'infrastructure sécuritaire du des SaaS et particulièrement des applications. Cette analyse doit faire apparaître les failles et les risques conséquents d'intrusions actives (tentatives de fraude, accès et manipulation illicites de données, interception de données critiques...), ainsi que celles virales ou automatisées, et ce suite à divers tests de vulnérabilité conduits dans le cadre de cette mission, qui doivent englober des opérations de simulation d'intrusions et tout autre test permettant d'apprécier la robustesse de la sécurité d’un SaaS et sa capacité à préserver les aspects de confidentialité, d'intégrité et de disponibilité.

Au cours de cette étape, nous devons, en réalisant des audits techniques de vulnérabilités, des tests et simulations d'attaques réelles :

* Dégager les écarts entre l'architecture réelle et celle décrite lors des entretiens ou dans la documentation, ainsi qu'entre les procédures techniques de sécurité supposées être appliquées (interviews) et celles réellement mises en œuvre,
* Évaluer, la vulnérabilité et solidité des composantes logicielles du SaaS (réseau, systèmes, mécanismes d'administration et de gestion, plates-formes ...), contre toutes les formes de fraude et d'attaques connues par les spécialistes du domaine au moment où l'audit est conduit, et touchant les aspects de confidentialité, intégrité et disponibilité des informations

Pour les tests intrusifs, nous devons nous engager à les effectuer sans compromettre la disponibilité, l'intégrité ou la confidentialité des SaaS. Ainsi, les tests réalisés ne doivent pas perturber la continuité de service du SaaS audité. Les tests critiques, pouvant provoquer des effets de bord, doivent être notifiés à l'entreprise et doivent être réalisés sous sa supervision, conformément à un planning préalablement établi et validé, et qui pourra concerner des horaires de pause et éventuellement de chôme.

**b.2.2.2 Outils utilisés**

Lors des audits techniques, l'utilisation d'outils commerciaux doit être accompagnée de la présentation d'une copie de la licence originale et nominative, permettant leur usage correct pour de telles missions (inexistence de restrictions quant à leur usage pour les audits : plages d'adresses ouvertes...).

De plus, étant donné qu'aucun produit commercial ne saurait prétendre, à lui seul, à une complétude totale, les outils disponibles dans le domaine du logiciel libre (et généralement utilisés par les attaquants) doivent être savamment déployés pour assurer une complétude correcte de cette phase, en s'appuyant, quand cela est possible, sur des scripts riches de mise en œuvre savante et combinée de ces outils.

Les outils proposés doivent inclure, sans s'y limiter, les catégories d'outils suivants :

* Outils de sondage et de reconnaissance réseau,
* Outils de test automatique de vulnérabilités,
* Outils spécialisés dans l'audit de chaque type de plateformes systèmes (OS...) présentes dans l'infrastructure,
* Outils spécialisés dans l'audit des SGBD existants,
* Outils de test de la solidité des objets d'authentification (fichiers de mots clés...),
* Outils de test de la solidité des moyens de sécurité (WAF, outils d'authentification...),

Cependant, il faudra fournir la référence et une description concise (résumé de la liste des fonctionnalités offertes) des outils et scripts qu'il compte utiliser, en spécifiant l'objectif, lieu (phase de l'audit) et types de fonctionnalités de l'outil ou script qui seront mises en œuvre.

b.2.3 Appréciation des risques

Dans cette phase et après avoir identifié les failles de sécurité techniques, il s'agit de suivre une approche méthodologique pour évaluer les risques encourus et leurs impacts sur la sécurité de la structure auditée. La phase d'appréciation des risques se déroulera en deux étapes :

**b.2.3.1 Étape 1 : Analyse**

Dans cette phase nous sommes amenés à :

* Identifier les processus critiques : les informations traitées, les actifs matériels, les actifs logiciels, les personnels... qui supportent ces processus,
* Identifier les menaces auxquelles sont confrontés ces actifs (intentionnelles ou non intentionnelles),
* Identifier les vulnérabilités (au niveau technique) qui pourraient être exploitées par les menaces,
* Identifier les impacts que les pertes de confidentialité, d'intégrité et de disponibilité peuvent avoir sur les actifs,
* Evaluer la probabilité réaliste d'une défaillance de sécurité au vu des mesures actuellement mises en œuvre.

**b.2.3.2 Étape 2 : Évaluation**

Dans cette étape nous sommes amenés à :

* Établir une classification des risques par niveaux, et déterminer le niveau du risque acceptable,
* Évaluer les risques, en fonction des facteurs identifiés dans la phase d'analyse, et les classifier par niveaux,
* Identifier les mesures préventives et les mesures correctives de sécurité à implémenter pour éliminer ou réduire les risques identifiés.

**b.3 Phase 3 : Synthèse des recommandations - Sensibilisation post-audit**

**b.3.1 Synthèses des recommandations**

A ce stade, nous devons, à la fin de la phase d'audit sur terrain, réaliser :

* Un rapport daté, signé par le responsable de l'audit et portant le cachet du titulaire, énumérant la liste des failles (classées par ordre de gravité et d'impact) ainsi qu'une évaluation de leurs risques,
* Les recommandations portant sur les actions détaillées correctives à entreprendre à court terme,
* La proposition d'un plan d'action cadre s'étalant sur trois années et présentant un planning des mesures stratégiques en matière de sécurité à entreprendre, et d'une manière indicative les moyens humains et financiers à allouer pour réaliser cette stratégie,
* Un rapport de synthèse, destiné à la direction générale, qui inclura d'une manière claire (destiné aux décideurs) les importants résultats de l'estimation des risques, un résumé des importantes mesures techniques préconisées dans l'immédiat et sur le moyen terme (jusqu'au prochain audit), ainsi que les grandes lignes du schéma directeur proposé.

Les recommandations portant sur les actions détaillées correctives à entreprendre à court terme doivent inclure au minimum :

* Les actions détaillées techniques urgentes à mettre en œuvre dans l'immédiat, pour parer aux défaillances les plus graves,
* Les actions techniques à mettre en œuvre sur le court terme (jusqu'à la date du prochain audit), englobant entre autres :
  + Les premières actions et mesures à entreprendre en vue d'assurer la sécurisation de l'ensemble SaaS audité, sur le plan technique (outils et mécanismes de sécurité à mettre en œuvre, incluant une référence aux opportunités et options offertes par les outils disponibles dans le monde du logiciel libre), ainsi qu'éventuellement des aménagements architecturaux de la solution de sécurité existante,
  + Une estimation des formations requises et des ressources humaines et financières supplémentaires nécessitées.

**b.3.2 Actions de sensibilisations post-audit**

Au cours de cette phase, nous devons réaliser des sessions de sensibilisation post-audit destinées aux responsables et aux acteurs qui interviennent sur le SaaS de l’entreprise. Ces sessions auront pour objectif :

* Une sensibilisation aux failles décelées et aux risques cachés encourus,
* L'octroi de la collaboration des utilisateurs, pour ce qui concerne la mise en œuvre de la politique de sécurité proposée en spécifiant l'objectif de cette politique et les bienfaits attendus.

A la fin de cette opération un PV sera dressé et signé conjointement par le titulaire et le maître d'ouvrage et des fiches d'évaluation de ces sessions seront remplies par les participants. Des copies de ce PV et de ces fiches seront jointes au rapport d'audit.

**c. Livrables**

Pour les réceptions des phases de la mission d'audit objet du présent cahier des charges, nous devons livrer à l’entreprise les documents suivants :

* Un rapport des actions de lancement de la
* Un rapport d'audit qui doit couvrir, au minimum les aspects exigés par les dispositions du présent cahier des charges et doit inclure au minimum les chapitres ou rapports suivants :
  + Une analyse des risques encourus,
  + Une liste des recommandations à appliquer dans l'immédiat, en tenant compte des spécificités de l'entité, de la classification des composants (criticité) et de la réalité actuelle des moyens humains et financiers.
  + Une section relative à l'audit technique, cette section est composée en :
    - Un audit des vulnérabilités existantes en précisant leur impact sur la pérennité des systèmes d'information et de communication de la structure ;
    - Une analyse des risques encourus,
    - Une liste des recommandations techniques à appliquer dans l'immédiat pour la correction des failles graves décelées.

Il est à noter que tous les travaux de test et d'analyse effectués doivent être consignés dans une annexe, en les ordonnant selon leur sévérité, en incluant au niveau du rapport un relevé des plus importantes failles et des moyens de les combler dans l'immédiat.

* Une section relative au plan d'action et stratégie de sécurité à appliquer sur le court terme (jusqu'au prochain audit), comprenant des recommandations précises quant aux mesures à prendre dans le court terme, afin de pallier aux failles et insuffisances décelées, incluant tous les nécessaires techniques en tenant compte pour ce qui concerne le déploiement d'outils et d'architectures de sécurité de l'option d'usage d'outils open-source et de la réalité financière et humaine de l'entité.
* Un rapport présentant un plan directeur sur trois années, permettant de mettre en œuvre une stratégie de sécurité cohérente et ciblée. Ce rapport sera mis en à jour lors des audits de la seconde et de la troisième année en tenant compte du taux de réalisation des mesures qui ont été adoptées depuis le dernier audit réalisé et des insuffisances enregistrées dans l'application des recommandations, ainsi que de l'audit de l'année en cours
* Un rapport de synthèse, destiné à la direction générale, qui inclura d’une manière claire (destiné décideurs) les importants résultats de l'estimation des risques, un résumé des importantes mesures techniques préconisées dans l'immédiat et sur le moyen terme (jusqu'au prochain audit), ainsi que les grandes lignes du schéma directeur proposé.

En, bref la réalisation d’un audit de sécurité applicatif s’étend sur un bon nombre d’aspect. Cependant, dans la pratique, chacun de ces points sera fonction du type de l’application intégrée au SaaS, selon par exemple qu’il s’agisse d’une application monolithique ou microservice. Aussi pour compte de l’entreprise LAO sarl le choix est tourné vers les applications orientées microservices qui correspond à la typologie applicative de notre travail.

#### SECTION 3 : Implémentations

Dans cette partie, nous allons montrer l’exécution d’un pentesting sur un environnement réservé à cet effet. Il s’agira particulièrement de l’étape d’exploitation d’un pentesting. Pour ce faire nous allons trouver et exploiter les vulnérabilités dans **OJS, qui est l’environnement que nous avons choisi utiliser pour exposer le travail que nous avons réalisée en entreprise**. Nous nous exécuterons en taches suivant l’ordre méthodologique des étapes du pentesting, citées avant.

##### Tâche 1 : Déploiement de l’OJS

Tout d’abord, déployons notre machine OJS et nous y accédons sur une IP donnée via un n'importe quel navigateur Web.

##### Tâche 2 : Configurer Burp Suite

Nous configurons l’outils Burp Suite

##### Tâche 3 : Parcourir l'application (Découverte)

C'est l'une des étapes les plus importantes pour trouver des vulnérabilités dans une application Web. Nous essayons de comprendre la nature et le comportement de l'application du point de vue du développeur. Certains des détails que nous avons observés sont :

* Dans une page Web sécurisée (http) ;

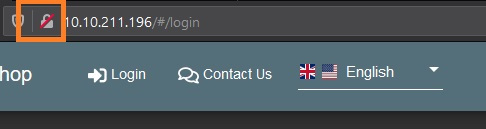


Figure:

* Dans un identifiant de connexion sécurisé et les détails des cookies ;

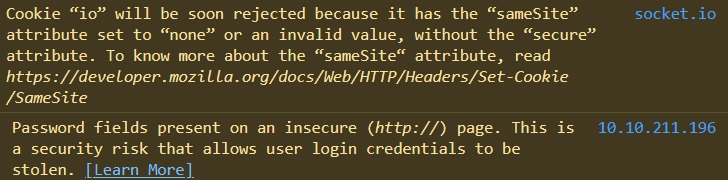


Figure:

* Dans une URL sécurisée ;

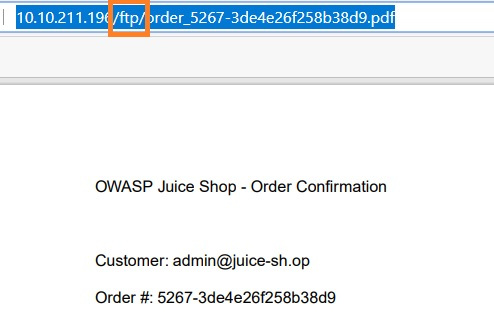


Figure:

##### Tâche 4 : Injection (Exploitation)

A ce niveau, nous allons tenter de nous connecter avec le compte administrateur à l'aide d’une injection SQL. Pour cela, nous allons nous connecter par force brute avec la suite Burp. Pour ce faire, une simulation de connexion est lancée avec comme « admin / admin » comme login et mot de passe. Puis nous interceptons la requête à partir de laquelle nous allons créer la charge nécessaire a notre attaque.

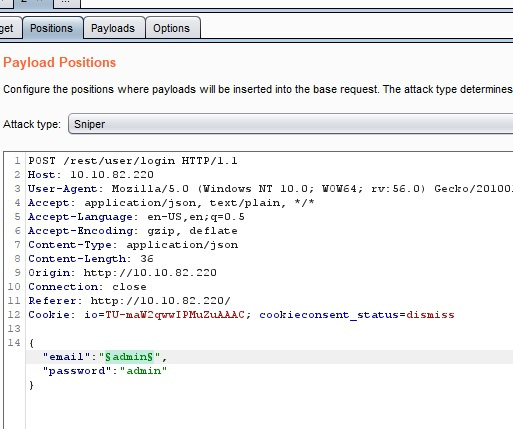


Figure:

Les résultats peuvent être trier selon un en-tête de colonne., nous obtenons ainsi plusieurs charges utiles qui de longueur égale, dans notre cas **"1001"**.

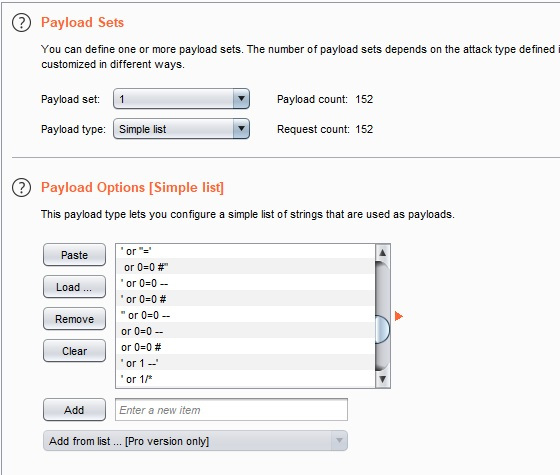


Figure: Juice shop

Une fois l'étape ci-dessus terminée, nous exécutons notre attaque en essayant « **a' ou 1=1– »**. Nous pouvons ainsi nous connecter grâce à la charge utile dans l'e-mail et tout mot de passe aléatoire,

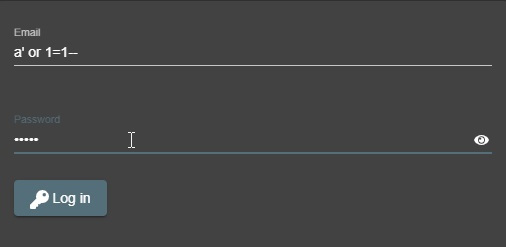


Figure :

La page de connexion a accepté la charge utile comme identifiant de messagerie de l'administrateur et a obtenu l'accès.

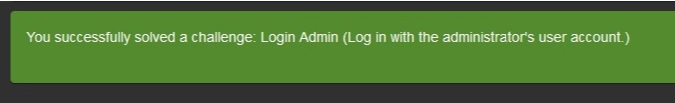


Figure:

##### Tâche 5 : Authentification interrompue

Ici, nous allons tout d’abord r**éinitialiser le**mot**de passe de Jim en utilisant le mécanisme de mot de passe oublié**. Puis, à partir de la tâche n°4, nous allons identifier le compte de l'administrateur en interceptant la demande de connexion, c’est-à-dire "**admin@ juice-sh.op**"

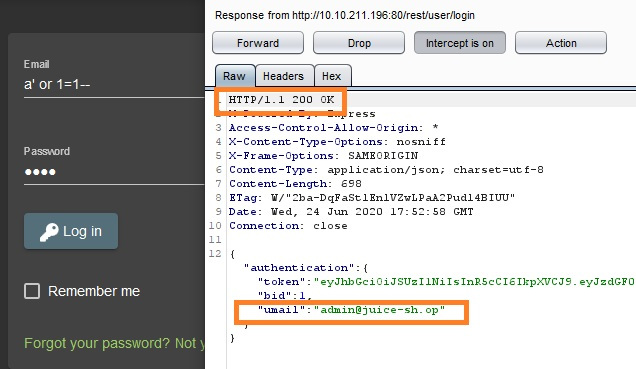


Figure:

Pour trouver le mot de passe Jim, nous avons dérivé l'identifiant de l'e-mail de Jim à partir de l'identifiant de l'e-mail de l'administrateur. Autrement dit, nous allons essayer d’avoir le mot de passe de Jim avec le mécanisme de mot de passe oublié en utilisant **« jim@jus-sh.op » comme adresse mail.**

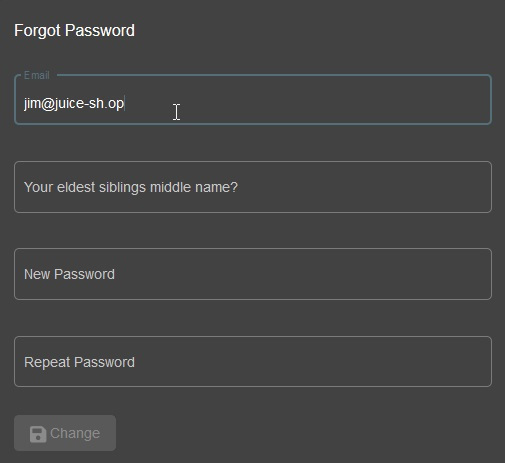


Figure:

Pour générer un nouveau mot de passe, la question de sécurité demande « le deuxième prénom du frère aîné de Jim ». De Google, nous avons obtenu les informations ci-apres (prévu par l’OWAPS).

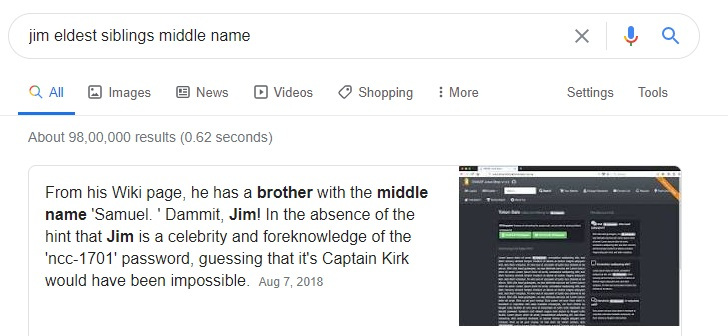


Figure:

En essayant **"Samuel"** comme deuxième prénom des frères et sœurs de Jim, nous parvenons bien à modifier le mot de passe de celui-ci.



Figure:

Maintenant, nous allons effectuer une sous authentification interrompue c’est-à-dire **« identifier le mot de passe administrateur », pour cela, nous** allons encore, forcer brutalement le mot de passe. A cet effet, nous entrons le nom d'utilisateur admin et un mot de passe aléatoire (**"1234"**), puis nous réalisons une interception avec suite Burp.

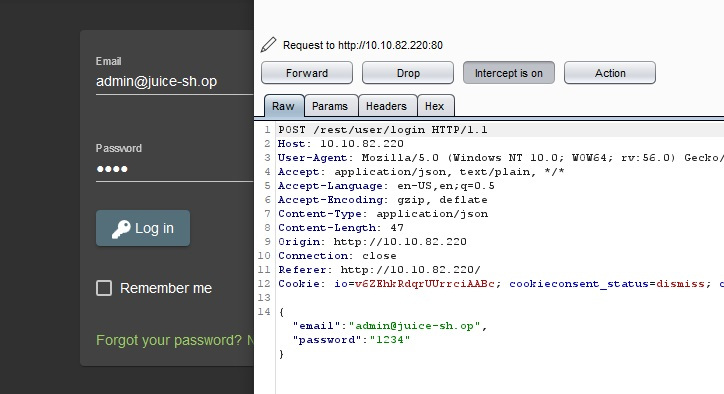


Figure:

Dans les résultats obtenus, nous utiliserons comme charge utile pour le mot de passe **« admin123 »**.

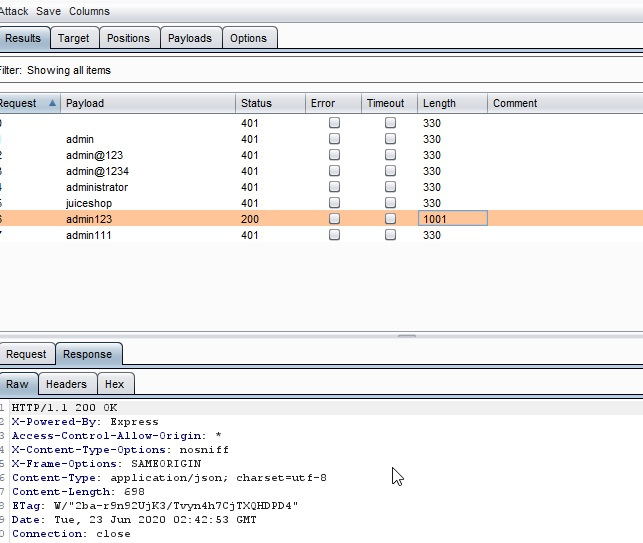
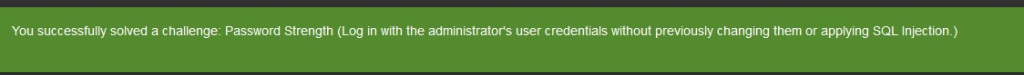


Figure:

Et avec un essai, sur le nom d'utilisateur " et le mot de passe **"admin123""admin@ juice-shop, nous obtenons un accès.**

Figure:

##### Tâche 7 : Exposition de données sensibles

L'exposition des données sensibles se produit lorsque l'application n'a pas réussi à protéger les informations sensibles telles que les informations d'identification de l'utilisateur, les détails de la carte de crédit par exemple. Nous allons tenter « **Accéder à un document confidentiel et entrez le nom du premier fichier avec l'extension " .md "**

Lorsque nous avons terminé de parcourir dans le cadre de la **tâche :3**, nous avons observé lors de la vérification de la confirmation de commande, que l'URL contenait des informations ftp. Vérifions ftp s'il contient un fichier avec l’extension **.md**.

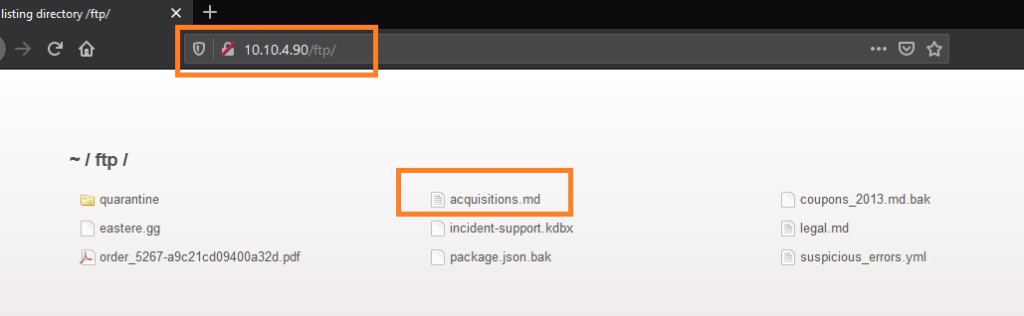


Figure:

On retrouve le fichier d'extension .md : « acquisitions.md »

##### **Tâche 7 : Contrôle d'accès interrompu**

Le contrôle d'accès brisé correspond aux vulnérabilités liées à l'autorisation de l'utilisateur, auxquelles l'utilisateur peut accéder, auxquelles il n'est pas autorisé à accéder. Ce type d'accès est appelé élévation verticale des privilèges, où l'utilisateur peut accéder à la page d'administration en manipulant la fonctionnalité de la page Web.

Ici, nous avons un total de 3 tâches, la première tâche est **"Accéder à la section d'administration du magasin"**

Nous aurons pu essayer avec « Administration », qui est l'une des approches courantes si les fonctionnalités des pages Web ne sont pas sécurisées ou non protégées. Ainsi n'importe qui peut facilement accéder aux fonctions administratives en naviguant directement vers l'URL appropriée, comme montré ci-après.

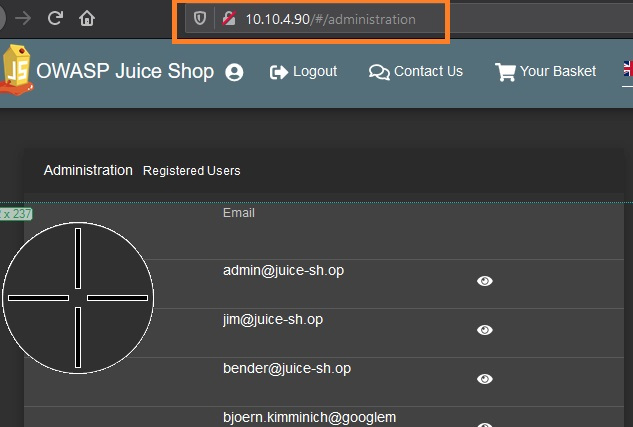


Figure:

La tâche suivante est " **Accéder au panier de quelqu'un d'autre** ".

En simulant une situation de shopping et nous voyons que sur la page de départ, en ouvrant l'inspecteur de stockage sous Développeur Web dans le navigateur Firefox Mozilla, nous avons la valeur « enchère ». Ici c'est "4",

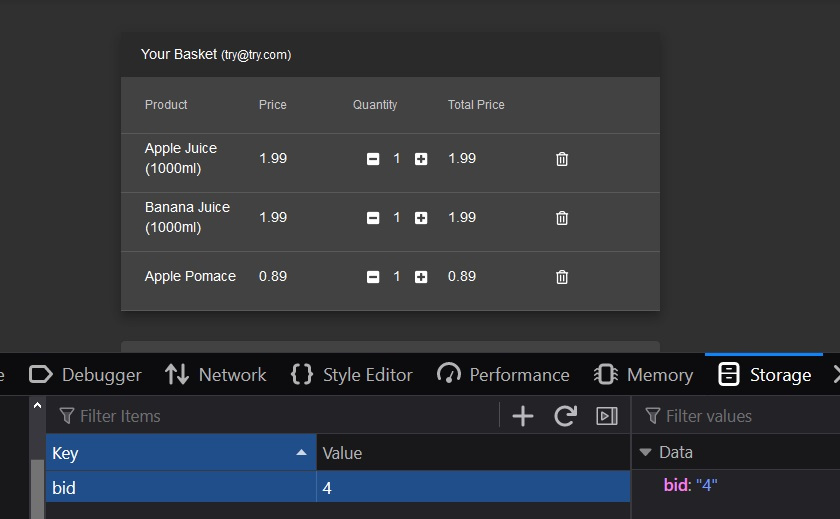


Figure:

En modifiant l'enchère de « 4 » à « 5 », l’actualisation de la page nous montre ce qui suit.

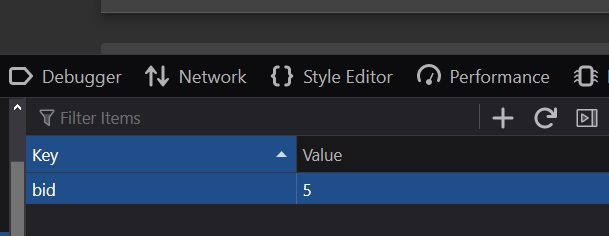


Figure:

Ainsi, nous avons réussi à accéder au panier de quelqu'un d'autre.

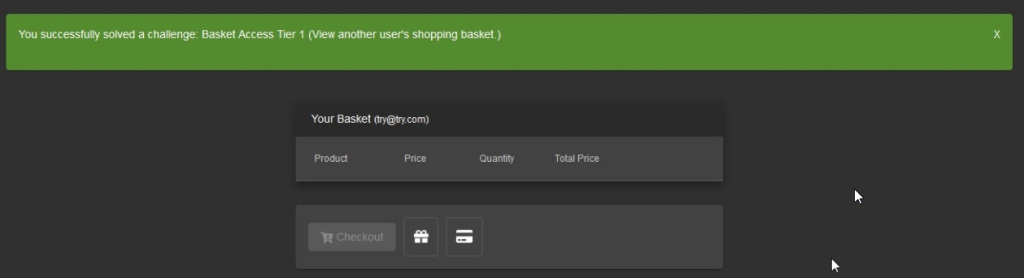


Figure:

La tâche suivante est **« Débarrassez-vous de tous les commentaires des clients 5 étoiles »**

Étant donne que nous avons déjà accédé à la page d'administration lors de la première tâche sous Broken Access Control, nous allons sur la même page. Et là nous avons l'avis client 5 étoiles, sur le côté droit, il y a une option pour supprimer l'avis en cliquant sur l'icône de la corbeille.

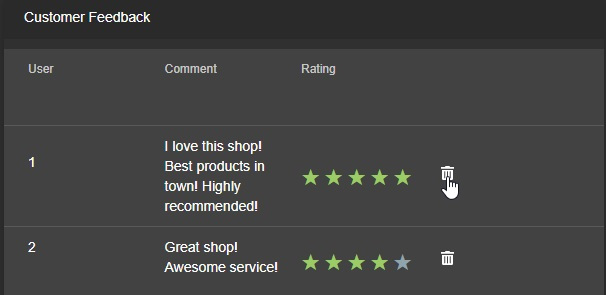


Figure:

Voici! Nous avons terminé avec succès les trois tâches du contrôle d'accès brisé.

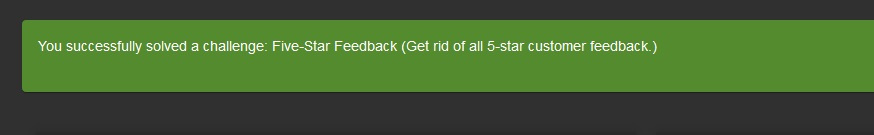


Figure:

##### Tâche 8 : Script inter-sites

Le Cross Site Scripting est une vulnérabilité de sécurité Web, qui permet à l'attaquant d'injecter un script côté client dans les pages Web utilisées par les utilisateurs.

Ici, nous allons effectuer deux taches en une, la première tâche est «**Exécuter le XSS en utilisant le suivi des commandes** ».

En nous connectant à n'importe quel compte, nous accédons suivi des commandes.

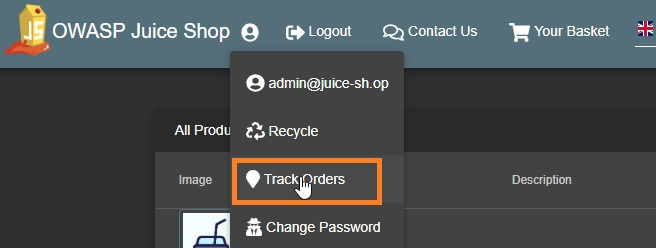


Figure:

Au lieu du numéro de commande, nous entrons un script comme indiqué ci-dessous.

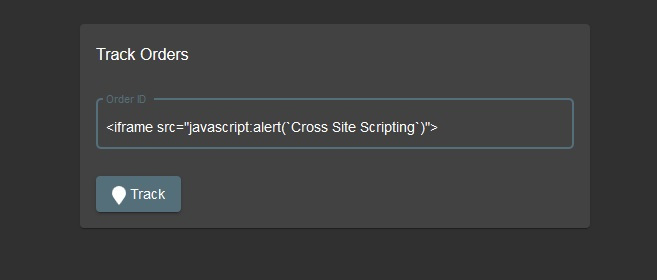


Figure:

Des vulnérabilités de script intersites reflétées surviennent lorsqu'une charge utile malveillante est injectée dans une requête et se répercute dans la réponse immédiate de l'application d'une manière dangereuse, comme indiqué ci-dessous,

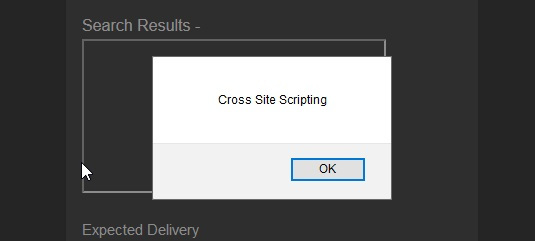


Figure:

Nous avons résolu la première tâche, passons maintenant à la deuxième tâche «**Exécuter XSS à l'aide du champ de recherche** ». Injectez maintenant la charge utile dans le champ de recherche comme indiqué ci-dessous,

La charge utile est " **<iframe src="javascript:alert(`XSS@SearchBox`)">** "

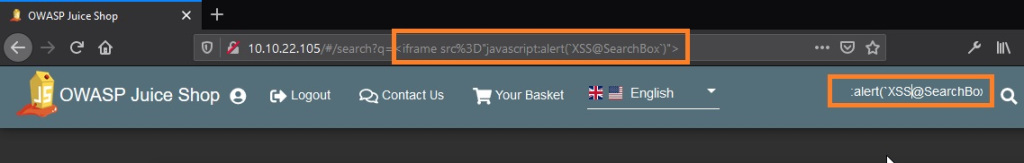


Figure:

Et nous avons comme résultat :

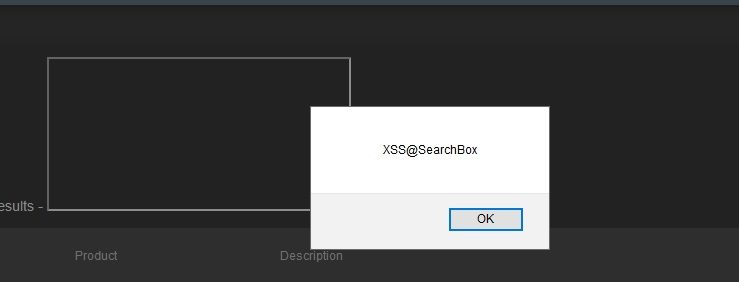


Figure:

A l’issue de cet audit de sécurité, nous avons examiné certaines des vulnérabilités courantes de l'application Web OWASP Juice. De même, il y en a beaucoup. Nous pouvons donc passer à la dernière phase de notre travail avec un rapport des résultats obtenus.

## Chapitre 4 : Mesures correctives, rapport

#### SECTION 1 : Mesures correctives

1.Évaluation des risques

Après la réalisation de notre pentest, nous devons à présent présenter le résultat de votre travail. Pour cela, nous allons effectuer une mise en forme et proposer des solutions pour la correction des vulnérabilités trouvées comme le prévoit le pentesting.

Pour chaque vulnérabilité trouvée, nous devons en évaluer le risque. Ce n’est pas systématique, mais dans les rapports de test d’intrusion très complet, il importe de réaliser évaluation du risque découlant d’une classification suivant trois axes.

1. **La sévérité**

Dans le domaine de la sécurité informatique, le système d'évaluation standardisé de la criticité des vulnérabilités - ou « Common Vulnerability Scoring System » (CVSS) - se base sur des critères objectifs et mesurables. Cette évaluation est constituée de 3 mesures appelées métriques :

* la métrique de base,
* la métrique temporelle,
* la métrique environnementale.

Ce standard CVSS multiplie la probabilité d’apparition d’un événement avec le niveau de gravité des conséquences de cet événement. La composante qui en résulte est la sévérité, ou criticité. Celle-ci est évaluée et adaptée au contexte de l’entreprise par le pentesteur.

1. **La complexité de la correction**

Pour chaque vulnérabilité identifiée, nous devons proposer une solution corrective et estimer le niveau de difficulté de correction, ainsi qu’il suit :

* Complexe : La correction de la vulnérabilité identifiée nécessite une modification importante dans l’organisation ou le code d’une application. Elle est donc très coûteuse en investissement ou en ressources.
* Modérée : La correction de la vulnérabilité identifiée nécessite une modification limitée dans l’organisation ou le code d’une application. Elle est donc moyennement coûteuse en investissement ou en ressources.
* Facile : La correction de la vulnérabilité identifiée nécessite une simple modification (par exemple : un fichier de configuration, changement de mot de passe). Elle est donc peu coûteuse en investissement ou en ressources.

Par exemple lorsqu’un mot de passe simple est laissé par défaut après la mise en production d'une application. La complexité de la correction est « facile » puisqu'il suffit de changer le mot de passe pour sécuriser l'application.

1. **La priorité de la correction**

Partant du constat de ceux qui précédent (sévérité et complexité), nous pourrons conseiller sur la priorisation de l’application des solutions correctives au niveau du SaaS. Nous pourrons alors avoir :

* Priorité 1 -> Urgent -> La correction de la vulnérabilité doit être faite dans les plus brefs délais.
* Priorité 2 -> Standard -> La correction de la vulnérabilité peut suivre le processus habituel de traitement.
* Priorité 3 -> Bas -> La correction de la vulnérabilité n’est pas urgente et reste à la convenance de vos équipes informatiques.

Ces trois dimensions devront être trouvées pour chaque vulnérabilité dans le rapport de test d'intrusion. Aussi, si nous reprenons exemple cite ci-avant, pour un mot de passe laissé par défaut après la mise en production d'une application, nous avions une complexité de correction « facile ». Or dans le cas d’un compte administrateur, il y a des droits étendus, la sévérité sera donc « intolérable ». Car logiquement, la préconisation d’un pentesteur sera une « priorité 1 » : « Urgent » pour l’application de la correction.

1. Les mesures correctives

La majorité, sinon toutes les vulnérabilités connues sont référencées dans la base de données CVE (Common Vulnerabilities and Exposures). Les détails de chacune d’elles y sont disponibles et accessible gratuitement sur leur site.

Chaque vulnérabilité est présentée dans un tableau, qui présente le détail des systèmes touchés et la façon dont on peut l’exploiter.

*Tableau : Model de présentation des vulnérabilités*

|  |  |
| --- | --- |
| **CVE ID et CWE ID** | C'est le code unique donné à chaque vulnérabilité découverte dans le monde informatique. |
| **Vulnerability Type(s)** | Dans cette colonne du tableau, nous avons l'information sur le type d'exploitation de la vulnérabilité. Cela peut être une exécution de code arbitraire, une exécution de code SQL, une attaque XSS, une attaque DoS, une attaque de type overflow, etc. |
| **Date** | Nous avons également l'information sur la date où la vulnérabilité a été rendue publique et la date de mise à jour. |
| **Score** | C'est le système d'évaluation dont je vous parlais tout à l'heure, avec une note CVSS qui est donnée pour évaluer la sévérité de la vulnérabilité. |
| **Gained Access Level** | Ce champ indique s'il est possible d'obtenir des accès grâce à cette vulnérabilité. Cela peut-être des accès système, utilisateur, ou administrateur. |
| **Access** | Dans cette colonne, nous avons une précision sur le type d'accès qui peut être local ou distant. |
| **Complexity** | Ici, c'est la complexité et donc la technicité qu'il est nécessaire de mettre en œuvre pour exploiter cette vulnérabilité. |
| **Conf. + Integ. + Avail** | Ces champs indiquent le niveau d’impact sur la confidentialité, l'intégrité et la disponibilité. |

Ces champs indiquent le niveau d’impact sur la confidentialité, l'intégrité et la disponibilité.

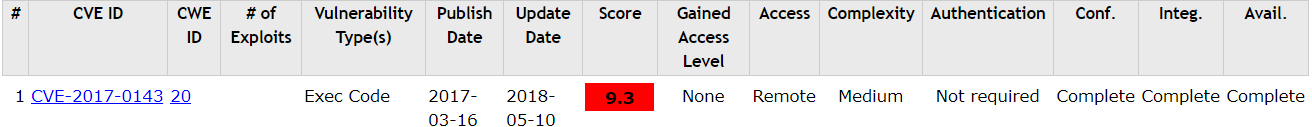


Figure : exemple de détails d’une vulnérabilité

Ainsi, dans un rapport de test d'intrusion, il doit apparaître la liste des vulnérabilités trouvées, la façon dont elles peuvent être exploiter, et une solution de correction. Aussi, une vulnérabilité sera classée selon les axes de la sévérité et de la complexité, afin de définir la priorité de la correction.

En outre, la correction des vulnérabilités terminée, on devrait prévoir un deuxième test d'intrusion complet pour s'assurer de la réussite de la correction. D’ailleurs, pour à l’obtention de certaines normes comme PCI DSS ou ISO27001.b, le test d’intrusion est très souvent nécessaire.

#### SECTION 2 : Rapport

TABLE DES MATIÈRES………………………………………………………………... 1

RÉSUMÉ…………………………………………………………………………………. 2

JOURNAL DE NARRATION ET D'ACTIVITÉ………………………………………… 3

CONSTATATS ET RECOMMANDATIONS…………………………………………… 7

COTES DE RISQUE………...………………………………………………………... 7

RÉSUMÉ DES CONSTATATIONS……...…………………………………………... 8

CONSTATS DE RISQUE CRITIQUE………………………………………………... 8

1. Défauts d'injection SQL…………………………………………………….. 8

2. Contournement d'autorisation

CONSTATS À HAUT RISQUE………………….………………………………….. 13

3. Défauts de script inter-sites………………………………………………… 13

CONSTATS À RISQUE MOYEN………….……………………………………….. 16

4. Normes de sécurité inadéquates pour le stockage des mots de passe………. 16

CONSTATAS À FAIBLE RISQUE…………………………………………………. 17

5. Exigences de complexité des mots de passe faibles………………………... 17

ORIENTATION STRATÉGIQUE……………………………………………………… 19

RÉSUMÉ

Nous avons effectué un test de pénétration de l'application Web de OWASP Juice Shop. La portée de cette évaluation était http://juice-shop.wtf.

Le graphique suivant montre le nombre de constats par risque pour ce rapport :

*Tableau :*

|  |  |  |  |
| --- | --- | --- | --- |
| **Critique** | **Haut** | **Moyen** | **Bas** |
| 2 | 1 | 1 | 1 |

Sur la base des conclusions de ce rapport, nous avons évalué que le risque global pour Juice Shop par rapport à la portée de cet engagement est très élevé :

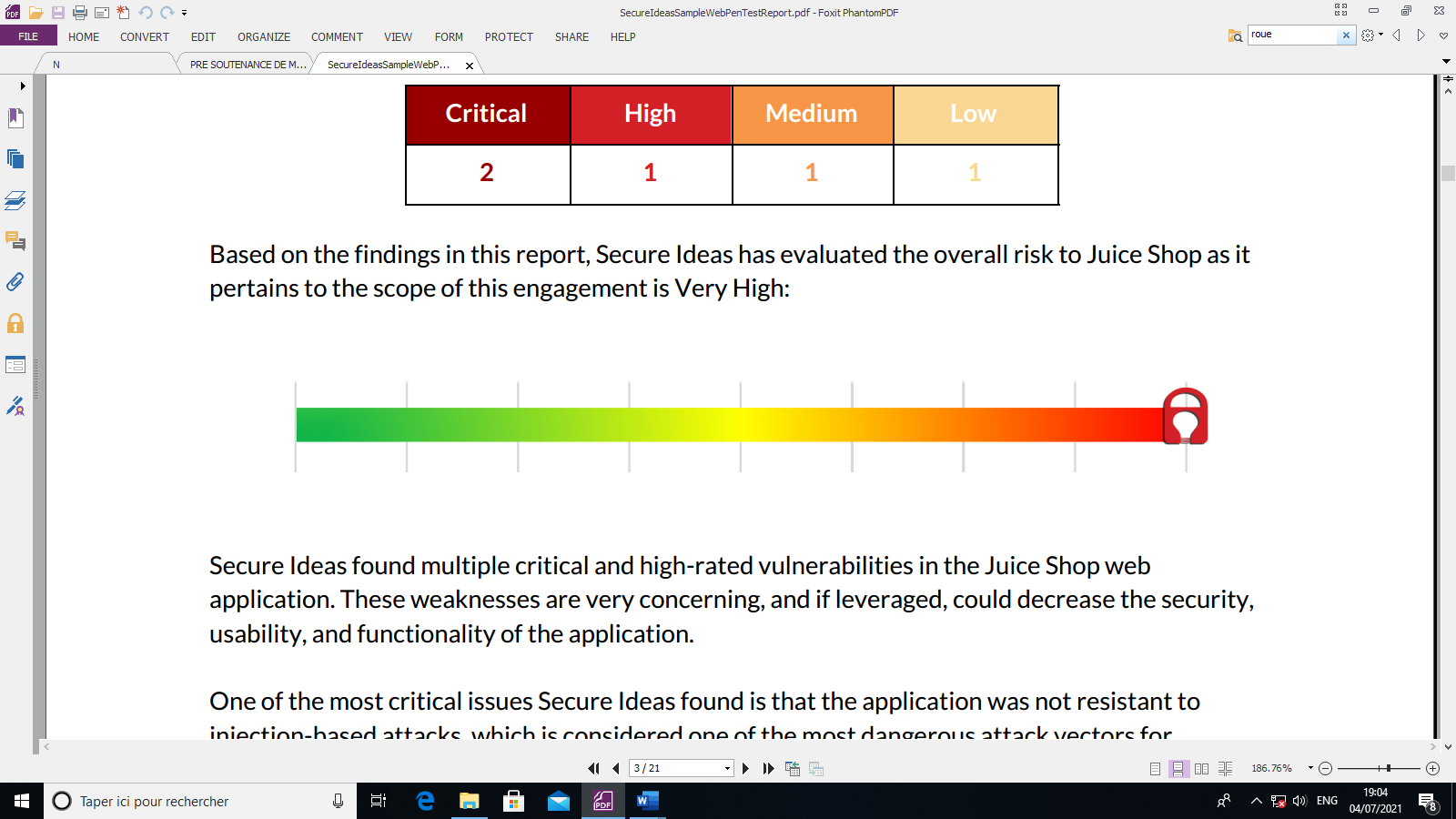


Figure:

Nous avons trouvé et bien notées plusieurs vulnérabilités critiques sur l’application Web Juice Shop. Ces faiblesses sont très préoccupantes et, si elles sont exploitées, pourraient diminuer la sécurité, la convivialité et la fonctionnalité de l'application.

L'un des problèmes les plus critiques que nous avons trouvé est dans cette application n'était pas résistante aux les attaques par injection, qui sont considérées comme l'un des vecteurs d'attaque les plus dangereux pour applications. Un exemple démontrant la gravité de cette vulnérabilité était notre capacité à élaborer une chaîne d'injection SQL simple (SQLi) pour se connecter en tant qu'administrateur de l'application, sans aucune connaissance préalable du nom d'utilisateur ou des informations d'identification. Il y a également eu plusieurs cas de Cross-Site scripting (XSS) dans toute l'application. Les failles SQLi et XSS peuvent facilement être corrigés grâce à un assainissement d'entrée et un crytage en sortie cohérents et sont discutés plus en détail dans la section de decouverte ci-dessous.

Un autre problème important découvert est une faille de contournement d'autorisation. Nous avons constaté qu'un l'attaquant peut utiliser un API pour créer un nouvel utilisateur avec n'importe quel rôle, y compris un accès administrateur. C'est en raison d'un manque de vérification des autorisations au sein de l'API. La cohérence est importante dans l'autorisation validation, et l'application doit l'appliquer à toutes les interfaces pour empêcher la non-protection.

Ces problèmes, ainsi que les autres problèmes détectés, sont décrits dans le rapport qui suit. Nous avons apprécié l'opportunité de travailler avec cette entreprise pour l'aider à améliorer sa sécurité.

* + 1. JOURNAL DE NARRATION ET D'ACTIVITÉ

Nous avons commencé par travailler avec notre méthodologie standard de Recon → Mapping → Découverte → Exploitation. Comme cette évaluation n'était pas une évaluation boîte noire, l'équipe a sauté la phase de reconnaissance initiale, en commençant par la cartographie et la découverte. Tout au long de l’engagement, nous avons mené plusieurs types d'activités sur chacune des interfaces Web au sein de l’application juice shop. La liste suivante détaille les activités et considérations de haut niveau effectuée lors de l'engagement. Cette liste n'inclut pas tous les tests effectués.

Réaliser la cartographie de l'application dans le périmètre

* Évalué pour les failles Web courantes telles que :
  + Authorization bypasses
  + JSON Web Token (JWT) manipulation
  + Cross-Origin Resource Sharing (CORS) misconfigurations
  + Cross-Site Request Forgery (CSRF)
  + Testing for Server-Side Request Forgery (SSRF)
  + Ineffective / misconfigured security controls
  + Injection flaws such as Cross-Site Scripting (XSS) and SQL Injection (SQLi)
  + Fuzzing of HTTP header values
  + Testing for HTTP Desync and Cache poisoning flaws
  + Fuzzing of query and body parameters
  + Client side JavaScript static and dynamic analysis
* Tester d'autres éléments à haut risque, y compris toutes les vulnérabilités testables répertoriées dans l'OWASP Top 10

Dans le cadre de la phase de cartographie, nous avons exploré toutes les fonctionnalités disponibles dans l'application en utilisant chaque rôle de compte fourni. En commençant en tant qu'utilisateur normal, nous avons commencé à construire une carte de la les fonctionnalités de l'application et les zones potentiellement vulnérables, telles que les formulaires de connexion, les pages de profil utilisateur ou champs de saisie pour les informations sensibles. Ce processus a également été répété dans le cadre d'une procédure administrative compte, et les différences de rôles et d'autorisations d'accès ont été notées. En cartographiant minutieusement l'application et les rôles, nous avions développé une bonne idée de la fonctionnalité et des fonctionnalités utilisées dans l'application, ainsi que la façon dont l'application était censée se comporter.

Ensuite, nous avons de nouveau parcouru l'application, mais cette fois du point de vue d'un utilisateur malveillant ou attaquant. Au lieu de considérer les actions attendues d'une application normale utilisation, nous avons appliqué diverses techniques liées à l'interception/manipulation des demandes sortantes ou réponses entrantes, en passant des données mal formées aux champs de saisie et en essayant de générer des réponses de la demande.

Lors de l'examen du formulaire de connexion, nous avons remarqué que les guillemets simples pouvaient être utilisés pour provoquer des erreurs sur le page, ce qui indique généralement une mauvaise gestion des entrées. Des sondages supplémentaires ont montré que l’Email champ est sensible à l’injection SQL. En entrant dans la chaîne de caractères « 'ou 1 = 1--» dans l’Email champ, ainsi que toute valeur dans le mot de passe champ, provoqué l'application pour évaluer l'état de connexion comme vrai. Cela nous a authentifiés en tant que première entrée dans la table de base de données Users, qui est la compte administrateur. Les résultats peuvent être vus dans la capture d'écran ci-dessous qui représente le profil de l'utilisateur actuellement connecté.

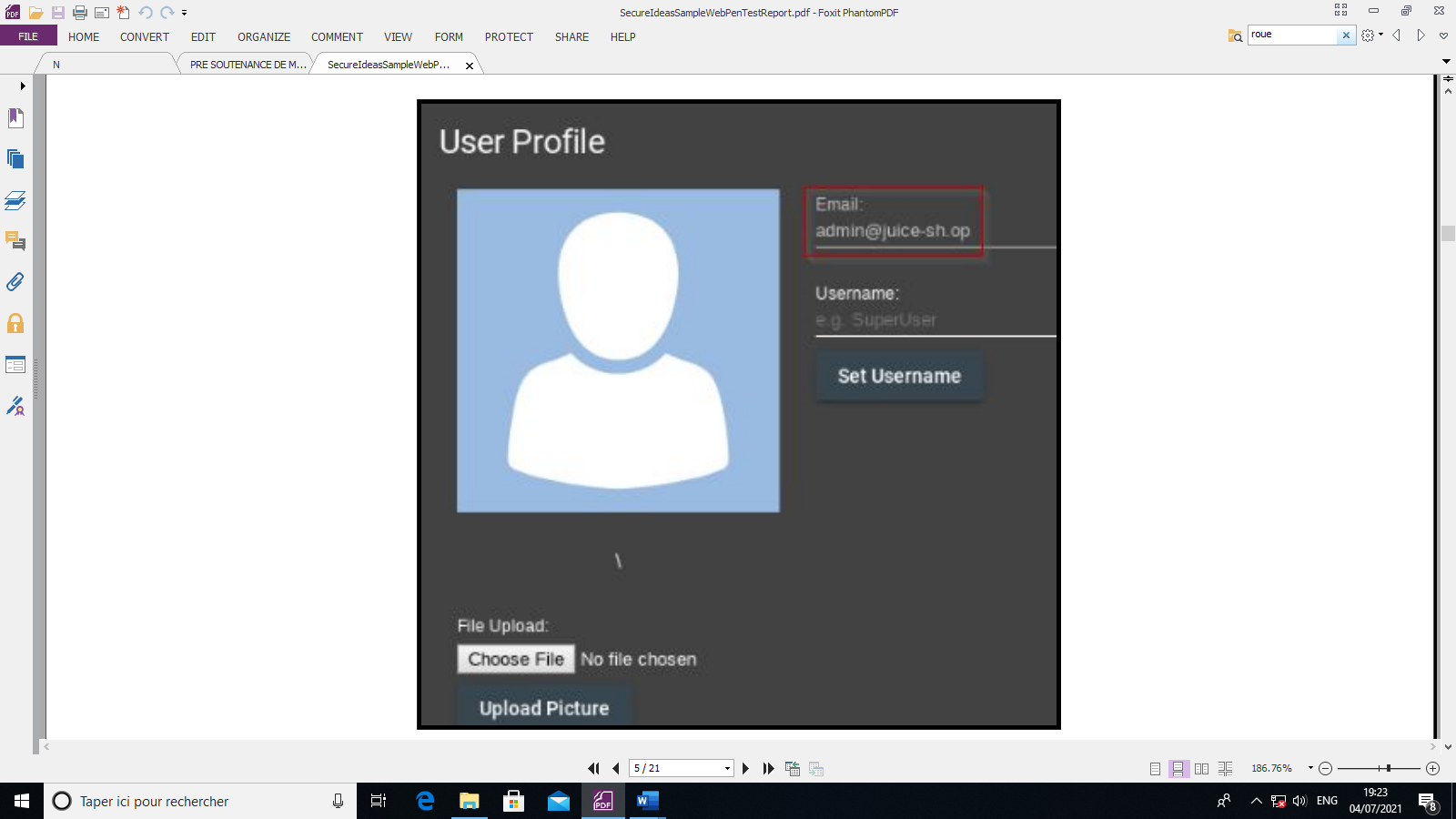


Figure :

Après avoir observé que ce champ peut être utilisé pour injecter différents types de commandes SQL, nous avons commencé expérimenter différentes requêtes pour voir quelles autres informations pourraient être obtenues dans ce manière. Nous avons découvert qu'en appliquant une légère modification à la requête, nous avons pu enregistrer dans le compte suivant de l'application. En utilisant une requête telle que, 'ou 1 = 1 et email non comme (« % admin% ») -, nous avons pu filtrer l’administration compte, le déplacement du « pointeur de connexion » à la compte suivant dans la base de données qui ne contient pas la chaîne admin. Faire preuve de créativité, et cette requête comme modèle de base, un attaquant pourrait éventuellement énumérer la table Users pour récolter chaque nom d'utilisateur qu'il contenait.

En raison de la criticité du SQLi, qui permet à un utilisateur non authentifié de contourner le login processus d'autorisation, nous avons rapidement contacté l’entreprise. Une brève une explication de la faille a été fournie, en désinfectant toute information spécifique qui ne devrait pas être envoyée par courrier électronique non sécurisé, et une réunion a été demandée pour examiner les conclusions découvertes dans l’application de la boutique de jus.

En attendant la réponse de l’entreprise, nous avons continué nos tests de l'application. Pendant ce temps, un autre problème important a été découvert, qui aggrave le risque associé au SQLi défaut noté ci-dessus. Après s'être connecté au compte de l'utilisateur suivant dans la base de données, nous avons pris quelques le temps d'inspecter la fonctionnalité « modifier le mot de passe ». Par défaut, le bouton de changement utilisé pour mettre à jour un mot de passe de l’utilisateur est activé uniquement lorsque les champs « le mot de passe actuel », « nouveau mot de passe », et « répéter nouveau mot de passe » sont correctement remplis. Cependant, nous avons découvert qu'en utilisant le f12 du navigateur outils de développement, le mot de passe d'un utilisateur peut être mis à jour sans connaître le mot de passe actuel. Ceci a été accompli en manipulant la page web de soumission. Comme on le voit ci-dessous, lors du changement de « mat-raised-button mat-primary de « désactivé » à « activé », le changement de mot de passe peut être traité par l'application, sans passer par le champ « mot de passe actuel ».

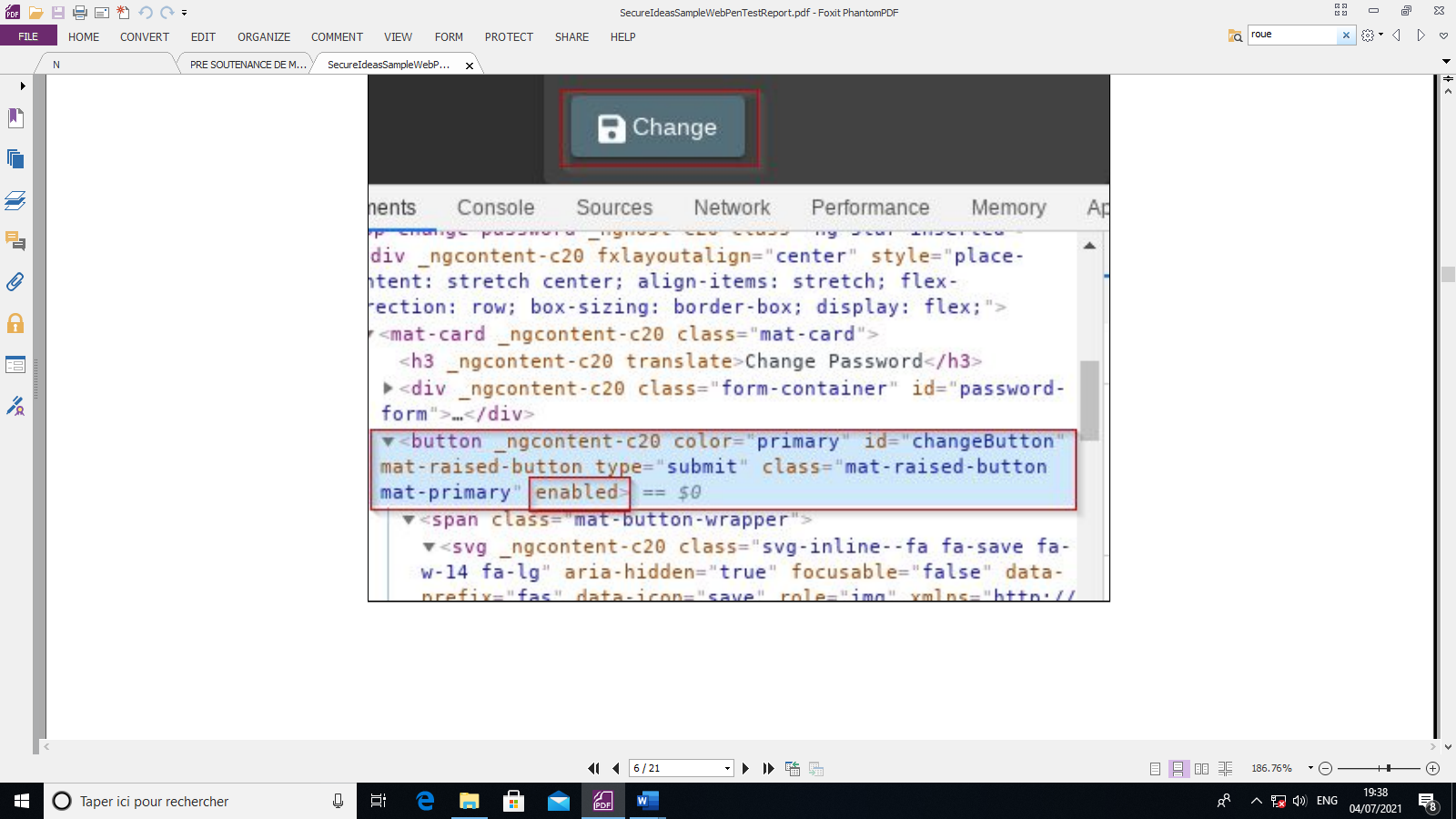


Figure :

En poursuivant avec les API testées dans cet engagement, nous avons constaté qu'un utilisateur non authentifié pourrait soumettre une simple requête POST et créer des utilisateurs avec des privilèges administratifs. Un exemple demande peut-être vu dans le bloc de code ci-dessous :

*Tableau :*

POST /api/Users HTTP/1.1

Host: juice-shop.wtf

Content-Type: application/json

Connection: close

Content-Length: 82

{"email":"admin","password":"admin","role":"admin"}

Dans cette demande, aucune authentification ou jeton n'est fourni, et aucune valeur de cookie n'est fournie. La seule exigence trouvée pour la création d'un compte administratif est le « Content-Type : application / json header », et quelques drapeaux de compte de base ajoutés dans le corps de la demande. Comme montré dans la capture d'écran suivante, cette requête POST a été utilisée pour créer avec succès un administrateur compte utilisateur, lui donnant l'administrateur rôle.

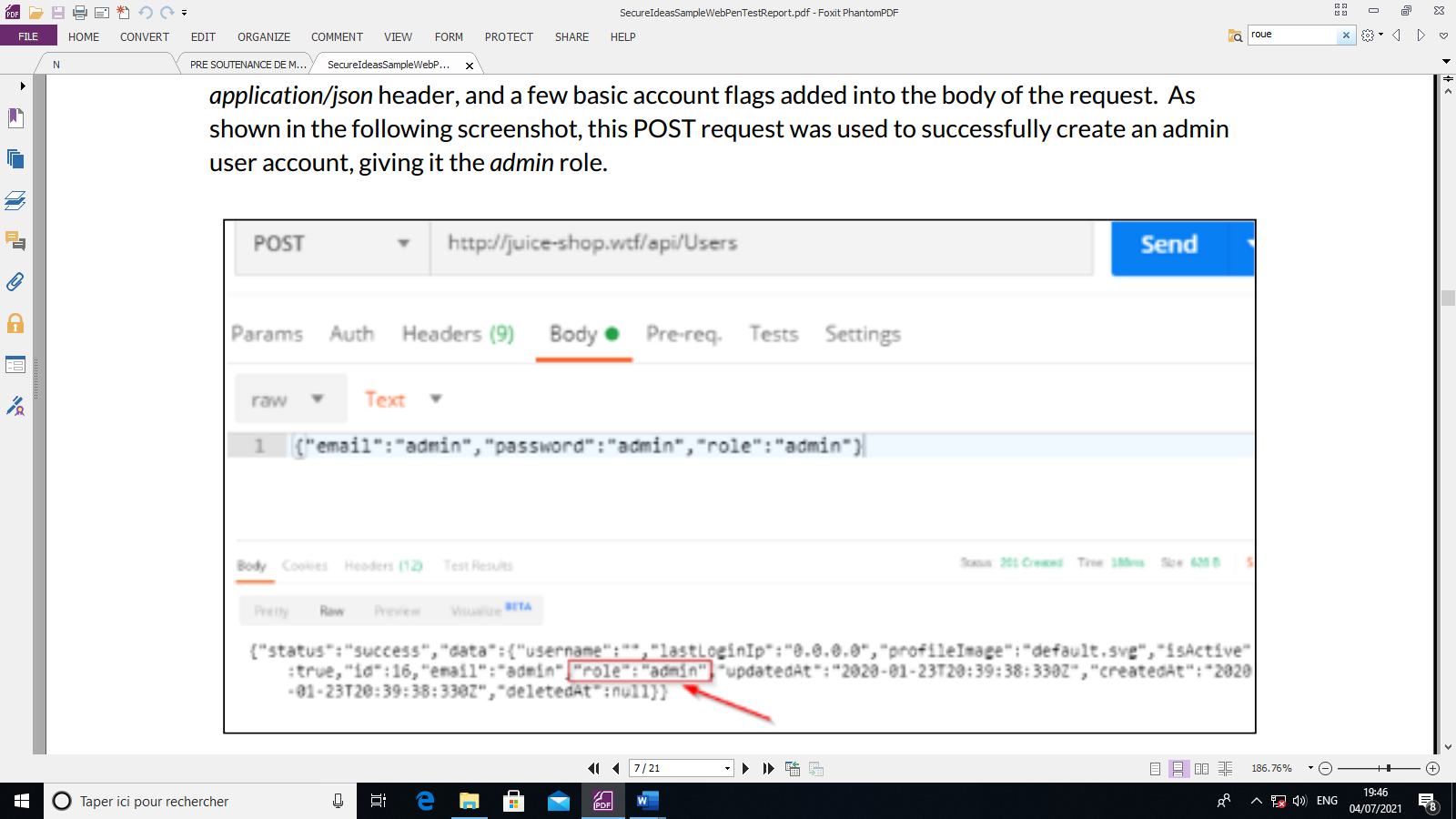


Figure :

En raison de la nature critique de ce contournement d'autorisation, un autre e-mail a été envoyé à l’entreprise. Une réunion a été organisée pour passer immédiatement en revue ces conclusions et obtenir des directives supplémentaires sur considérations pour le reste de ce test. Nous avons ensuite rencontré l'équipe technique de l’entreprise à propos des découvertes critiques découvertes, et discuté de diverses options liées au risque et l'assainissement de ces éléments. Elle n'a pas tardé à répondre, et a déjà commencé une analyse des causes premières pour déterminer la meilleure façon de mettre en œuvre les correctifs conformément aux politiques et processus internes de l’entreprise. Des efforts supplémentaires seront consacrés à l'examen et à la détermination de la meilleure façon de résoudre les problèmes de sécurité tout au long du processus de développement de l'application.

Ces problèmes, ainsi que les autres problèmes découverts, sont décrits dans le rapport qui suit.

* + 1. CONSTATATIONS ET RECOMMANDATIONS

Ce rapport décrit nos conclusions collectées à partir des tests, ainsi que nos recommandations qui aideront l’entreprise à réduire ses risques et éliminer les vulnérabilités trouvées.

1. **Côté Risque**

Chaque résultat est classé en tant que risque critique, élevé, moyen ou faible en fonction de notre jugement professionnel et notre expérience. Pour déterminer le risque, nous prenons en compte chacun des aspects suivants :

* Menaces potentielles : cela comprend une évaluation des acteurs potentiels de la menace et le niveau de compétence
* Probabilité d'attaque : les considérations incluent les motivations de l'attaquant, la complexité de vecteur d'attaque et atténuation potentielle des contrôles de sécurité
* Impact possible​ : Pour chaque résultat, nous considérons les dommages potentiels à l’organisation résultant d'une attaque réussie

Chacun de ces facteurs est évalué individuellement et en combinaison pour déterminer le risque global la désignation. Ces évaluations sont basées sur notre jugement professionnel et notre expérience. Le niveau de risque suivant les descriptions, démontre les types de vulnérabilités désignées dans chaque catégorie.

**a.1 Critique**

Les vulnérabilités trouvées qui sont activement exploitées dans la nature et sont connues pour conduire à des exploitations par des attaquants externes. Ces failles de sécurité sont susceptibles d'être ciblées et peuvent avoir un impact important sur l'entreprise. Ceux-ci nécessitent une attention immédiate sous la forme d’une solution de contournement ou protection temporaire. Lorsqu'une est découverte, nous arrêtons immédiatement tout test et contactons le client pour obtenir des instructions supplémentaires. Des exemples de ceci peuvent inclure des systèmes externes avec des exploits d'exécution de code à distance connus ou des interfaces d'accès à distance avec des informations d'identification faibles ou par défaut.

**a.2 Haute**

Vulnérabilités découvertes pouvant conduire à une exploitation par des attaquants internes ou distants. Ces failles de sécurité sont susceptibles d'être ciblées et peuvent avoir un impact significatif sur l'entreprise. Ces défauts peuvent nécessiter une attention immédiate pour une protection temporaire, mais nécessitent souvent une protection plus systémique changements dans les contrôles de sécurité. Quelques exemples incluent des défauts d'injection de commande, l'utilisation de la fin de vie logiciel et les informations d'identification par défaut.

**a.3 Moyen**

Vulnérabilités ou services trouvés qui pourraient contribuer indirectement à un incident plus important ; ou sont directement exploitables dans une mesure quelque peu limitée en termes de disponibilité et/ou d’impact. Cette classe de vulnérabilité est peu susceptible de conduire à elle seule à un compromis significatif, cependant peut poser un danger substantiel lorsqu'il est combiné avec d'autres. Quelques exemples incluent faible la sécurité de la couche de transport sur une transaction sensible, une segmentation du réseau insuffisante ou l'utilisation de bibliothèques de logiciels vulnérables.

**a.4 Faible**

Des vulnérabilités ou des services qui, trouvés seuls, ne sont pas directement exploitables et présentent peu de risque, mais peuvent fournir des informations qui facilitent la découverte ou l'exploitation réussie d'autres défauts. Les exemples incluent la divulgation des versions du logiciel serveur et les messages de débogage.

**a.5 Résumé des constats**

Le tableau suivant résume les résultats. Chaque résultat est détaillé par risque immédiatement après le tableau récapitulatif.

*Tableau :*

|  |  |
| --- | --- |
| **Découverte** | **Risque** |
| 1. SQL Injection Flaws | Critique |
| 2. Authorization Bypass | Critique |
| 3. Cross-Site Scripting Flaws | Haute |
| 4. Inadequate Security Standards for Password Storage | Moyen |
| 5. Weak Password Complexity Requirements | Faible |

**b. Constats de risque critique**

**b.1 SQL Injection Flaws**

*Tableau :*

|  |  |
| --- | --- |
| **Normes de l'industrie** |  |
| Top 10 de l'OWASP | A1 : injection |
| NIST 800-53 | SI-10 : Validation de la saisie des informations |

**b.1.1 Résumé**

Lorsque des données entrent dans une application Web sans être correctement nettoyées, elles peuvent exposer l’application à plusieurs catégories de vulnérabilités. L'une de ces catégories est l'injection de langage de requête structuré (SQL) par un tiers. Ce type d'attaque est communément appelé injection SQL.

L'injection SQL se produit lorsque des données sont insérées ou ajoutées dans un paramètre d'entrée d'application, et cette entrée est utilisée pour construire dynamiquement une requête SQL. Lorsqu'une application Web ne fonctionne pas correctement désinfecter les données, qui sont transmises pour créer dynamiquement des instructions SQL, il est possible pour un attaquant pour modifier la construction des instructions SQL back-end.

Certains des risques potentiels comprennent :

* Perte de données sensibles ou confidentielles
* Données sensibles ou confidentielles altérées
* Contournement de l'authentification
* Contournement d'autorisation
* Accès au système d'exploitation sous-jacent
* D'autres attaques contre les utilisateurs de l'application (XSS, CSRF)

Une façon d'exploiter ce type de vulnérabilité est via Blind SQL Injection. L'injection SQL aveugle est identique à une attaque standard par injection SQL, sauf que lorsqu'un attaquant tente d'exploiter une application, plutôt que d'obtenir un message d'erreur utile, l'attaquant obtient à la place une page générique spécifié par le développeur. Cela rend plus difficile l'exploitation d'une attaque potentielle par injection SQL mais pas impossible. Un attaquant peut toujours accéder aux données en demandant une série de Vrai et Faux questions via des instructions SQL.

**b.1.2 Découverte**

Nous avons découvert que la page de connexion de l'application Juice-shop est vulnérable à l’Injection SQL. Cela est dû à l'utilisation d'entrées non nettoyées fournies par l'utilisateur. En utilisant le paramètre de « l '= OR 1 = 1-- », comme le nom d’utilisateur et le mot de passe , nous avons été capable de se connecter comme administrateur.

Étant donné qu'Admin était le premier utilisateur répertorié dans l'application, il a donc été utilisé afin comme charge utile. Ce que le montrent les captures d’écran ci-dessous, avec le compte administrateur indiqué en premier dans l’application. Des comptes supplémentaires pourraient être accessibles en utilisant « ’ ou 1 = 1 » et un e - mail différent de « (% admin%) » ; « -- » etc .

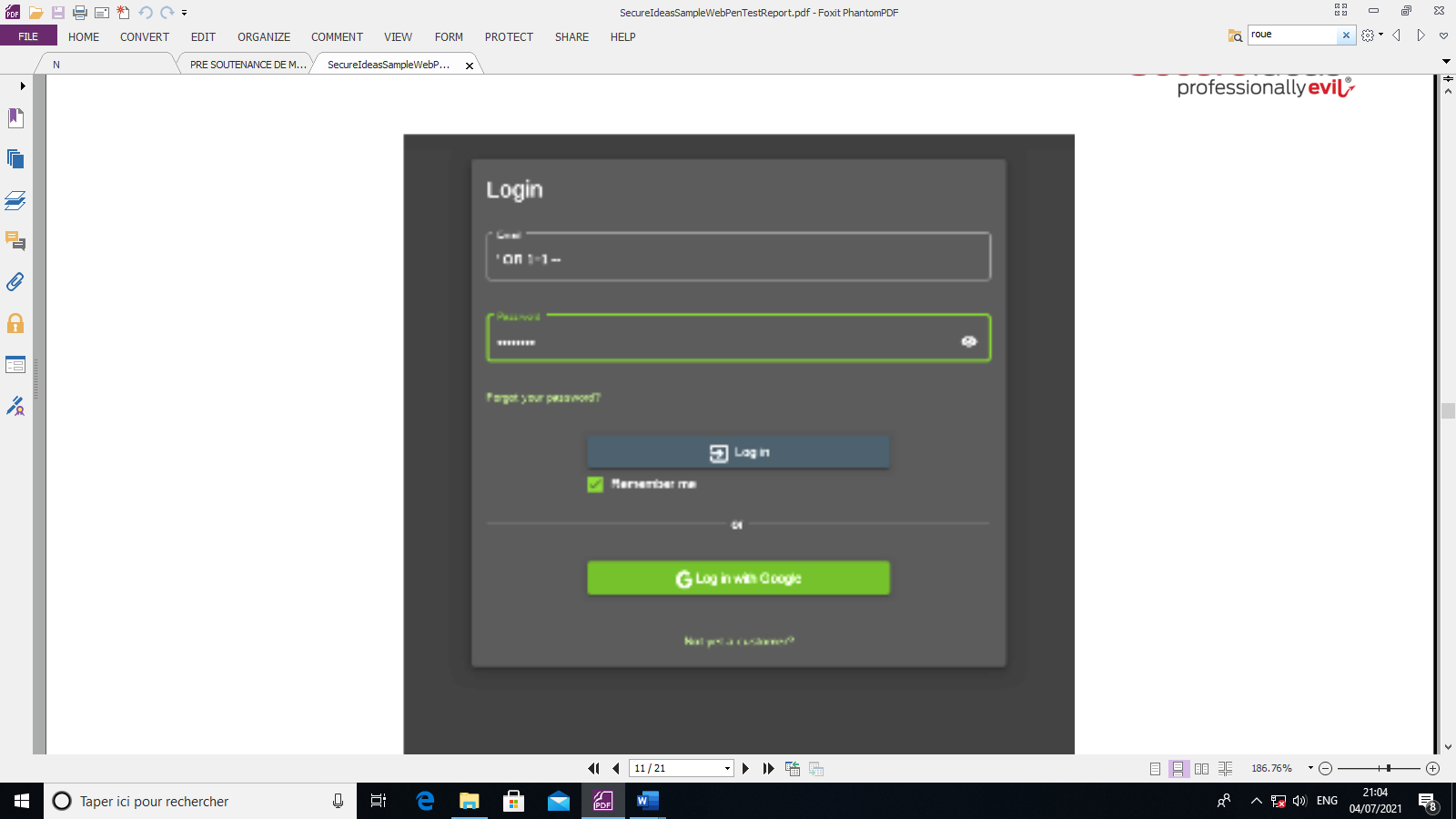


Figure :

**b.1.3 Recommandations**

Nous recommandons à lentreprise d'utiliser des requêtes paramétrées lors de l'interaction avec une base de données en arrière-plan. Les requêtes paramétrées sont une méthode où la requête est créée dans le code de l'application sans les valeurs nécessaires. Les espaces réservés sont utilisés et remplacés lors de l'exécution par les valeurs du l'utilisateur ou la transaction. Les requêtes actuellement paramétrées sont la meilleure protection contre les attaques d'injection SQL.

Si, pour une raison quelconque, les requêtes paramétrées ne sont pas possibles, nous recommandons à l’entreprise d’effectuer une validation en entrée pour empêcher cette forme d'attaque. Les développeurs doivent s'assurer que l'application valide que l'entrée de l'utilisateur est exactement du type souhaité par le développeur. Par exemple, si Juice Shop n'attend que des caractères alphanumériques dans l'entrée, alors l'application doit effectuer la saisie filtrage pour rejeter toute autre chose. Ceci est considéré comme une approche de liste blanche.

De plus, nous recommandons que l’entreprise gère correctement toutes les instructions et commandes SQL dans le code afin que les messages d'erreur du SGBD ne soient pas renvoyés directement au navigateur.

Les développeurs de Juice Shop peuvent également utiliser une bibliothèque de sécurité commune pour effectuer un filtrage en entrée et cryptage en sortie. Les implémentations doivent suivre les meilleures pratiques OWASP pour prévenir cette vulnérabilité, que l’entreprise choisisse ou non d'utiliser une bibliothèque pour ces tâches (<https://owasp.org/www-project-cheat->sheets/cheatsheets/SQL\_Injection\_Prevention\_Cheat\_Sheet.html).

**b.2 Authorization Bypass**

*Tableau :*

|  |  |
| --- | --- |
| **Normes de l'industrie** |  |
| Top 10 de l'OWASP | A5 : Contrôle d'accès cassé |
| NIST 800-53 | Application de l'accès AC-3 |

**b.2.1 Résumé**

Le contournement d'autorisation est une faille dans le logiciel ou un trou dans la planification de la sécurité où un utilisateur ou un attaquant est en mesure d'accéder à des données ou à des fonctionnalités pour lesquelles l'utilisateur n'est pas autorisé. Cette vulnérabilité n'exige pas qu'un attaquant malveillant augmente le risque pour une entreprise ; le simple le fait que des utilisateurs non autorisés aient accès à une infrastructure commerciale augmente les risques pour l’entreprise. Le problème central du contournement d'autorisation est le manque de validation au sein de l'application.

Lorsque l'application Web accepte l'entrée d'un utilisateur et utilise cette entrée pour récupérer des données ou fournir l'accès, il est essentiel que l'application valide que l'utilisateur a réellement l'autorisation d’effectuer cette action. Lorsque cette validation n'a pas lieu, ou peut être trompée, l'application est vulnérable aux attaques.

Les risques auxquels les entreprises sont confrontées en cas de contournement d'autorisation incluent l'introduction de bogues dans le code via les erreurs des utilisateurs, un attaquant accédant aux parties administratives de l'application ou la perte de données commerciales importantes à un voleur de données.

**b.2.2 Découverte**

Nous avons découvert que l'application Juice shop contient une faille de contournement d'autorisation. Pendant les tests, nous avons pu créer un compte administrateur avec une session non authentifié.

Dans l'API Juice Shop, nous avons découvert qu'un attaquant ou un utilisateur malveillant pouvait créer un nouvel utilisateur avec le rôle d'administrateur.

La description suivante explique comment nous avons pu effectuer cette attaque.

* Créer une requête Post dans Postman API testing tools sur <https://juice>-shop.wtf/api/Users
* Ajoutez une ligne dans le corps de la demande en utilisant l'instruction suivante {« e-mail » : « admin », « mot de passe » : « admin », « » » « » » admin"}
* Envoyer la demande au point de terminaison de l'API
* Visitez la page de connexion pour vous connecter en utilisant un nouveau compte administrateur

Comme indiqué ci-dessous, le nouvel utilisateur a été créé avec des privilèges administratifs :

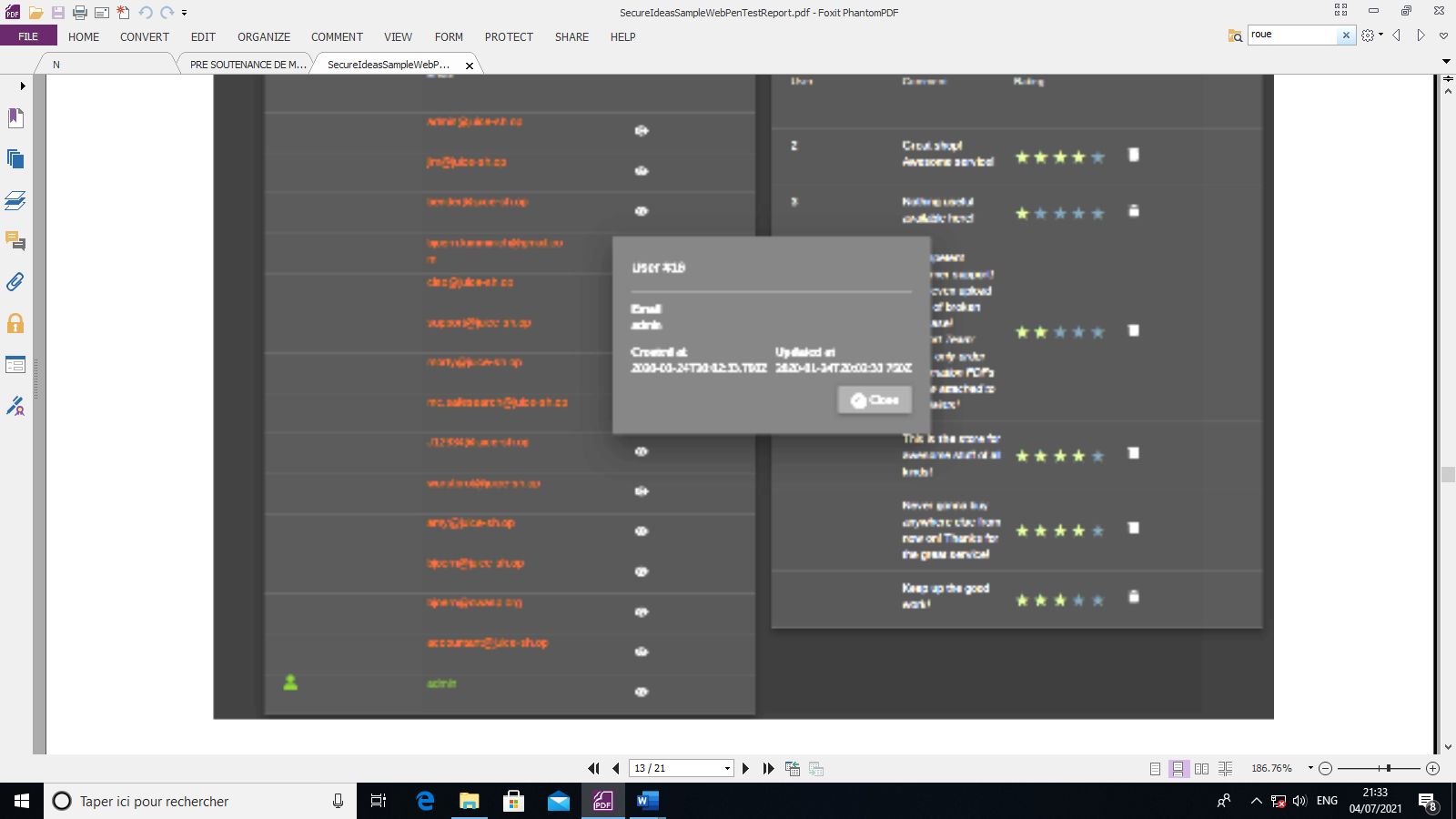
****

Figure:

**b.2.3 Recommandations**

Nous recommandons que la faille de contournement d'autorisation soit corrigée immédiatement en raison de l'exposition d'accès administratif via l'API.

La première étape pour remédier à cette faille consiste à modifier l'application pour valider l'autorisation informations. Lentreprise doit modifier le code de l'application pour vérifier qu'un utilisateur est autorisé à consulter les informations avant de les renvoyer au navigateur. Si l'utilisateur n'est pas autorisé, l'application doit renvoyer un message d'erreur au lieu des informations demandées.

La deuxième étape consiste à ne jamais faire confiance à l'entrée fournie par l'utilisateur ou à s'attendre à ce que le code côté client soit protégé contre manipulation. Chaque autorisation doit être validée par les services backend, et l'exposition de ce processus de validation doit être caché autant que possible à tout processus côté client. Cela aidera à assurer que toute entrée de l'utilisateur est traitée en toute sécurité.

L'étape suivante consiste à inclure un système de journalisation et de surveillance dans l'application pour détecter les tentatives d’accéder aux informations des autres membres. Ces journaux peuvent ensuite être examinés pour déterminer si quelqu'un est tentative d'attaque de l'application.

Chaque fois qu'un utilisateur tente d'accéder à des informations ou à des fonctionnalités qui lui sont interdites, l'application peut alerter les membres du personnel de la tentative. Cela peut être effectué de plusieurs manières. Le plus simple est que l'application envoie un e-mail ou un SMS au personnel d'assistance de Juice Shop. L’entreprise pourrait modifier également l'application pour envoyer des messages à une solution de surveillance centrale, si celle-ci a été mis en œuvre au sein de l'infrastructure de Juice Shop. Si modifier l'application de cette manière n'est pas préféré, l’entreprise peut également utiliser un script qui analyse les fichiers journaux pour les messages de tentatives d'exploitation.

Un script peut alors effectuer l'action choisie pour alerter le personnel de l’entreprise.

**c. Constats à haut risque**

**c.3. Cross-Site Scripting Flaws**

*Tableau :*

|  |  |
| --- | --- |
| **Normes de l'industrie** |  |
| Top 10 de l'OWASP | A5 : Contrôle d'accès cassé |
| NIST 800-53 | Application de l'accès AC-3 |

**c.3.1 Résumé**

Ne pas filtrer les entrées non fiables fournies par l'utilisateur peut exposer une application Web à plusieurs catégories de vulnérabilités. L'une de ces catégories est l'injection de code HTML ou JavaScript par un tiers. Ce type d'attaque est généralement appelé « Cross-Site Scripting » ou XSS.

Une façon courante d'exploiter cela consiste à utiliser un vecteur d'attaque d'ingénierie sociale et un lien conçu.

Cela exploiterait une faille dans un ou plusieurs paramètres de l'URL et de la chaîne de requête. Lorsque la cible l'utilisateur suit le lien, le code malveillant s'exécute dans le navigateur de la cible, dans le cadre d’une page vulnérable.

Les failles XSS sont généralement classées selon deux attributs : selon qu’elles soient persistantes et selon qu’elles soient reflétés. Lorsqu'un exploit persistant est utilisé, la charge utile est stockée et s'exécute à nouveau lors de visites ultérieures sur la page vulnérable. L'exemple classique est la persistance côté serveur dans la base de données. Étant donné que les données de la base de données peuvent être partagées entre les utilisateurs, il est possible pour un attaquant pour simplement ajouter la charge utile via un champ de données partagé afin de contourner le besoin pour l'ingénierie sociale. Cela suppose que l'attaquant puisse ajouter la charge utile de l'un ou l'autre un compte légitime ou un contexte non authentifié. Même lorsque l'ingénierie sociale est nécessaire pour introduire la charge utile, si elle se trouve dans des données partagées, elle peut toujours atteindre d'autres utilisateurs en plus de la cible original. Cependant, la persistance n'est pas nécessairement toujours sur le serveur et pourrait à la place être stockée dans les cookies définis par JavaScript. Dans les applications les plus modernes, les API localStorage et IndexedDB côté client peuvent également être utilisées.

L'autre attribut utilisé pour la classification est de savoir s'il s'agit d'un défaut reflété. S'il est reflété, le défaut est dans le traitement des entrées envoyées au serveur et renvoyées dans une réponse. L'exemple persistant dans la base de données fait cela, et pourrait donc être considéré à la fois comme reflété et persisté. Un exemple non persistant serait un message d'erreur renvoyé par le serveur qui inclut de manière non sécurisée une valeur de l'entrée.

Dans tous les cas, les scripts malveillants sont exécutés dans un contexte qui semble provenir de la cible. Cela donne à l'attaquant un accès complet au document récupéré, fournissant un contrôle presque illimité sur l'expérience utilisateur de la victime sur l'application. Une grande variété d'options est disponible pour créer un exploit efficace, qui peut incorporer certains des éléments suivants :

* Envoi de données applicatives à un serveur contrôlé par l'attaquant
* Utilisation de la session de la victime pour accéder à des données ou fonctionnalités supplémentaires
* Vol des cookies qui ne sont pas protégés par le httponly flag
* Manipuler la vue présentée à la victime à des fins d'ingénierie sociale, comme simuler un délai d'expiration de session pour demander une connexion ou convaincre l'utilisateur d'installer quelque chose
* Vol de données à partir de zones de saisie sensibles, telles que les informations d'identification de compte ou de carte de crédit informations
* Lancement d'attaques ou collecte de données à partir d'autres applications ouvertes à l'interaction avec le domaine actuel via une politique de partage des ressources inter-origine (CORS : cross-origin resource sharing), potentiellement en utilisant les informations d'identification stockées dans les cookies de la victime
* Modification des liens sur la page pour inclure la charge utile de script intersites dans d'autres pages en tant que l'utilisateur navigue sur le site

**c.1.2 Découverte**

Nous avons découvert que l’application Juice Shop est vulnérables aux scripts intersites (XSS) en raison de l'utilisation par l'application de l'entrée dans la réponse à l'utilisateur. De nombreux défauts identifiés ont été conservés dans la base de données, et nombre d'entre eux pourraient être exploités par un utilisateur non authentifié attaquant sans s'appuyer sur une attaque d'ingénierie sociale directe telle que le phishing.

Un exemple de faille XSS se trouve dans la page de profil utilisateur de l'application Juice Shop. Un attaquant peut remplacer l’identifiant piste résultat avec une charge utile iFrame JavaScript. Lorsqu'une charge utile, telle que <iframe src = "javascript: alert (` xss`)"> est présentée dans la barre d'adresse du navigateur, il provoque la application pour intégrer l'attaque dans la page Web résultante. La capture d'écran ci-dessous est ce que le navigateur victime verrait.

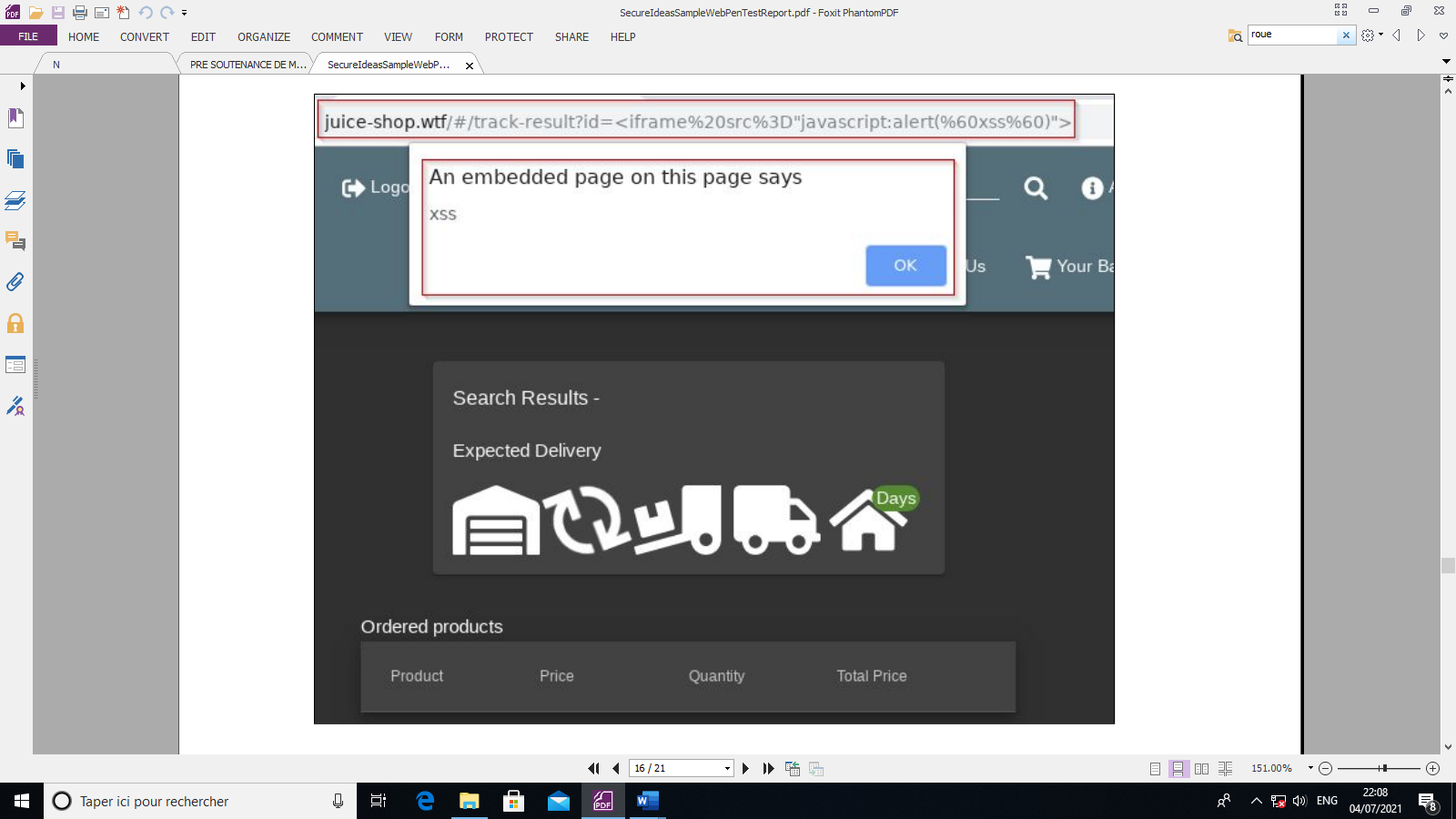


Figure:

**c.1.3 Recommandations**

Nous recommandons à l’entreprise d'effectuer à la fois la validation d'entrée et l'encodage de sortie pour éviter cette forme d'attaque. Les développeurs doivent s'assurer que l'application valide que l'entrée de l'utilisateur est exactement du type prévu par le développeur. Par exemple, si Juice Shop n'attend que des caractères alphanumériques caractères dans l'entrée, alors l'application doit effectuer un filtrage d'entrée pour rejeter tout le reste. L'encodage offre une protection supplémentaire en garantissant que des données hostiles, telles que JavaScript, ne seront pas envoyées au navigateur. De cette façon, si une attaque dépasse le filtrage d'entrée, elle sera débloquée ou rendue non malveillant par l'encodage de sortie.

Les développeurs de Juice Shop peuvent utiliser une bibliothèque de sécurité commune pour effectuer ce filtrage d'entrée et cette sortie cryptée. Les implémentations doivent suivre les meilleures pratiques OWASP pour prévenir cette vulnérabilité, que l’entreprise choisisse ou non d'utiliser une bibliothèque pour ces tâches. Ces recommandations peuvent être trouvé à :

https://owasp.org/www-project-cheat-sheets/cheatsheets/Cross\_Site\_Scripting\_Prevention\_Cheat\_Feuille.html.

**d. constatations à risque moyen**

**d.1 Inadequate Security Standards for Password**

*Tableau :*

|  |  |
| --- | --- |
| **Normes de l'industrie** |  |
| Top 10 de l'OWASP | A6 : Mauvaise configuration de la sécurité |
| NIST 800-53 | SC-28 Protection des informations au repos  IA-2 Identification et authentification |

**d.1.1 Résumé**

Le stockage des mots de passe est un point clé pour sécuriser les actifs d'une entreprise. Si des mots de passe sont stockés ou transmis via de mauvais algorithmes ou pire, en texte clair, puis l'ensemble du système logiciel de l'entreprise est vulnérable à l'exploitation si la table des mots de passe est accédée par un attaquant.

Si les mots de passe ne sont pas changés régulièrement, si l'algorithme n'est pas salé (en veillant à ce qu'il n'y ait pas deux les mots de passe codés sont les mêmes), et si le cryptage est inadéquat, alors une entreprise est confrontée à la risque qu'un attaquant accède à chaque compte de la base de données pendant un certain temps jusqu'à l'attaque est découvert.

**d.1.2 Découverte**

En raison de la faille d'injection SQL ci - dessus, nous avons trouvé les utilisateurs de Juice Shop et les mots de passe. L'analyse du tableau des mots de passe a révélé que les mots de passe sont stockés sous la forme d'un hachage MD5 de le mot de passe d'origine de l'utilisateur. La capture d'écran suivante montre un échantillon des données.

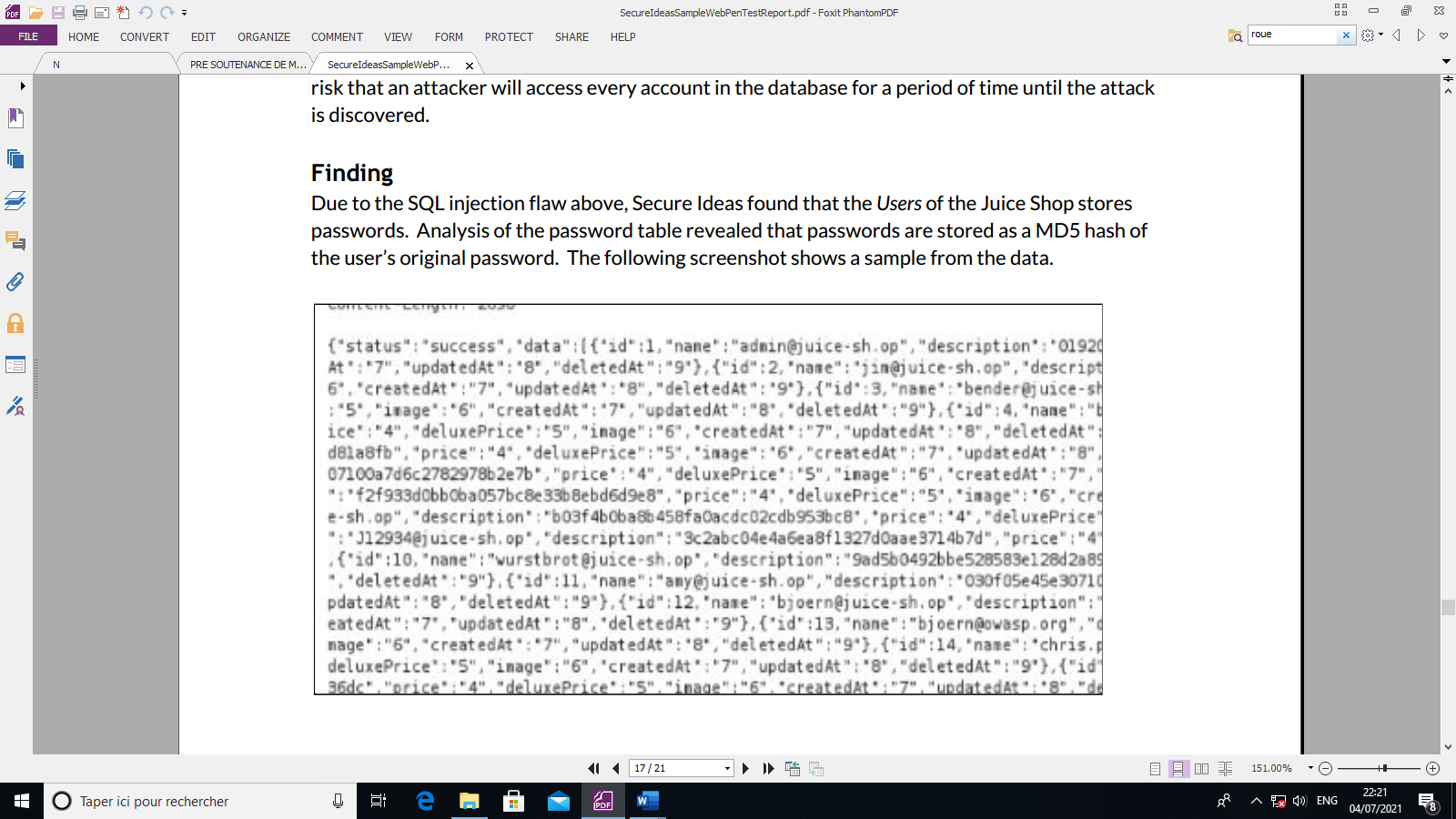


Figure:

MD5 est un algorithme de hachage connu pour avoir des problèmes permettant des collisions cryptographiques, ce qui signifie que deux données différentes peuvent produire le même hachage MD5. MD5 se prête aussi aux attaques par force brute en raison de la puissance de calcul relativement faible nécessaire pour générer un MD5 hacher.

De plus, les hachages MD5 que nous avons trouvés n'étaient pas salés. Cela est évident par le fait que plusieurs des hachages de mot de passe dans la capture d'écran précédente sont identiques. Les sels de mot de passe font il est beaucoup plus difficile pour un attaquant de déchiffrer un hachage de mot de passe car il nécessite que l'attaquant connaisse le sel valeur afin de commencer à casser les hachages.

**d.1.3 Recommandations**

Nous recommandons que l’entreprise stocke les mots de passe dans la base de données avec un code à sens unique sécurisé algorithme de hachage ainsi qu'un sel.

Un sel unique par utilisateur permet de s'assurer que chaque hachage de mot de passe apparaît différent même si le même mot de passe a été sélectionné par plusieurs utilisateurs. S'assurer que chaque hachage est différent est une mesure qui augmente la complexité, et donc le temps nécessaire, pour casser les hashs. Lorsque l'application doit authentifier un utilisateur, il doit hacher le mot de passe fourni, avec le sel, et comparer les hachages.

La ressource OWASP suivante fournit plus d'informations sur la valeur des sels et corrige la mise en œuvre : https://www.owasp.org/index.php/Password\_Storage\_Cheat\_Sheet

De plus, nous recommandons à l’entreprise d'utiliser la fonction de hachage PBKDF2 pour stocker mots de passe. Si PBKDF2 n'est pas possible pour une raison quelconque, alors l’entreprise devrait s'en remettre au "scrypt" ou des fonctions « bcrypt ». Toutes ces solutions de hachage sont conçues pour être lourdes en termes de calcul, donc qu'un attaquant avec une liste volée de hachages aura besoin de ressources importantes pour les casser. Chacun de ces les algorithmes sont également adaptatifs au fil du temps, ce qui signifie qu'ils peuvent être configurés pour devenir plus informatiquement lourde à mesure que la technologie devient plus rapide, créant ainsi un degré de « future preuve » dans les algorithmes.

**e. Constatations à faible risque**

**e.1 Weak Password Complexity Requirements**

*Tableau :*

|  |  |
| --- | --- |
| **Normes de l'industrie** |  |
| Top 10 de l'OWASP | A6 : Mauvaise configuration de la sécurité  A5 : Contrôle d'accès cassé  A2 : Authentification brisée |
| NIST 800-53 | AC-3 : Contrôle de l'accès |

**e.1.1 Résumé**

L'un des aspects testés lors du test d'intrusion était l'exigence de complexité du mot de passe de l’application Juice Shop. Pour la plupart des applications, le mot de passe est le seul facteur d’authentification qui donne accès à toutes les autres informations. Pour cette raison, il est impératif que les utilisateurs créent des mots de passe forts difficiles à attaquer. Malheureusement, la plupart des utilisateurs ne comprendre l'importance des mots de passe forts ou comment les créer. Développeurs d'applications doit prendre la responsabilité de développer des applications d'une manière qui oblige les utilisateurs à créer des mots de passe capables de résister aux attaques courantes de recherche de mots de passe.

Il existe trois types courants d'attaques par devinette de mot de passe. La première est une attaque par force brute dans quels attaquants essaient chaque combinaison de chaque lettre afin de finalement trouver la bonne le mot de passe. Attaques par dictionnaire utilisent une liste de mots de passe communs tels que Password1 et abc123.

Le troisième type d'attaque est une attaque hybride dans laquelle l'attaquant utilise des mots de passe communs qui ont été mutilé avec des techniques de force brute. Par exemple, l'attaquant pourrait essayer le mot secret suivi de toutes les combinaisons possibles de chiffres et de symboles à 2 chiffres. Cela peut réussir lorsque les utilisateurs ajoutent des chiffres et des symboles à la fin de leur mot de passe pour se conformer au mot de passe conditions.

**e.1.2 Découverte**

Nous avons constaté que, bien que l'application tente d'imposer l'utilisation de mots de passe complexes, les exigences de complexité du mot de passe sont plus faibles que celles recommandées pour ce type d'application.

Nous avons trouvé que l'application a été testée mots de passe autorisés tels que admin123 et password123 . Ces types de mots de passe se trouvent couramment dans les dictionnaires largement accessibles. Comme

En effet, des idées sécurisées utilise généralement la password123 chaîne contre les systèmes mettre en œuvre le verrouillage de compte car il est généralement trouvé comme mot de passe pour les comptes sur le Web applications.

**e.1.3 Recommandations**

L’entreprise devrait renforcer les exigences de mot de passe dans l'application. Bien qu'il fonctionne vérification de la complexité, elle devrait augmenter ces vérifications en fonction des normes de l'industrie. Une source d'une norme de l'industrie provient du SANS Institute. SANS publie un guide gratuit sur les mots de passe complexité, disponible sur :

https://www.sans.org/security-resources/policies/general/pdf/password-construction-guidelines

Selon ce guide, les mots de passe doivent contenir au moins quatorze caractères, de préférence phrase secrète. De plus, les mots de passe doivent éviter l'un des défauts suivants :

* Contenir huit caractères ou moins
* Contenir des informations personnelles telles que des dates de naissance, des adresses, des numéros de téléphone ou des noms des membres de la famille, des animaux domestiques, des amis et des personnages fantastiques
* Contenir des modèles numériques tels que aaabbb, qwerty, zyxwvuts ou 123321
* Ou une version de Welcome123, password123 ou Changeme123

Nous recommandons également que l’entreprise examine les options permettant d'ajouter une authentification multifacteur à l'application. Les informations d'identification traditionnelles des utilisateurs sont facilement réutilisées par les attaquants une fois volées. En appliquant une deuxième forme d'authentification, telle que quelque chose que l'utilisateur sait, quelque chose qu'il est ou quelque chose que l’utilisateur a, l’entreprise peut garantir que les comptes d'utilisateurs sont beaucoup plus difficiles à compromettre. En plus, les mots de passe moins complexes présentent moins de risques lorsque des facteurs d'authentification supplémentaires sont requis.

Enfin, l’entreprise doit examiner la journalisation de l'application pour confirmer que les tentatives de connexion infructueuses sont enregistrées dans un journal d'erreurs. Les journaux doivent être examinés quotidiennement pour détecter les comptes d'utilisateurs avec une valeur supérieure à nombre normal de tentatives de connexion infructueuses.

**f. orientation stratégique**

Nous avons effectué un test de pénétration d'applications Web pour Juice Shop. En testant cette application nous avons pu avoir une idée générale de la posture de sécurité de l’entreprise et souhaitons mettre les considérations stratégiques suivantes à sa disposition :

* **Fournir une formation au codage sécurisé pour les développeurs :** Trouver et corriger les défauts après coup est le moyen le plus coûteux pour gérer vulnérabilités de sécurité des organisations. Il faut beaucoup de temps et d'efforts aux développeurs pour examiner les problèmes découverts, revoir le code, apporter les modifications appropriées, travailler par le biais de tests d'assurance qualité, puis de déployer les modifications. Alternativement, la formation développeurs pour comprendre les failles de sécurité et éviter les vulnérabilités pendant le processus de développement est beaucoup plus efficace et efficient. Malheureusement, la plupart des sites de formation pour développeurs enseignent de manière adéquate le codage sécurisé. Au cours de cette évaluation, nous avons trouvé des preuves que suggère que de nombreux développeurs de l’entreprise ne sont pas correctement formés pour éviter les erreurs courantes. L’entreprise devrait envisager de fournir régulièrement une formation au codage sécurisé à tous les développeurs.
* **Envisagez l'authentification multifacteur :** L'authentification multifacteur est recommandée pour les employés à utiliser lors de l'accès systèmes sensibles tels que VPN, contrôleurs de domaine et autres systèmes critiques ou ressources sensibles. Les facteurs du mécanisme d’authentification multi-facteurs se divisent en trois catégories :
* la connaissance (quelque chose qu'ils savent),
* la possession (quelque chose qu'ils ont)
* et l’inhérence (quelque chose qu'ils sont).

Il existe une large gamme de systèmes qui peuvent être mis en œuvre directement dans les systèmes d'authentification Windows Server ainsi que les serveurs Linux et applications. Un système commun trouvé dans les environnements d'entreprise est le RSA SecurID solution. Un autre système populaire dans les environnements à plus petite échelle est disponible auprès de Duo Sécurité ou Google Authenticator. Quelle que soit la solution choisie, le plus important aspect de la mise en œuvre de l'authentification multifacteur est de s'assurer qu'au moins deux des facteurs sont nécessaires pour accéder et non simplement le même facteur nécessaire à plusieurs reprises (c'est-à-dire, mot de passe plus empreinte digitale au lieu de plusieurs mots de passe).