**REPUBLIC OF CAMEROON**

Peace – Work – Fatherland

\*\*\*\*\*\*\*\*\*\*\*\*

**Ministry of Higher Education**

\*\*\*\*\*\*\*\*\*\*\*\*

**University of Dschang**

\*\*\*\*\*\*\*\*\*\*\*\*

**Coast University Institute**

\*\*\*\*\*\*\*\*\*\*\*\*

**Higher Institute of Technology and Industrial Design**

**REPUBLIQUE DU CAMEROUN**

Paix – Travail – Patrie

\*\*\*\*\*\*\*\*\*\*\*\*

**Ministère de l’Enseignement Supérieur**

\*\*\*\*\*\*\*\*\*\*\*\*

**Université de Dschang**

\*\*\*\*\*\*\*\*\*\*\*\*

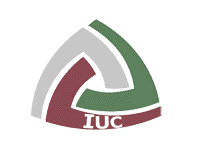
**Institut Universitaire de la Côte**

\*\*\*\*\*\*\*\*\*\*\*\*

**Institut Supérieur des Technologies et du Design Industriel**

******

**AUDIT DE LA SECURISATION D’UN SAAS : Le pentesting**

****

**Mémoire rédigé et soutenu en accomplissement partiel, en vue de l’obtention du Master**

**Professionnel par :**

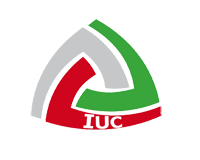
**NKAN MASSOM Calvin Rodrigue**

**Matricule : IUC19E0052419**

**Spécialité : Système d’Information et Génie Logiciel (SIGL)**

**Ingénieur des travaux en Informatique et Télécommunications**

Supervisé et encadré par :

****

**Année Académique 2020 – 2021**

**Superviseur** :

Dr. MEUKALEUNI Cyrille

*Enseignant à l’IUC*

**Encadreur académique** :

**M. FOTSO KUATE Victor**

*Enseignant à l’IUC*

**Encadreur Professionnel** :

M. LAGOUE NJINTHE Rodrigue

*Head of R&D LAO SARL*

**Superviseur** :

Dr. MOSKOLAI Justin

*Enseignant à l’IUC*

**Encadreur académique** :

**M. TEKEU Hyppolite**

*Enseignant à l’IUC*

**Encadreur Professionnel** :

M. LAGOUE NJINTHE Rodrigue

*CEO Founder LAO SARL*

**Année Académique 2020 – 2021**

******

# **DEDICACE**

A MA FAMILLE

# **REMERCIMENTS**

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, sa disponibilité, ses nombreux encouragements et surtout ses judicieux conseils 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 ; il a partagé ses connaissances et expériences dans ce milieu, tout en m’accordant sa confiance, ses conseils et la patience ;
* **M. TSOPBENG David**, le Directeur de l’Institut Supérieur des Technologies et du Design Industriel (ISTDI), pour tous les efforts qu’il a fournis 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 FOUEKEN** , 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 leurs soutient moral, et financier, pour les conseils qu’ils me donnent au quotidien, ainsi que mes frères et sœurs pour leur amour inconditionné ;
* J’adresse mes sincères remerciements à tous les **professeurs**, **intervenants** et toutes les **personnes** qui par leurs paroles, leurs écrits, leurs conseils et leurs critiques ont guidé mes réflexions et ont accepté de me rencontrer et de répondre à mes questions 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**

[**DEDICACE** 1](#_Toc75371383)

[**REMERCIMENTS** ii](#_Toc75371384)

[**SOMMAIRE** iii](#_Toc75371385)

[**ABREVIATIONS** v](#_Toc75371386)

[**LISTE DES FIGURES** vi](#_Toc75371387)

[**LISTE DES TABLEAUX** vii](#_Toc75371388)

[**RESUME** viii](#_Toc75371389)

[**ABSTRACT** ix](#_Toc75371390)

[**INTRODUCTION GENERALE** x](#_Toc75371391)

[**PARTIE I : ETAT DE L'ART** 1](file:///C:\Users\win10\Documents\memo\memo-03.docx#_Toc75371392)

[**CHAPITRE I : REVUE DE LA LITTERATURE** 2](#_Toc75371393)

[SECTION1 : CONCEPTES ET DEFINITIONS 2](#_Toc75371394)

[SECTION2 : LES MENACES CONTRE LA SÉCURITÉ DES APPLICATIONS 5](#_Toc75371395)

[SECTION3 : ANALYSE DES PROBLEMES DE SÉCURITÉ DES APPLICATIONS EXISTANTES 10](#_Toc75371396)

[**CHAPITRE 2 : CADRAGE DU PROJET** 12](#_Toc75371397)

[SECTION 1 : CONTEXTE DU PROJET ET ETUDE DE L’EXISTANT 12](#_Toc75371398)

[SECTION 2 : CAHIER DE CHARGE 27](#_Toc75371399)

[SECTION 3 : METHODOLOGIE ET OUTILS UTILISES 35](#_Toc75371400)

[**PARTIE II : ANALYSE, SPECIFICATION ET IMPLEMENTATION** 49](file:///C:\Users\win10\Documents\memo\memo-03.docx#_Toc75371401)

[**CHAPITRE 3 : ANALYSE ET SPECIFICATION** 50](#_Toc75371402)

[**CHAPITRE 4 : RESULTATS OBTENUS ET DISCUTION** 51](#_Toc75371403)

[**CONCLUSION GENERALE** 52](#_Toc75371404)

[**Annexes** 53](#_Toc75371405)

[**Table de matières** 54](#_Toc75371406)

[**Références bibliographiques** 55](#_Toc75371407)

[IV.1 Compétences **Error! Bookmark not defined.**](#_Toc75371408)

[IV.2- Certifications de soutien **Error! Bookmark not defined.**](#_Toc75371409)

[IV.3- Logiciels et outils **Error! Bookmark not defined.**](#_Toc75371410)

[IV.4- Considérations supplémentaires **Error! Bookmark not defined.**](#_Toc75371411)

[CAHIER DES CHARGES **Error! Bookmark not defined.**](#_Toc75371412)

[I- LES FAILLES DE WINOWS XP **Error! Bookmark not defined.**](#_Toc75371413)

[II- CAS D’UTILISATION: prise en en main du shell distant de windows XP à l’aide de KALI LINUX **Error! Bookmark not defined.**](#_Toc75371414)

[CONCLUSION **Error! Bookmark not defined.**](#_Toc75371415)

[SECTION4 : RÉSOLUTION DES ERREURS PENDANT TOUT LE CYCLE DE VIE DU DÉVELOPPEMENT LOGICIEL **Error! Bookmark not defined.**](#_Toc75371416)

# **ABREVIATIONS**

# **LISTE DES FIGURES**

[Figure 1 : Sécurité des applications Web : les enjeux les plus courants (source : site IBM)5](#_Toc74898328)

[Figure 2 : Illustration graphique (source : Rapport enquête réalisée par Sanctum - IBM)**Error! Bookmark not defined.**](#_Toc74898329)

# **LISTE DES TABLEAUX**

[Tableau1 : Défis - assurance2](#_Toc75098150)

[Tableau2 : Défis - Finance3](#_Toc75098151)

[Tableau3 : Défis – Industrie pharmaceutique3](#_Toc75098152)

[Tableau4 : Défis – Industrie pharmaceutique4](#_Toc75098153)

[Tableau5 : Menaces - usurpation d’identité6](#_Toc75098154)

[Tableau 6 : Menaces - falsification7](#_Toc75098155)

[Tableau 7 : Menaces - répudiation7](#_Toc75098156)

[Tableau 8 : Menaces - divulgation de l’information8](#_Toc75098157)

[Tableau 9 : Menaces - Déni de service (DoS)8](#_Toc75098158)

[Tableau 10 : Menaces - élévation de privilèges9](#_Toc75098159)

[Tableau 11 : Coûts relatifs calculés en fonction du délai écoulé entre l’apparition de l’erreur et sa détection**Error! Bookmark not defined.**](#_Toc75098160)

[Tableau 12 : Avantages et inconvénients – Tests manuels**Error! Bookmark not defined.**](#_Toc75098161)

[Tableau 13 : Avantages et inconvénients – Tests automatiques**Error! Bookmark not defined.**](#_Toc75098162)

[Tableau 14 : Avantages et inconvénients – Test boite blanche**Error! Bookmark not defined.**](#_Toc75098163)

[Tableau 15 : Avantages et inconvénients – Test boite grise**Error! Bookmark not defined.**](#_Toc75098164)

# **RESUME**

Avec la demande sans cesse croissante de la demande logiciel et les limites infrastructurelles que rencontres les consommateurs de ces produits, le cloud vient offrir une porte de sortie au monde, en libérant les entreprises et autres, de l’obtention, de l’entretien et des couts liés à l’infrastructure qui doit héberger leurs applications. Et d’entre les propositions de solutions, nous avons le SaaS (Software As A Service). Sauf que, cette mise à disposition du patrimoine logiciel des organismes, via internet l’expose au danger de celui-ci. En effet le nombre d’attaque évolue considérablement chaque année, avec des raisons pouvant être d’ordre politique, économique, ou tout simplement du challenge. Il a donc fallu penser à protéger les logiciels en générale et pour en venir à ce qui nous intéresse, les SaaS. Cependant, cette sécurisation s’épuise au fil du temps, car une application sécurisée aujourd’hui ne le sera plus demain, ce qui va soulever la question de la veille permanente et donc des évaluations continues des logiciels, érigées en audit de sécurisation de celles-ci. Il est donc question pour nous de définir et réaliser un audit spécifique a une optimisation sécuritaire pour les applications de type SaaS, afin de minimiser le plus que possible la réussite de toutes entreprises malveillantes à leur encontre. Pour cela nous avons étudié le SaaS et son contexte sécurité, puis les possibilités d’audit existantes afin d’en ressortir la meilleure. Ensuite nous avons implémenté cet audit dans les règles de l’art pour exposer les failles d’un SaaS et en proposer les axes de correction. Et enfin nous avons réalisé des analyses afin de proposer de meilleurs pratiques.

# **ABSTRACT**

With the ever-increasing demand for software and the infrastructural limitations that consumers of these products face, the cloud provides a way out for the world, freeing businesses and others from obtaining, maintaining and costs related to the infrastructure that must host their applications. And among the proposed solutions, we have SaaS (Software As A Service). Except that this provision of the software heritage of organizations, via the Internet, exposes it to the danger of it. Indeed, the number of attacks changes considerably each year, with reasons that may be political, economic, or simply challenge. We therefore had to think about protecting software in general and to come to what interests us, SaaS. However, this security becomes exhausted over time, because an application that is secure today will no longer be so tomorrow, which will raise the question of permanent monitoring and therefore continuous evaluations of software, set up as a security audit of those. -this. It is therefore a question for us of defining and carrying out a specific audit for a security optimization for SaaS type applications, in order to minimize as much as possible the success of all malicious attempt against them. For this we have studied SaaS and its security context, then the existing audit possibilities in order to come out the best. Then we implemented this audit according to the rules of the art to expose the flaws of a SaaS and to propose the axes of correction. And finally we carried out analyzes in order to propose best practices.

# **INTRODUCTION GENERALE**

Du fait de la crise sanitaire avec le covid-19 et des mesures, dont la distanciation sociale et les mesures barrières qui en résultent, il apparaît sans conteste que les possibilités logicielles, 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. Aussi l’industrie numérique bénéficie d’une émancipation tant sur le plan des équipements informatiques que sur le plan de la production des logiciels. Cependant ces logiciels sont conçus et réalisés pour répondre 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. Dans cette situation, les solutions logicielles répondant au mieux, en faveur des nouvelles conditions (mesures barrières, distanciation et autres...) apparaissent vitales pour les économies.

Pour satisfaire cela, ces solutions logicielles doivent non seulement être stratégiquement alignées aux organismes qu’elles permettent de gérer, mais également assurer, sans interruptions ni erreurs, et surtout de manière sécurisée une continuité de la gestion de leurs activités. Ce qui nécessite un audit de la sécurisation des services applicatifs accessibles à distance, tels les SaaS afin de satisfaire la quiétude des entreprises et la satisfaction croissantes des clients qui en sont les consommateurs, pour une amélioration financière considérable. D’où la question du meilleur processus à mettre en œuvre pour réaliser un audit de la sécurisation des applications SaaS, pour le cas qui nous concerne.

L’audit de la sécurisation étant l’évaluation de l’ensemble des méthodes et moyens mise en place pour assurer en toutes circonstances, la sécurité de l’activité de son consommateur, mais aussi pour ceux qui mettent sur pieds des applications SaaS, qui fournissent beaucoup d’avantages dont une grande efficacité. En outre, l’audit de la sécurisation de ces applications répond à l’utilisation d’un scénario programmé pour vérifier la conformité d’une ou plusieurs de leurs fonctions à intervalles réguliers et/ou à travers des plates-formes multiples. Sauf qu’aujourd’hui, la plupart des applications quelles qu’elles soient, doivent fréquemment être mis à jour fournissant de nouvelles caractéristiques et fonctions avec une expérience utilisateur améliorée due à une ergonomie de qualité et autres. Mais hélas, et assez souvent, des possibilités de compromissions non-négligeables en résultent. N’importe quel changement peut faire apparaitre de nouveaux contextes de compromission, ce qui implique que chaque changement doit être suivi d’un audit, pour lui et pour l’environnement qu’il intègre désormais. Dans le cas contraire on prend le risque d’exposer l’application et ses utilisateurs a des attaques en tout genre. Lors de la réalisation d’un audit de sécurisation applicative, une évaluation de la conformité se révèle très vite limitée, car elle souligne l’alignement du point de vu de la norme, pourtant l’efficacité n’en est que plus accru que si à ceci vient s’ajouter une étude des vulnérabilises possibles et surtout leur mise à l’épreuve pour éliminer les faux positifs éventuels. Car ainsi on peut circonscrire au mieux les mesures à prendre pour se rapprocher d’un état de sécurité optimale, d’autant plus que la sécurité absolue reste un mythe, et par la même éviter à l’entreprise réalisatrice de ces applications de supporter des couts inutiles de sécurisation. Donc à la question principale évoquée ci-dessus, s’ajoute la question de savoir, comment optimiser le processus d’audit de la sécurisation des applications SaaS en testant ces applications tel un attaquant.

Par conséquent, répondre à de telles préoccupations, confère à l’entreprise en charge, la garantie de la mise sur pied d’applications fiables mais surtout sécurisées, en tenant compte du contexte global actuel. Ainsi pour l’objectif qu’est le nôtre, il sera question, de démonter les possibilités qu’offre un d’audit de sécurité optimal pour une organisation de réalisations d’applications SaaS. Et le déroulement de ce travail suivra le découpage qui suit : un état de l’art et une étude sur la mise en place de l’audit de la sécurisation des applications de type SaaS, un cahier de charge pour la description du travail à effectuer, une analyse et une optimisation d’une approche audit, puis après une mise en œuvre, une évaluation des résultats obtenus

# **PARTIE I :** **ETAT DE L'ART**

# **CHAPITRE I : REVUE DE LA LITTERATURE**

Aujourd'hui, les pirates sur internet qui attaquent les applications sont capables de les utiliser contre les organisations pour les rendre vulnérable, les mettre dans l’embarras ou les voler. Les pares-feux et SSL sont devenus la norme et pourtant, d’après des enquêtes (de IBM), trois sites Web sur quatre sont vulnérables aux agressions, et la vaste majorité de ces offensives vise la sécurité des applications. Les entreprises font confiance à la sécurité du réseau et de l’hôte, mais souvent, ces mesures ne suffisent pas pour empêcher ces attaques contre les applications Web. La sécurité des applications diffère de celle du réseau et du système hôte. Les approches traditionnelles d’implémentation de la sécurité du réseau et de l’hôte ne conviennent pas à ce niveau. Il est donc question de donner le pourquoi des mesures à pour renforcer la sécurité de des applications.

## SECTION1 : CONCEPTES ET DEFINITIONS

### 1.1 Les défis de la sécurité des applications

Plusieurs entreprises ont dû affronter une grande diversité de défis. Dans les tableaux qui suivent, nous récapitulons les enjeux métier et techniques soulevés par certains d’entre eux, et pour lesquels nous présenterons des solutions un peu plutard dans cet exposé.

#### 1.1.1 En termes d’assurances : exemple d’une très grande société d’assurances gérant plusieurs millions de clients particuliers ou groupes de clients.

Tableau1 : Défis - assurance

|  |  |
| --- | --- |
| Défis métier | * Mise en conformité avec les réglementations officielles. * Sécurisation proactive des enregistrements clients sensibles. * Intégration de la sécurité des applications à une stratégie de sécurité multiniveaux, avec la prise en charge de plus de 4 000 praticiens dans 60 hôpitaux. * Suppression des coûts liés à la détection et à la correction des problèmes de sécurité au stade de post-production. |
| Défis techniques | * Croissance rapide du site. * Identification de nombreux défauts de sécurité des applications sur les sites (suite à un audit). * Garantie de la sécurisation des informations et des transactions des clients. * 95 % des données sont considérées comme confidentielles. |

Ref: “All-Out blitz against Web app attacks,” Network World, 17 mai 2004. (http://www.networkworld.com/techinsider/2004/0517techinsidermain.html)

#### 1.1.2 En termes de finance : une grande banque commerciale américaine dotée d’un portefeuille de plus de $200 milliards

Tableau2 : Défis - Finance

|  |  |
| --- | --- |
| Défis métier | * Respect de la politique de l’entreprise exigeant l’intégration de la sécurité des applications au cycle de vie du développement. * Plus de 3 000 applications nouvelles et existantes. * Réduction du coût global du cycle de vie du développement. * La banque dépense >$1M par an en « piratage éthique » afin de détecter les vulnérabilités avant ou après un déploiement. |
| Défis techniques | * Site très connu et massivement visité. * Dispersion du développement d’applications dans de nombreuses unités commerciales. * Incohérence des tests manuels et de l’analyse du code par les développeurs. * Les développeurs n’ont ni les outils ni les connaissances des techniques de test de la sécurité. |

#### 1.1.3 Industrie pharmaceutique : un grand laboratoire de produits pharmaceutiques investissant beaucoup dans la recherche

Tableau3 : Défis – Industrie pharmaceutique

|  |  |
| --- | --- |
| Défis métier | * Mise en conformité avec les réglementations internes et officielles en matière de protection des données. * Évaluation exacte des risques associés à chaque application. * Volonté de restriction des ressources consacrées à la sécurisation des applications et des données. |
| Défis techniques | * Grand nombre de groupes de développement et de secteurs d’activités disparates. * Nombreuses acquisitions et partenariats stratégiques. * Données confidentielles soumises des réglementations très strictes. * Respect des réglementations internationales. |

#### 1.1.4 En termes de loisirs et médias : une grande chaîne de télévision américaine

Tableau4 : Défis – Industrie pharmaceutique

|  |  |
| --- | --- |
| Défis métier | * Très forte exposition de la marque. * Profil haut de gamme très diversifié et caractéristiques média prêtant souvent à controverse. |
| Défis techniques | * Sites très dynamiques et très souvent modifiés. * Développement d’application entièrement décentralisé. * Petite équipe chargée de la sécurité des applications. * Plannings de production soumis à des délais très tendus. |

### 1.2 Définition de la sécurité des applications

Le modèle de référence OSI (Open System Interconnection) (Ref: International Organization for Standardization (www.iso.org) ) définit sept couches de protocoles réseau, chaque message transitant par chacune de ces sept couches. La couche supérieure, la couche 7, est la couche d’une application qui utilise des protocoles tels que HTTP. HTTP permet de transmettre des messages incluant du contenu de type HTML, XML, SOAP et des services Web. Dans le cadre de ce livre blanc, nous allons étudier en particulier les offensives contre les applications véhiculées par HTTP.

Les pares-feux classiques peuvent se révéler inefficaces contre les offensives transitant par HTTP. L’attaquant d’une application utilise des requêtes HTTP valides en passant par des ports connus, de sorte que les pares-feux réseau, de par leur conception, autorisent volontairement ce trafic pourtant nuisible, tout simplement parce que la couche réseau considère qu’il s’agit d’un bon trafic. En effet, l’élément nuisible n’est pas la requête HTTP elle-même mais les données qu’elle contient. Souvent, ces données dangereuses sont des données entrées par l’utilisateur, spécialement formatées ou organisées dans le but de modifier le comportement d’une application. Les offensives contre les applications peuvent par exemple autoriser un accès sans restriction aux bases de données, exécuter des commandes système arbitraires ou modifier le contenu d’une application.

Une sécurité des applications bien pensée interdit à un utilisateur mal intentionné de modifier le comportement d’une application. Les facteurs les plus fréquents responsables de la sécurité insuffisante des applications sont :

* Les obligations en matière de sécurité des applications, pour peu qu’elles aient été définies, sont souvent perçues comme impossibles à mettre en pratique, bien elles sont dotées d’une connotation négative (interdictions), ou encore elles s’avèrent trop floues.
* Le test de la sécurité des applications est effectué uniquement en cas d’audit.
* Les équipes chargées de la conception et de la définition des exigences des applications considèrent que la sécurité concerne l’équipe réseau ou informatique.
* Les procédures de test standard concernent principalement le comportement fonctionnel.
* Seuls les quelques rares « spécialistes de la sécurité » de l’entreprise sont conscients des menaces pesant sur la sécurité des applications.
* Le test de la sécurité des applications se limite à de courts créneaux au cours desquels des pirates dits éthiques tentent de simuler ce que feraient les vrais pirates.

## SECTION2 : LES MENACES CONTRE LA SÉCURITÉ DES APPLICATIONS

Les menaces contre la sécurité permettent de définir les technologies de sécurité qui peuvent être les plus efficaces pour défendre une application. Il est souvent préférable d’envisager des mesures défensives d’ordre général avant d’opter pour une technologie particulière. En choisissant cette approche, l’entreprise a la certitude de choisir les meilleures technologies pour leurs mérites et non sur leur seule réputation.

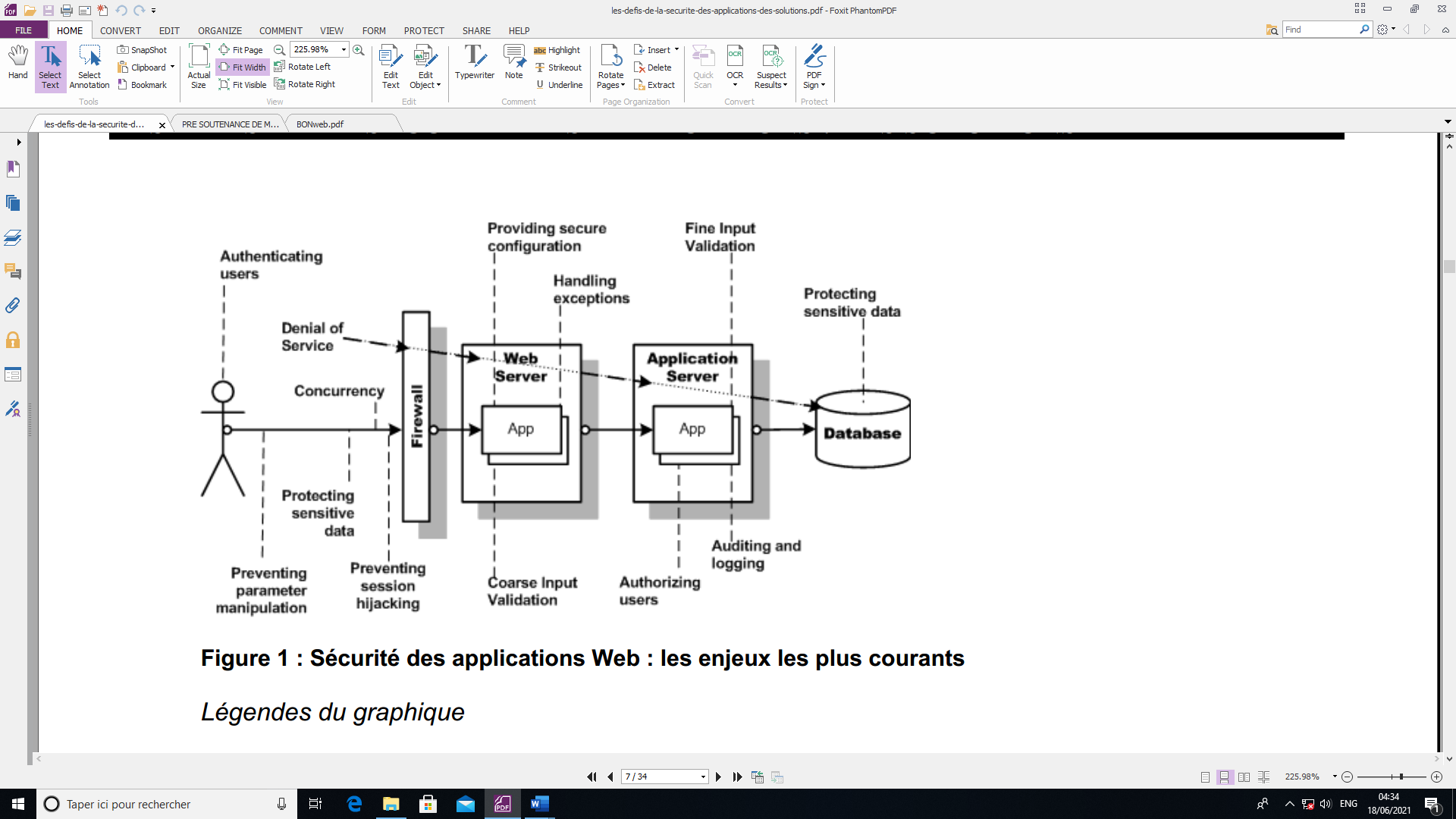


Figure 1 : Sécurité des applications Web : les enjeux les plus courants (source : site IBM)

Légendes du graphique :

* Authenticating users = Authentification des utilisateurs
* Denial of service = Déni de service
* Handling exceptions = Gestion des exceptions
* Providing secure configuration = Configuration sécurisée
* Fine Input Validation = Validation des entrées correctes
* Protecting sensitive data = Protection des données sensibles
* Concurrency = Simultanéité
* Firewall = Pare-feu
* Web server = Serveur Web
* Application Server = Serveur d’application
* Database = Base de données
* App = App.
* Protecting sensitive data = Protection des données sensibles
* Preventing parameter manipulation = Prévention contre la manipulation de paramètres
* Preventing session hijacking = Prévention du piratage de session
* Coarse input Validation = Validation de données incorrectes
* Authorizing users = Autorisation des utilisateurs
* Auditing and logging = Audit et journalisation

Par la suite, nous nous intéressons aux menaces les plus courantes, dont le type diffère selon l’application, ainsi qu’au solutions les plus souvent utilisées.

### 2.1 Usurpation d’identité

Chaque fois qu’un utilisateur sollicite un accès à des informations non publiques, l’entreprise doit pouvoir vérifier l’authenticité de l’identité qu’il revendique. En général, il est possible d’empêcher l’usurpation d’identité en appliquant une authentification rigoureuse. Ainsi que de se protéger en sécurisant les informations d’authentification.

*Tableau5 : Menaces - usurpation d’identité*

|  |  |
| --- | --- |
| Exemples : | * Un attaquant entre les informations d’authentification d’un autre utilisateur. * Un attaquant modifie le contenu d’un cookie ou d’un paramètre afin de se faire passer pour un autre utilisateur ou de faire croire que le cookie provient d’un autre serveur. |
| Erreurs courantes | * Usage d’une authentification basée sur les communications pour autoriser l’accès aux données d’un utilisateur. * Usage d’informations d’authentification non chiffrées qui peuvent être interceptées et réutilisées par un espion. * Stockage d’informations d’authentification dans des cookies ou des paramètres. * Usage de méthodes d’authentification « bricolées maison » ou non testées. * Le logiciel client n’est pas autorisé à authentifier l’hôte lorsque cela est nécessaire. * Utilisation d’une authentification provenant d’un domaine sécurisé erroné. |
| Solutions possibles | * Structures de sécurisation fournies par le système d'exploitation (par exemple Kerberos). * Usage de jetons chiffrés, tels que des cookies de session. * Usage de signatures numériques. |

### 2.2 Falsification

La falsification consiste à modifier ou supprimer une ressource sans autorisation.

*Tableau 6 : Menaces - falsification*

|  |  |
| --- | --- |
| Exemples | * Défiguration d’un site Web. * Modification des données pendant leur transit. |
| Erreurs courantes | * Autorisation accordée à des sources de données sans validation préalable. * Assainissement des données en entrée afin d’empêcher l’exécution de code indésirable. * Droits d’exécution accordés à des comptes dont les privilèges ont été augmentés. * Absence de chiffrement des données sensibles. |
| Solutions possibles | * Usage de la sécurité au niveau du système d'exploitation pour verrouiller les fichiers, les répertoires et les autres ressources. * Validation de données. * Affectation d’un code de hachage ou d’une signature aux données en transit (SSL ou IPsec). |

### 2.3 Répudiation

La répudiation consiste à nier qu’une action s’est produite. Une offensive de répudiation a pour but de tenter de détruire, de masquer ou de modifier des preuves.

*Tableau 7 : Menaces - répudiation*

|  |  |
| --- | --- |
| Exemples | * Suppression de journaux. * Usurpation d’identités pour faire une demande de modification. |
| Erreurs courantes | * Processus d’autorisation et d’authentification insuffisant, voire inexistant. * Journalisation inadaptée. * Autorisation de présence d’informations sensibles sur des canaux de communications non sécurisés. |
| Solutions possibles | * Règles d’authentification strictes, d’audits, d’enregistrements de transaction, de journaux ou de signatures numériques. |

### 2.4 Divulgation d’informations

La divulgation d’informations consiste simplement à révéler publiquement des informations privées. La gravité d’une telle attaque varie selon la quantité et le degré de sensibilité des informations diffusées. La falsification de données est la capacité à modifier des informations divulguées.

*Tableau 8 : Menaces - divulgation de l’information*

|  |  |
| --- | --- |
| Exemples | * Vol de mots de passe. * Obtention d’informations de carte de crédit ou d’autres informations personnelles identifiables (PII) similaires. * Obtention d’informations sur le code source de l’application et/ou de ses systèmes hôte. |
| Erreurs courantes | * Possibilité pour un utilisateur authentifié d’accéder aux données d’autres utilisateurs. * Autorisation de présence d’informations sensibles sur des canaux de communications non sécurisés. * Choix inadapté des algorithmes et des clefs de chiffrement. |
| Solutions possibles | * Stockage non permanent des informations pendant une session. * Usage du hachage et du chiffrement chaque fois que possible. * Comparaison des données utilisateur avec l’authentification de l’utilisateur. |

### 2.5 Déni de service (dos)

Une attaque de type Déni de service (DoS) a pour effet de réduire la disponibilité normale d’une application. Les attaques DoS revêtent deux formes :

* L’invasion par surcharge, qui consiste à envoyer un nombre très élevé de messages pour faire tomber un serveur ;
* Le verrouillage, où la requête a pour effet de rendre les temps de réponse du serveur extrêmement longs en consommant des ressources ou en rendant les ressources indisponibles.

Ce type d’offensives peut se dérouler à n’importe quel niveau du modèle OSI. Elles sont relativement faciles à orchestrer, mais difficiles à contrer.

*Tableau 9 : Menaces - Déni de service (DoS)*

|  |  |
| --- | --- |
| Exemples | * Envoi d’un trop grand nombre de requêtes simultanées à l’application. * Envoi de requêtes qui provoquent le redémarrage de l’application ou des temps de réponse très longs. |
| Erreurs courantes | * Placement d’un trop grand nombre d’applications ou d’applications conflictuelles sur un seul serveur. * Test d’unités incomplet. |
| Solutions possibles | * Filtrage des paquets avec un pare-feu. * Usage d’un programme d’équilibrage de charge pour réduire le nombre de requêtes émanant d’une seule source. * Usage de protocoles asynchrones pour gérer les requêtes demandant des calculs intensifs avec une reprise des erreurs adaptée. |

### 2.6 Elévation des privilèges

Une augmentation des privilèges signifie que l’on reçoit les droits plus élevés que ceux qui sont normalement attribués.

*Tableau 10 : Menaces - élévation de privilèges*

|  |  |
| --- | --- |
| Exemples | * Un utilisateur se voit affecter des droits d’administration. * Un employé accède à un rôle de manager. |
| Erreurs courantes | * Exécution de processus du serveur Web avec les droits « root » ou « administrateur ». * Erreurs de codage permettant des dépassements de la mémoire tampon, plaçant l’application dans un état de débogage élevé. |
| Solutions possibles | * Usage d’un contexte de privilèges réduits au minimum chaque fois que possible. * Usage de langages sécurisés et d’options de compilateur pour éviter ou contrôler les dépassements du tampon. |

## SECTION3 : ANALYSE DES PROBLEMES DE SÉCURITÉ DES APPLICATIONS EXISTANTES

Le cycle de vie des erreurs liées à la sécurité des applications (apparition, détection, correction) est exactement identique à celui de n’importe quel autre type d’erreur liée aux applications. Ce livre blanc a pour principal thème l’amélioration des processus spécifiques à la sécurité utilisés lors de la création d’applications. (cf l’Annexe D : « Utilisation de CMMI pour améliorer la sécurité des applications »).

Plusieurs entreprises possèdent des applications et des systèmes anciens déjà déployés qui ont besoin de protection. Mais le travail d’évaluation et de gestion de l’infrastructure est exigeant, coûteux et doit s’effectuer de manière continue. Il existe de nombreux ouvrages et sources de référence sur le Web qui expliquent comment gérer de façon optimale la sécurité d’une application déjà en place. En fait, la plupart de ces références partent du principe qu’il n’ait pas la possibilité d’apporter des corrections à de telles applications.

### 3.1 Découverte et inventaire

* Effectuer un inventaire complet des systèmes et applications d’une entreprise : il s’agit de recueillez des informations techniques (IP, DNS, système d'exploitation utilisé, etc.) ainsi que des informations métier, par exemple identification de la personne ayant autorisé le déploiement ou de la personne à prévenir en cas d’arrêt de l’application.
* Rechercher les vulnérabilités et les failles de systèmes : se renseigner notamment sur les offensives connues susceptibles de menacer un système d'exploitation, un serveur Web et les autres produits tiers dont dépendent l’application. Ces offensives sont normalement publiées et sont immédiatement disponibles. Dans l’idéal, avant de déployer une application sur un serveur, il faut lui appliquer des correctifs, le renforcer et l’analyser.
* Rechercher les vulnérabilités des applications aux offensives connues : ici, il faut examiner notamment les requêtes HTTP utilisées par l’application et essayer de manipuler les données. Pour ce type d’évaluation, un test de type « boîte noire » est généralement utilisé.
* Tester la gestion de l’authentification des applications et des droits des utilisateurs.
* Arrêter tous les services inconnus.

### 3.2 Evaluation et analyse de la répartition des risques

* Évaluer les risques auxquels sont exposés les applications et systèmes. Prêter une attention particulière aux magasins de données, au contrôle des accès, au profilage et aux droits des utilisateurs.
* Établir un classement par priorités des vulnérabilités des applications détectées lors des évaluations. (cf Annexe E : Classification des risques avec le modèle Microsoft DREAD).
* Évaluez la conformité aux réglementations internes, sectorielles et officielles. Définir ce qui est acceptable et ce qui ne l’est pas.

### 3.3 Mesures de protection et contrôle des dommages

* Appliquer des correctifs, si disponible, à l’application et/ou à l’infrastructure.
* Il ne sera pas toujours possible de remédier aux problèmes de sécurité d’une application. Dans ce cas, il faut protéger au maximum le défaut de sécurité afin d’empêcher ou de réduire son exposition aux agressions. On peut mettre en place un pare-feu d’application pour la protéger, ou restreindre, désactiver ou relocaliser l’application.

### 3.4 Continuité de la surveillance et de l’analyse

* Les évaluations sont planifiées dans le cadre de processus documentés de gestion des modifications. Cette opération conclut le cycle en lançant une nouvelle phase de reconnaissance.

Dans ce chapitre, nous avons expliqué la revue de la littérature en présentant les concepts et définitions des concepts autour des défis de la sécurisation des applications, avec les menaces contre la sécurité des applications et une analyse des problèmes de sécurité des applications existantes

. Dans la suite, nous présenterons les tests ci-dessus cités dans le contexte des audits de sécurité appliques au SaaS, en présentant par la même les notions autour de ce type d’application.

# **CHAPITRE 2 : CADRAGE DU PROJET**

## SECTION 1 : CONTEXTE DU PROJET ET ETUDE DE L’EXISTANT

De manière globale, nous avons la cybernétique qui est l’étude des systèmes de commande et de communication entre êtres vivants, machines et autres. Et la sécurité est 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

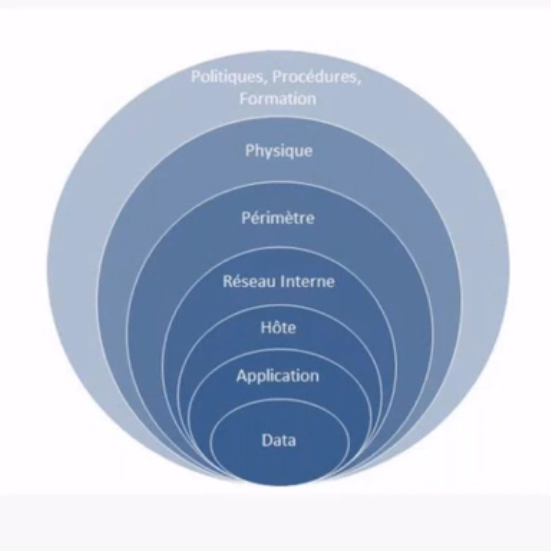


Figure 3 : Défense en profondeur (Source : google)

Généralement, les sociétés s’arrêtent au niveau de la couche hôte car le reste c’est plus chez eux, pour tant des choses doivent être faites (certaines seulement), ce que nous verrons dans le cas particulier des SaaS.

### 1.1 LE CLOUD COMPUTING ET LE SaaS

#### 1.1.1 Le cloud computing

Le cloud computing (*Informatique dans le nuage*) est une technique permettant de gérer des ressources (serveurs, …) et d’adapter très rapidement une infrastructure à des variations de charges de manière totalement transparente pour l’administrateur et les utilisateurs.

Les produits proposés en mode cloud computing ne se trouvent plus souvent sur un serveur informatique hébergé chez l’utilisateur, mais dans un nuage formé de l’interconnexion de serveur géographique distinct réalisé au niveau de ferme de serveurs géante (data center : *centre névralgique de traitement des données*). Ceci est rendu possible par le procédé de virtualisation qui consiste à faire fonctionner plusieurs systèmes d’exploitation ainsi que leurs applications associées sur un seul serveur physique. La virtualisation permet ainsi de recréer plusieurs applications virtuelles sur une seule et même machine physique. Ce qui offre au cloud la possibilité de se diversifier de manière typée comme décrit dans la suite.

#### 1.1.2- IaaS (Infrastructure As A Service)

Il s’agit de la mise à disposition à la demande, de ressources d’infrastructure, dont la plus grande partie est localisée à distance dans des data center.

L’IaaS permet l’accès aux ressources (SERVEURS) et à leurs configurations pour les administrateurs de l’entreprise. Le client a la possibilité de louer des clusters, de la mémoire ou du stockage de données. Le coût est directement lié au taux d’occupation.

##### 1.1.2.1 Avantages

* Grande flexibilité ;
* Contrôle total des systèmes (administration à distance par SSH ou remote desktop via le protocole UNDP) qui permet d’installer tout type de logiciel métier ;

##### 1.1.2.2 Inconvénient

* Besoin d’administrateur système comme pour les solutions de serveurs classiques sur site ;

**NB :** Les cibles sont les responsables d’infrastructures informatiques, comme Amazon EC2qui est un des principaux qui propose ce genre d’infrastructure.

#### 1.1.3 PaaS (Platform As A Service)

Il s’agit des plateformes du nuage regroupant principalement les systèmes virtualisés et leur SE. Le PaaS dispose d’environnement spécialisé au développement comprenant les langages en plus de pouvoir délivrer les logiciels en mode SaaS, les outils et les modules nécessaires. L’avantage de ces environnements est qu’ils sont hébergés par un prestataire basé à l’extérieur de l’entreprise. Ce qui permet de ne disposer d’aucune infrastructure et de personnel de maintenance (donc de pouvoir se consacrer au développement).

##### 1.1.3.1 Avantage

* Le déploiement est automatisé par des logiciels supplémentaire à acheter ou à installer ;

##### 1.1.3.2 inconvénients

* Limitation à une ou deux technologies (PYTHON ou JAVA pour *Google App Engine*, .Net pour *Ms Azure*, propriétaire pour Force.com) ;
* Pas de contrôle des machines virtuelles sous-jacentes ;
* Convient uniquement aux applications web ;
* Les cibles sont les développeurs ;

#### 1.1.4 SaaS (Software As A Service)

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

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.

##### 1.1.4.1 Gestion du cycle de vie d’un produit SaaS :

De manière générique la notion de cybersécurité doit être prise dans le cadre de la gestion du cycle de vie d’un produit SaaS, c’est à dire qu’à ce niveau nous avons parcouru que les solutions techniques, sauf qu’il n’est pas possible de déterminer les quelles devront être utilisées si tous les aspects du produit SaaS n’ont pas été pris en compte en amont, et si n’y a pas eu une analyse de ce qui doit être fait en termes de sécurité. Le danger étant que si l’entreprise ne va pas suffisamment loin, elle s’expose à la compromission, et aller trop loin explose le coup de développement et à ne plus être compétitif. Aussi, il faut bien prendre en compte :

* La contrainte sur le développement – secure by design et qualité logicielle
* La maîtrise de la fabrication
* Le déploiement sur le site
* La maintenance, maintien en condition de sécurité, mise à jour
* La fin de vie et destruction du produit SaaS

Tout ceci dans le cycle de vie va alors permettre de capitaliser les expériences et amener l’entreprise à avoir un produit SaaS qui soit au niveau de fiabilité attendu.



Figure 4 : Cycle de vie d’un produit SaaS (Source : google)

##### 1.1.4.2 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.

##### 1.1.4.3 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 au type d’application qui y sont hébergées, et dont les applications basées sur les microservices.

### 1.2 Application (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

#### 1.2.1 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.

#### 1.2.2 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.

### 1.3 Sécurisation d’une application SaaS

Concernant les réseaux informatiques, le modèle de référence OSI (Open System Interconnection) définit sept couches de protocoles réseau, avec pour chacune, une utilité bien définie. La couche qui nous intéresse est la couche 7 : la couche application, qui utilise des protocoles tels que HTTP permettant de transmettre des messages incluant du contenu de type HTML, XML, SOAP et des services Web. Les pares-feux classiques, par exemple, sont bien souvent inefficaces contre les attaques liées au protocole HTTP. Aussi, un attaquant de l’application, peut bien s’en servir passant par des ports connus, de sorte que les pares-feux réseau, de par leur conception, autorisent volontairement ce trafic pourtant nuisible, tout simplement parce que la couche réseau considère qu’il s’agit d’un bon trafic. En fait, le souci n’est pas tant la requête HTTP elle-même mais les données qu’elle contient. Souvent, ces données à risques sont entrées par l’utilisateur, spécialement formatées ou organisées afin de modifier le comportement des applications. Les offensives contre les applications peuvent par exemple autoriser un accès sans restriction aux bases de données, exécuter des commandes système arbitraires ou modifier le contenu d’un SaaS.

Une sécurité des applications bien pensée interdit à l’utilisateur mal intentionné de modifier le comportement d’une application. Les facteurs qui bien souvent engagent la sécurité insuffisante des applications SaaS sont :

* Les obligations en matière de sécurité des applications, pour peu qu’elles aient été définies, sont souvent perçues comme impossibles à mettre en pratique, bien elles sont dotées d’une connotation négative (interdictions), ou encore elles s’avèrent trop floues.
* Le test de la sécurité des applications est effectué uniquement en cas d’audit.
* Les équipes chargées de la conception et de la définition des exigences des applications considèrent que la sécurité concerne l’équipe réseau ou informatique.
* Les procédures de test standard concernent principalement le comportement fonctionnel.
* Seuls les quelques rares « spécialistes de la sécurité » de l’entreprise sont conscients des menaces pesant sur la sécurité des applications.
* Le test de la sécurité des applications se limite à de courts créneaux au cours desquels des pirates dits éthiques tentent de simuler ce que feraient les vrais pirates.

A ce stade, il nous importe de déterminer les particularités d’une application SaaS, et donc ce qui différencie sa sécurité de celle d’un SI par exemple. Aussi nous pouvons-nous citer quelques-unes de leurs contraintes techniques en matière de sécurité, a l’instar de :

* La (non)connaissance du contexte d’utilisation : les contextes d’utilisation des applications SaaS
* Puissance de calcul limitée
* La réalisation: la résolution de certains problèmes peut mettre en péril la sécurité d’une application SaaS, comme gérer le moment entre la résolution d’un problème de sécurité et sa prise en compte effective
* La gestion des logs : pour la maitrise de ce qui se passe et de ce qui s’est passée, comme :
  + La remontée des logs
  + L’exploitation des logs dans l’application SaaS
* Isolation physique
* Comportement en cas d’incident : La conduite à tenir en cas d’incident sur l’infra, sur lequel nous n’avez aucune autorité, alors la menace se propage
* La gestion de mise à jour : Pour une application déployée dans le monde, besoin est de disposer d’un système permettant de centraliser l‘ensemble, ce qui n’est pas nécessairement possible, car la mise à jour dépend très souvent des clients, qui même avec une mis à jour nécessaire pour un navigateur par exemple, on ne peut être sure qu’ils aient installée les mises à jour indiquées.
* La surveillance opérationnelle : Pour une application SaaS, nous avons des cas ou c’est un client au mieux, qui doit surveiller et déterminer le moyen de s’interfacer correctement avec elle ou de se dégager de la responsabilité qui va avec.

Nous pouvons alors retenir à ce niveau :

* L’intégrité d’une application SaaS avec :
  + L’accès sécurisé : pour le démarrage total ou partiel à l’application SaaS, ne doit démarrer que ce qui a été prévu
  + Le checksum et codes de contrôle : pour le code de contrôle qui permet de s’assurer que ce qui est en train de s’exécuter est bien ce que se doit de l’être, et que les processus qui s’exécutent sont toujours intègres
  + L’intégrité du comportement des liens de communication : pour s’assurer que les liens de communication sont corrects : utilisation de sondes de paramétrage (scriptes de vérification)
  + Le pprocessus de réalisation : qui participe à l’intégrité de l’application SaaS
  + L’intégrité des mises à jour
* L’authentification : Pour vréifier que celui qui fait quelque chose, est habilitée à le faire. S’il y a un accès root par exemple dans le produit SaaS, faire en sorte que le client puisse en changer le mot de passe, ou ne pas en vendre avec un mot de passe par défaut, mais un produit SaaS qui oblige à changer le mot de passe à la première connexion, ou un mot de passe diffèrent pour chaque produit SaaS. Avec :
  + Des certificats : En fonction de la technicité des produits SaaS, de leur puissance de calcul et de leur environnement, il faut avoir la possibilité de mettre en place les notions de certificat donc qui vont permettre d’identifier et d’échanger des clés de chiffrement
  + Stockage des identifiants : Pour stocker des identifiants
  + Lien aux services d’identification (LDAP) : pour se connecter à un service qu’on appelle des LDAP (service centralise chez le client ou là ou est déployé le produit SaaS) qui permet d’avoir une gestion centralisée des authentifications (à implémenter pour laisser l’opportunité au client de gérer sa propre confidentialité). Ici la société n’est pas maitresse de tout mais peut proposer des services, qui permettent d’amener à une bonne sécurité s’ils sont utilisés correctement
  + Authentification multi-facteur : Pour utilise autre chose qu’un mot de passe (compteur sur un smart phone qui donne un code qui change tout le temps et ainsi qu’un mot de passe, une empreinte biométrique...)

Cependant, il y en a bien d’autre encore, et tous ces éléments contribuent à assurer qu’il n’y a que ceux qui sont habilités à le faire, qui font des actions d’un genre ou d’un autre.

* Chiffrement des communications : pour stocker les secrets : les protocoles et les échanges sont basés sur des clés et des certificats que permettent le chiffrement et la consultation contrôlée de l’information, ainsi que de vérifier leur intégrité
  + L’authentification dans certains cas si les clés ont certaines particularités
  + Problèmes de stockage et de mise à jour de clés, dégradations des performances de débit de latence, avec des ssolutions matérielles comme : TPM, HSM, processeurs dédiés... En faisant cependant attention aux licences d‘export

Néanmoins, la connaissance et la mise en pratique de ces mesures de sécurités ne sont pas l’apanage en matière de sécurité, car des entreprises sont tout de même victimes d’attaque dans le grand internet. Il faut encore s’assurer de l’application effective de ces éléments de sécurité. Il convient alors de les évaluer du point de vue de la sécurité, mais surtout de la sécurisation, de manière à s’assurer qu’elles remplissent les conditions de sécurité avec par exemple un audit de sécurisation.

### 1.4 Audit de sécurité

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. L'auditeur dresse également une série de recommandations pour supprimer les vulnérabilités découvertes.

Un système d'information doit être protégé. Cependant tous les organismes n'ont pas besoin du même niveau de sécurité. Un contrat doit donc mentionner les secteurs à tester. 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.

#### 1.4.1 Procédure de l’audit de sécurité

Cette partie peut se découpe en plusieurs étapes, a 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.

#### 1.4.2 Les techniques d'audit de sécurité

##### 1.4.2.1 L'audit dit "boite blanche"

La méthode dite « white box » consistant à tenter de s'introduire dans le système en ayant connaissance de l'ensemble du système, afin d'éprouver au maximum la sécurité du réseau.

###### 1.4.2.1.1 Audit de code

Il existe des bases de vulnérabilités très fiables pour les applications répandues. Néanmoins, pour des applications moins utilisées, ou codées par l'entreprise elle-même, il peut être nécessaire d'analyser leur sécurité. Si les sources de l'application sont disponibles, il faut lire et comprendre le code source, pour déceler les problèmes qui peuvent exister. Notamment, les débordements de tampon (Buffer Overflow), les bugs de format, ou pour une application web, les vulnérabilités menant à des injections SQL... L'audit de code est une pratique fastidieuse et longue. De plus, elle ne permet généralement pas, en raison de la complexité, de dresser une liste exhaustive des vulnérabilités du code. Des méthodes automatiques existent, et permettent de dégrossir le travail, avec des outils comme RATS. Mais se reposer uniquement sur ce genre de méthodes peut nous faire passer à côté de problèmes flagrants pour un humain.

###### 1.4.2.1.2 Audit de configuration

Les audits de configuration permettent d'expertiser l'architecture technique déployée et de mesurer la conformité des configurations des éléments qui la composent (serveurs, bases de données, équipements réseau, pare-feu, autocommutateurs privés, etc.) avec la politique de sécurité définie. Ils en exposent les points faibles de l'architecture et se concentrent sur les actions à entreprendre pour mettre en œuvre un processus de sécurisation par couche. La réalisation d'un audit de configuration est par nature non destructrice (contrairement à certaines étapes d'un test d'intrusion). Ce type d’audit est destiné à toutes les entreprises.

###### 1.4.2.1.3 Audit de gestion des habilitations

Les audits de gestion des habilitations permettent d’analyser les accès aux ressources systèmes ou applicatives et impliquant des utilisateurs internes ou externes à l'entreprise. Ainsi, les comptes et les droits fantômes seront détectés, les actions de fraude seront rendues plus difficiles et la détection des actes malicieux sera facilitée. Ce type d’audit est destiné à toutes les entreprises.

###### 1.4.2.1.4 Audit déclaratif

Les audits déclaratifs permettent d’obtenir des résultats reposant uniquement sur les déclarations lors d'entretiens avec les acteurs du système audité : cela introduit un biais du au contrôle volontaire/involontaire des audités sur les informations délivrées. Ce type d’audit est destiné à toutes les entreprises.

###### 1.4.2.1.5 Audit organisationnel

Les audits organisationnels permettent de mesurer et d’identifier les risques des éléments critiques de l'entreprise (processus métier, outils de production dont le système informatique, ...). Ils représentent une optique à long terme. Ils sont importants pour préserver un niveau de sécurité dans le temps. Ils sont réalisés à l'aide de méthodes formelles telles Méhari, CRAMM, COBIT. Les audits organisationnels prennent en compte la sécurité en général dans l'entreprise. Cependant, c’est une démarche lourde qui peut mobiliser une équipe de consultants spécialisés durant plusieurs semaines, et elle est de ce fait rarement appliquée à des entreprises de taille plus modeste. Ce type d’audit est donc destiné aux grandes entreprises.

##### 1.4.2.2 L'audit dit "boite noire"

La méthode dite « black box » consistant à essayer d'infiltrer le réseau sans aucune connaissance du système, afin de réaliser un test en situation réelle.

###### 1.4.2.2.1 - Audit de vulnérabilités

Les audits de vulnérabilités permettent de détecter les éventuelles failles de sécurité du système d'information d’une entreprise tel qu'il peut être vu de l'extérieur, c'est à dire depuis

Internet. L'opération est possible à l'aide de scanners de vulnérabilités. Ceux-ci lancent des attaques connues sur le réseau cible (hormis celles qui pourraient neutraliser les systèmes évalués). L'avantage de ces tests tient à leur rapidité, leur simplicité de mise en œuvre et leur faible coût. De plus, les audits de vulnérabilités ne sont pas destructeurs. Leur inconvénient, bien sûr, est que ces tests ne détectent vraiment que les failles connues et plutôt simples à exploiter. Ce type d’audit est destiné aux PME.

###### 1.4.2.2.2 Test d’intrusion

Les tests d'intrusion permettent de valider périodiquement le niveau de sécurité du système d'information et d'en mesurer les variations. Ils sont réalisés de manière récurrente. Ils consistent à éprouver les moyens de protection d'un système d'information en essayant de s'introduire dans le système en situation réelle à partir de l’extérieur de l’entreprise. Ils ne permettent pas de garantir la sécurité du système, dans la mesure où des vulnérabilités peuvent avoir échappé aux testeurs. Ce type d’audit est destiné à toutes les entreprises.

###### 1.4.2.2.3 Audit technique

Les audits techniques permettent d’évaluer le niveau de sécurité par analyse interne des systèmes en place. On se place dans du court terme, pour mettre à niveau la sécurité dans l'urgence. Ils permettent d’étudier les éléments techniques en production dans l’entreprise et d’en valider le niveau de sécurité. Il s'agit d'une prestation hautement technique, dont la plupart des PME peuvent très bien se passer. Ce type d’audit est donc destiné aux grandes entreprises.

###### 1.4.2.2.4 Le Fuzzing

Pour les applications boite noire, où le code n'est pas disponible, il y a l'analyse de code, qui est le fuzzing. Cette technique consiste à analyser le comportement d'une application en injectant en entrée des données plus ou moins aléatoires, avec des valeurs limites. Contrairement à l'audit de code qui est une analyse structurelle, le fuzzing est une analyse comportementale d'une application.

Nous pouvons donc retenir ici, qu’il importe de se tenir au courant des dernières vulnérabilités. Ainsi, s'abonner à des mailing-list (securityfocus.com, seclists.org) et consulter fréquemment des sites de veille sont des moyens efficaces d'être informé.

## SECTION 2 : CAHIER DE CHARGE

### 2.1 Intérêt et objet de la mission

L’intérêt de notre travail se perçoit clairement en ce qu’un produit logiciel qui rencontre la confidentialité des données, leur intégrité et leur disponibilité permanentes pour un client, en garantissant de bons résultats dans les meilleurs délais pour sécurité optimale, sont autant de facteurs qui contribuent 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 réalisé un audit sécurité sur une des applications SaaS d’une une entreprise dont LAO sarl, conformément aux dispositions du décret N°2004-1250 du 25 mai 2004 et aux dispositions du présent cahier des charges. Cette mission doit être pilotée par, un chef de projet ayant des aptitudes en sécurité informatique, conformément au décret n°004-1249 du 25 mai 2004. Elle est annuelle et renouvelable par tacite de reconduction avec une durée maximale de trois (3) ans. Cet audit prend comme référentiel de base la norme ISO/CEI 27002 et suit une approche méthodologique aussi proche que possible de ce référentiel.

Généralement, ceci concerne les aspects organisationnels, physiques et techniques relatives à la sécurité de l’application SaaS inclus dans le périmètre de cet audit, mais il se restreint dans le ce cadre a la couche application comme nous le verrons dans la suite de notre travail.

Ainsi, pour la réalisation de notre mission, nous avons opté pour une approche méthodologique, en indiquant les références de la méthodologie adoptée, tout en gardant comme référentiel normatif la norme ISO/CEI 27002.

### 2.2 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 (section1 – 1.4). 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.

### 2.3 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.

#### 2.3.1 Phase 1 : Préparation de la mission

##### 2.3.1.1 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.

##### 2.3.1.2 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.

##### 2.3.1.3 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.

#### 2.3.2 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.

##### 2.3.2.1 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.

##### 2.3.2.2 Volet audit technique

###### 2.3.2.2.1 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.

###### 2.3.2.2.2 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.

2.3.2.3 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 :

###### 2.3.2.3.1 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.

###### 2.3.2.3.2 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.

#### 2.3.3 Phase 3 : Synthèse des recommandations - Sensibilisation post-audit

##### 2.3.3.1 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.

##### 2.3.3.2 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.

### 2.4 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 : METHODOLOGIE ET OUTILS UTILISES

Les applications réalisées au sein de LAO sarl sont de type microservice et dans le cadre de notre audit nous avons opté pour le pentesting qui est une méthode qui ne cesse de montrer son efficacité dans le processus de sécurisations des applications.

### 3.1. Les tests d’intrusion : pentesting

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.

Ceux-ci diffèrent 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.

#### 3.1.1. Historique

Le besoin en matière de protection l’information, a toujours été une préoccupation importante pour les organismes, et ce qu’importe leur nature (publique 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.

#### 3.1.2. 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 organisationnel, 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…).

#### 3.1.3. Les vecteurs d’attaque

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 1 – 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. |

De manière spécifique et en fonction de ces caractéristiques, nous avons par exemple comme vecteurs d’attaque :

* Les serveurs : Internet Information Services (IIS), Apache et ngnix,
* Les services de noms : Active Directory (AD) et BIND,
* Les services de messagerie tels que Microsoft Exchange, Dovecot, Postfix, Sendmail, Lotus Notes,
* Les services et protocoles d’accès distant : Point-to-Point Tunneling Protocol (PPTP), Layer 2 Tunneling Protocol (L2TP), Internet Protocol Security (IPSec) et OpenVPN.

### 3.2. Les stratégies de tests

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.

#### 3.2.1. 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 2 – 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. |

#### 

#### 3.2.2. 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 3 – 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. |

#### 

#### 3.2.3 Le 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 4 – 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. |

### 3.3. Les étapes d’un test d’intrusion

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*

#### 3.3.1. 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.

#### 3.3.1.1 Considérations relatives aux prestataires de services

Dans un organisme, les pentests sont souvent réalisés avec l’aide des prestataires de services spécialisés pour un soutien, en vue d’une optimisation du test. Lors de l’étape de planification, afin qu'il puisse élaborer une stratégie optimale de tests. Dans ce cas, la consultation avec les détenteurs de systèmes et les principaux intervenants en sécurité de l’information de l’organisme permettront de me circonscrire les systèmes à tester.

En outre, un organisme qui désire faire affaire avec un prestataire, devra prendre en compte plusieurs éléments lors de la préparation des devis. Plusieurs considérations au niveau des compétences techniques requises et des solutions logicielles potentielles sont citées dans le chapitre 4. Aussi, Il est important, de bien définir les rôles, les responsabilités ainsi que les règles d’engagement des parties, lors de l’élaboration du document d’appel d’offres, du contrat ou de l’entente. Ce qui sera décrit dans la suite.

##### 3.3.1.2. 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. Nous proposons plusieurs considérations en matière de compétences techniques de l’équipe de réalisation dans le chapitre4.

Tableau 5 – 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. |

##### 3.3.1.3. 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.

#### 3.3.2. Découverte

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

##### 3.3.2.1. Types de vulnérabilités

Nous avons pu catégoriser les vulnérabilités par type. De cette manière, nous avons pu entre 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 7 – 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).

##### 3.3.2.2. 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.

##### 3.3.2.3. 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 analyser 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 comme dans l’exemple de l’annexe V.

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 décrites au chapitre 4.

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.

#### 3.3.3. 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.



Figure 3 – Les sous-étapes de l’exploitation

##### 3.3.3.1 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.

##### 3.3.3.2. 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.

##### 3.3.3.3. 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.

##### 3.3.3.4. 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.

##### 3.3.4. 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.

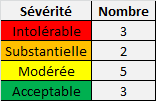
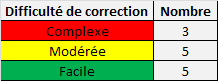
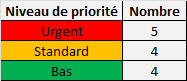
Le rapport se présente de façon diverse suivant le responsable en charge de la réalisation du pentest (prestataire ou autre). Cependant, il est souvent formaté de la manière suivante :

* **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.



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.

# **PARTIE II : ANALYSE, SPECIFICATION ET IMPLEMENTATION**

# **CHAPITRE 3 : ANALYSE ET SPECIFICATION**

# **CHAPITRE 4 : RESULTATS OBTENUS ET DISCUTION**

# **CONCLUSION GENERALE**

# **Annexes**

# **Table de matières**

# **Références bibliographiques**