# **SOMMAIRE**

[**SOMMAIRE** 1](#_Toc75097742)

[**LISTE DES FIGURES** 2](#_Toc75097743)

[**LISTE DES TABLEAUX** 3](#_Toc75097744)

[**INTRODUCTION GENERALE** 4](#_Toc75097745)

[**CHAPITRE I : REVUE DE LA LITTERATURE** 6](#_Toc75097746)

[SECTION1 : CONCEPTES ET DEFINITIONS 6](#_Toc75097747)

[SECTION2 : LES MENACES CONTRE LA SÉCURITÉ DES APPLICATIONS 9](#_Toc75097748)

[SECTION3 : ANALYSE DES PROBLÈMES DE SÉCURITÉ DES APPLICATIONS EXISTANTES 13](#_Toc75097749)

[SECTION4 : RÉSOLUTION DES ERREURS PENDANT TOUT LE CYCLE DE VIE DU DÉVELOPPEMENT LOGICIEL 14](#_Toc75097750)

[**CHAPITRE 2 : CADRAGE DU PROJET** 19](#_Toc75097751)

[SECTION 1 : CONTEXTE DU PROJET ET ETUDE DE L’EXISTANT 19](#_Toc75097752)

[SECTION 2 : CAHIER DE CHARGE 33](#_Toc75097753)

# **LISTE DES FIGURES**

[Figure 1 : Sécurité des applications Web : les enjeux les plus courants (source : site IBM) 8](#_Toc74898328)

[Figure 2 : Illustration graphique (source : Rapport enquête réalisée par Sanctum - IBM) 14](#_Toc74898329)

# **LISTE DES TABLEAUX**

[Tableau1 : Défis - assurance 6](#_Toc75098150)

[Tableau2 : Défis - Finance 7](#_Toc75098151)

[Tableau3 : Défis – Industrie pharmaceutique 7](#_Toc75098152)

[Tableau4 : Défis – Industrie pharmaceutique 8](#_Toc75098153)

[Tableau5 : Menaces - usurpation d’identité 10](#_Toc75098154)

[Tableau 6 : Menaces - falsification 10](#_Toc75098155)

[Tableau 7 : Menaces - répudiation 11](#_Toc75098156)

[Tableau 8 : Menaces - divulgation de l’information 11](#_Toc75098157)

[Tableau 9 : Menaces - Déni de service (DoS) 12](#_Toc75098158)

[Tableau 10 : Menaces - élévation de privilèges 12](#_Toc75098159)

[Tableau 11 : Coûts relatifs calculés en fonction du délai écoulé entre l’apparition de l’erreur et sa détection 14](#_Toc75098160)

[Tableau 12 : Avantages et inconvénients – Tests manuels 17](#_Toc75098161)

[Tableau 13 : Avantages et inconvénients – Tests automatiques 17](#_Toc75098162)

[Tableau 14 : Avantages et inconvénients – Test boite blanche 18](#_Toc75098163)

[Tableau 15 : Avantages et inconvénients – Test boite grise 18](#_Toc75098164)

# **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.

# **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.

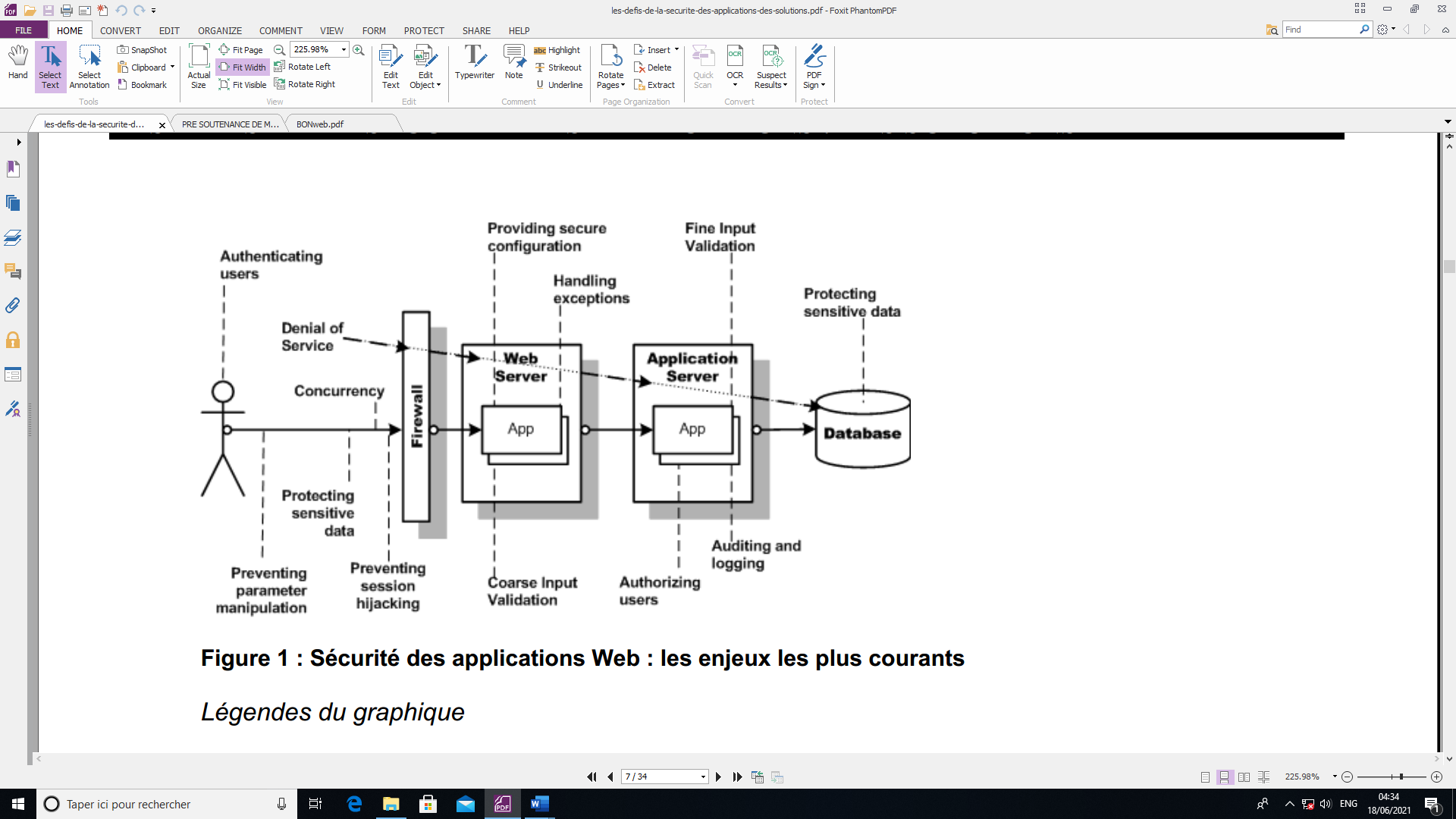


Figure 1 : Sécurité des applications Web : les enjeux les plus courants (source : site IBM)

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 PROBLÈMES 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.

## SECTION4 : RÉSOLUTION DES ERREURS PENDANT TOUT LE CYCLE DE VIE DU DÉVELOPPEMENT LOGICIEL

### 4.1 Coût de la correction des erreurs dans le cycle de vie du développement logiciel

Toutes les applications sont soumises à des pressions au cours du cycle de vie du développement logiciel (SDLC), telles que les délais de mise au marché à respecter pour rester concurrentiel, une complexité croissante, une augmentation des risques métier. Plus les erreurs de l’application passent inaperçues pendant longtemps, plus les coûts augmentent, et parfois en flèche. Le Tableau (?)7 est extrait d’une enquête NIST de 2002 (lien direct : <http://www.nist.gov/director/prog-ofc/report02-3.pdf>).

Tableau 11 : Coûts relatifs calculés en fonction du délai écoulé entre l’apparition de l’erreur et sa détection

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Erreur**  **détectée à la**  **conception** | **Erreur**  **détectée**  **au codage** | **Erreur**  **détectée à**  **l’intégration** | **Erreur détectée**  **dans la version**  **bêta** | **Erreur détectée**  **dans la version**  **finale approuvée** |
| **Erreurs de conception** | 1x | 5x | 10x | 15x | 30x |
| **Erreurs de codage** |  | 1x | 10x | 20x | 30x |
| **Erreurs d’intégration** |  |  | 1x | 10x | 20x |

Lorsqu’une erreur de conception est détectée lors de la version finale approuvée d’une application, le coût nécessaire pour la corriger peut aller jusque 30 fois à ce qu’il aurait été si l’erreur avait été corrigée lors de la phase de conception. Et encore, ce coût ne correspond qu’au coût de la correction par l’équipe responsable de l’application : il n’intègre pas les autres facteurs tels que la perte d’une part de marché, le ternissement de la réputation de l’entreprise ou le degré de satisfaction des clients.

Une enquête réalisée par Sanctum (racheté par Watchfire en 2004) et portant sur plus de 100 applications utilisées par les sites de grandes entreprises et d’organismes publics révèle quelques chiffres très dérangeants sur le taux de non-conformité à la sécurité. L’enquête a ainsi constaté que 92 % de toutes ces applications échouaient aux tests de sécurité effectués lors des phases d’intégration ou de production. La durée moyenne de résolution des erreurs était estimée à 2 mois et demi, et le coût pour l’équipe responsable avoisinait les $25 millions. Lors d’un nouveau test des applications défaillantes, 20 % des tests de sécurité (soit 16 % du total) se sont soldés par un nouvel échec. La moitié des applications ayant essuyé un second échec (8 % au total) n’ont jamais réussi ces tests.

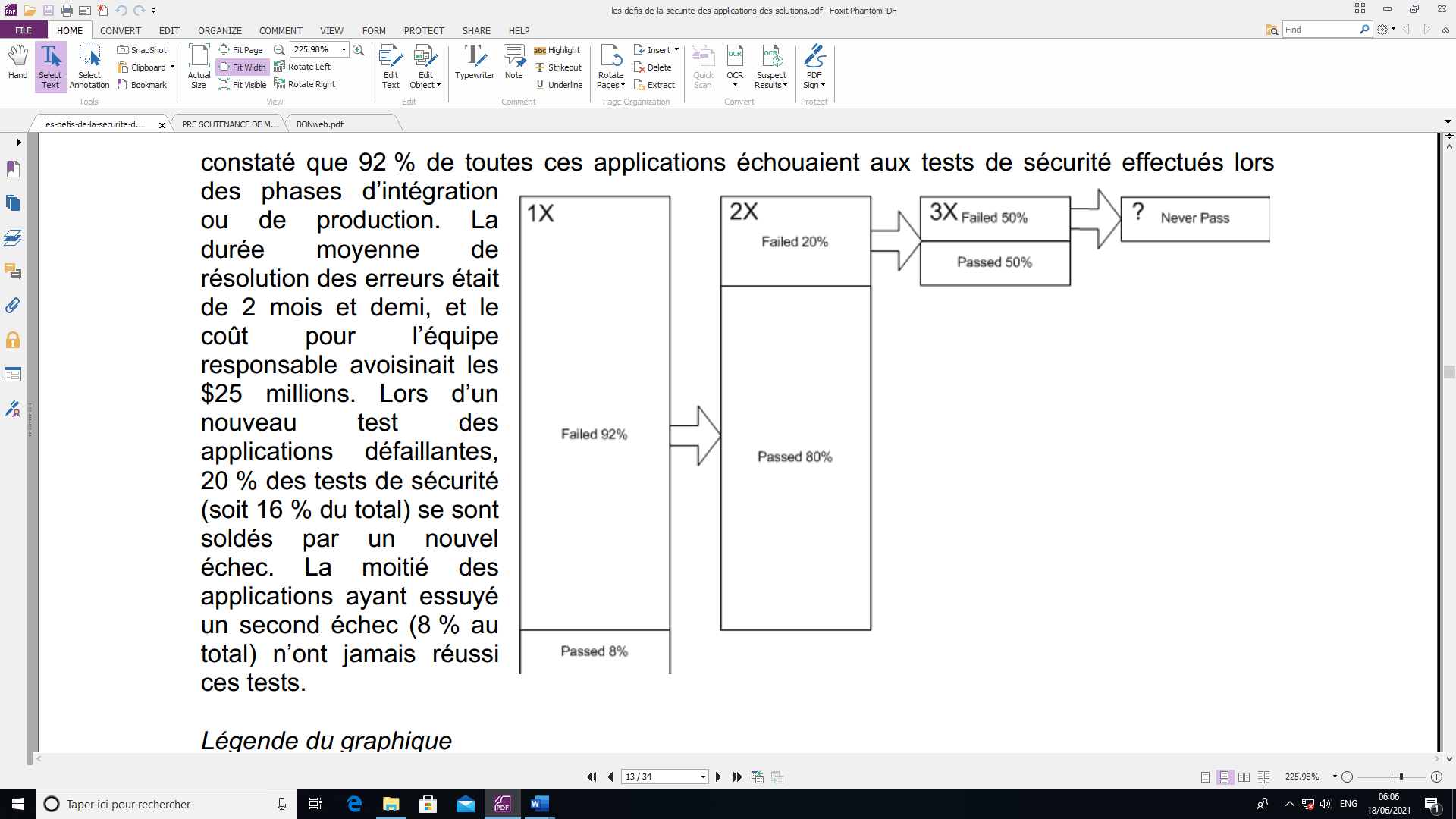


Figure 2 : Illustration graphique (source : Rapport enquête réalisée par Sanctum - IBM)

Légende du graphique :

* Failed 20% = Échec 20 %
* Failed 50% = Échec 50 %
* Passed 50% = Réussite 50 %
* Never Pass = Échec définitif
* Failed 92% = Échec 92 %
* Passed 8% = Réussite 8 %
* Passed 80% = Réussite 80 %

Compte tenu de la forte probabilité d’échec aux tests de sécurité du coût représenté par une détection tardive d’erreurs lors du cycle de vie du développement logiciel, il apparaissait donc vivement conseillé d’améliorer les tests de sécurité pendant tout ce cycle et de dépister très tôt les erreurs de sécurité.

### 4.2 Types d’erreurs rencontrées dans le cycle de vie du développement logiciel

Avant d’aborder les moyens de détection et de correction des erreurs, nous allons passer en revue les différents types d’erreurs que nous tentons de corriger, en fonction du moment auquel elles apparaissent dans le cycle de vie du développement logiciel.

#### 4.2.1 Erreurs au stade de la définition des exigences

L’équipe responsable des exigences dispose en général d’un tableau global des critères de sécurité que son application va devoir respecter. Mais si cette équipe ne connaît pas les menaces pesant sur la sécurité des applications, il y a peu de chances qu’elle communique les exigences spécifiques à l’application aux concepteurs, notamment la nécessité de rechercher les menaces ciblant cette application en particulier.

4.2.2 Erreurs au stade de la conception

Pour un ensemble d’exigences approprié, il faut déterminer à quel moment l’équipe de conception peut-elle faire fausse route. De plus, cette équipe doit être sensibilisée aux menaces liées à la sécurité de l’application, avec un choix technologique satisfaisant ou adapté rassurer les concepteurs, de ce qu’ils ont pris les bonnes mesures pour répondre à une exigence de sécurité particulière. Surtout qu’une connaissance insuffisante de la sécurité signifie que le système de test va être incomplet ou inefficace, voire inexistant.

#### 4.2.3 Erreurs au stade du codage

Pour une entreprise de développement, avec une conception saine à partir de laquelle ses développeurs vont pouvoir travailler, il faut déterminer les erreurs qui risquent s’introduire. En général, l’erreur se produit si, au lieu de rédiger du code entièrement nouveau, ils réutilisent du code comportant des imperfections, ou génèrent du code à l’aide d’un assistant IDE non sécurisé. Le danger étant d’effectuer une validation des données incorrecte ou de ne pas utiliser correctement les fonctions de sécurité du système choisi pour l’application.

#### 4.2.3 Erreurs se produisant à un stade ultérieur

Dans la plupart des entreprises, les connaissances en matière de sécurité des applications sont souvent détenues par une poignée de collaborateurs membres d’équipes d’audit de sécurité chargées de réaliser des intrusions ou des piratages volontaires pour tester les défenses de l’entreprise. Ces équipes opèrent dans des créneaux limités et examinent l’application vers la fin du cycle de vie du développement logiciel, dans l’espoir d’intercepter des erreurs de sécurité avant la commercialisation de l’application.

Aussi, la centralisation de cette fonction est due au fait qu’un pirate éthique compétent est rare (comprendre cher), et que ces équipes réalisent souvent des audits commandés de systèmes de production. Comme nous le verrons, une telle approche constitue un goulot d’étranglement, est coûteuse et ne permet pas de détecter toute la gamme d’erreurs de sécurité des applications.

### 4.3 Approches générales du test de la sécurité des applications

Les erreurs de sécurité des applications ne sont que des erreurs. Même s’il est nécessaire de mettre en œuvre des compétences spécifiques pour créer les tests de détection des erreurs de sécurité, lorsque ce type d’erreur est décelé, le processus de correction est entièrement identique à celui employé pour remédier aux autres erreurs.

#### 4.3.1 Tests manuels

Les tests d’infiltration ou de sécurité sont souvent effectués par une petite équipe de spécialistes. Ils font souvent appel à des outils et des scripts connus.

Tableau 12 : Avantages et inconvénients – Tests manuels

|  |  |
| --- | --- |
| Avantages | * Génération de tests bien ciblés adaptés à des fonctions d’application spécifiques. |
| Inconvénients | * Les tests de sécurité sont effectués par un nombre restreint de spécialistes. * Risque d’erreurs humaines. * Poste de dépense à récurrence très fréquente. * Les contraintes de temps limitent les applications couvertes. |

#### 4.3.2 Tests automatiques

Les tests automatiques sont en général conçus de l’une des deux façons suivantes :

* Une approche de bas en haut, le développeur de code créant spécialement des tests destinés à des fonctions ou des méthodes données ;
* Une approche de haut en bas, les tests étant conçus par les équipes d’assurance qualité du point de vue de l’utilisateur final.

La création et la maintenance des tests automatiques nécessitent des investissements plus élevés que les tests manuels. Ces frais sont normalement compensés par des améliorations au niveau de la qualité, la diminution des tests de réception, et l’amélioration des processus de développement itératif.

4.3.3 Le test de la boîte noire

Le test de la boîte noire, parfois appelé test système, est une approche de haut en bas. Il part du principe que les rouages internes de l’application sont inconnus. La connaissance de l’application se limite à en examiner les entrées et les sorties. Il s’agit de la forme la plus courante des tests de sécurité, utilisée par les responsables d’audit, les testeurs manuels et les pirates éthiques. Le ou les tests consistent à modifier des entrées utilisateur « normales » pour amener l’application à se comporter de façon inattendue.

Tableau 13 : Avantages et inconvénients – Tests automatiques

|  |  |
| --- | --- |
| Avantages | * Test ne nécessitant que peu de connaissances, voire aucune, de l’application. * Outils bien rodés d’automatisation des tests. |
| Inconvénients | * Le test peut uniquement être exécuté lorsque toutes les composantes de l’application sont prêtes à être testées (en général dans un environnement avancé de transfert ou de production). * Des modifications des entrées utilisateur peuvent provoquer un grand nombre de transactions. Ne pas tenir compte des résultats de ces transactions ou parvenir à les annuler est souvent problématique pour les systèmes de production. * En raison de la visibilité limitée de l’application, des défauts peuvent passer inaperçus. |

#### 4.3.4 Le test de la boîte blanche

Le test de la boîte blanche, parfois appelé « test du source », teste chacun des composants de d’une application. Souvent, ce test est effectué au niveau d’une méthode ou d’une fonction. Il est exécuté de façon à déceler des erreurs de fonctions et est souvent associé à des outils d’analyse de code et soumis à des examens par des homologues.

Tableau 14 : Avantages et inconvénients – Test boite blanche

|  |  |
| --- | --- |
| Avantages | * Reconnaissance bien définie des défauts des fonctions testées * Intégrations bien rodées aux environnements IDE des développeurs. |
| Inconvénients | * Ce type de test, portant sur la source, ne reconnaît pas les défauts liés aux exigences et à la conception. * Reconnaissance médiocre des erreurs de sécurité, car de nombreuses offensives portent sur plusieurs composants ou ont une durée particulière non prise en charge par le test unitaire. * Les tests sont souvent écrits par l’auteur du code. Or, si le développeur n’est pas formé à la sécurité, il ne peut savoir de quels tests il a besoin. |

#### 4.3.5 Le test de la boîte grise (utilisation d’un système de test défini par l’application)

Ce système de test associe à la fois des tests de type boîte noire et boîte blanche.

Généralement, on utilise un tel système pour créer l’état de l’application et le test d’événement qui ne sont pas disponibles dans les outils de test disponibles dans le commerce (cf Annexe F :

« Test de sécurité déterminé par les événements », donne un exemple possible d’un tel système).

Tableau 15 : Avantages et inconvénients – Test boite grise

|  |  |
| --- | --- |
| Avantages | * C’est la méthode la plus complète, associant à la fois un test au niveau du système et au niveau de l’unité. * Les tests déterminés par les actions fournissent des tests basés sur l’état et le moment d’occurrence. * Les agents et/ou les proxys peuvent être utilisés pour des tests basés sur les causes et les effets. * Le système de test peut être créé pour permettre des tests d’audit en production sans impact sur les données de production. * Le système de test peut surveiller les flux de données via l’application. |
| Inconvénients | * La méthode doit être définie dans le cadre des phases de définition des exigences et de conception. * Le travail nécessaire à la création d’un système de test est souvent aussi important que l’application à tester. |

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, ainsi que quelques approches de résolution des erreurs pendant le cycle de vie d’une application. 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 règlementaires 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 maitriser 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.

* 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?
* 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

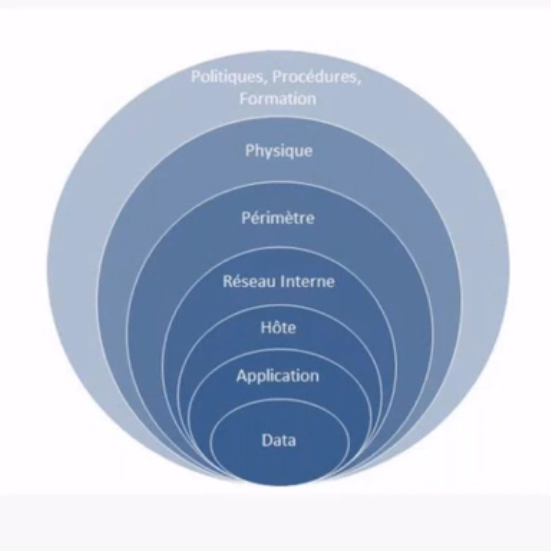


Figure 3 : Défense en profondeur(Source : google)

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 :

* Contrainte sur le développement – secure by design et qualité logicielle
* Maitrise de la fabrication
* Déploiement sur le site
* Maintenance, maintien en condition de sécurité, mise à jour
* 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 Objet de la mission

LAO sarl qui est une société fait dans la réalisation d’applications SaaS, dont quelques-unes sont : Viwanda, FRET, Surpervisor qu’on peut retrouver sur leur site (www.lao.cm) se propose d’effectuer une mission d’audit de la sécurité de ses applications 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. 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 le 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 securite, 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 decompose en cinq phases :

#### 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 oeuvre doivent être examinés et validés. Ces réunions déboucheront, entre autre, sur la finalisation des plannings précis et détaillés de mise en oeuvre de la mission. Quant aux résultats de cette réunion, il 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 et
* 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 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 autres tests 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 oeuvre,
* Evaluer, 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 oeuvre 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.

Diagramme de Gantt

Le diagramme de GANTT est un outil permettant de modéliser la planification des

tâches nécessaires à la réalisation d’un projet, étant donné la relative facilitée de lecture des

diagrammes GANTT, cet outil est utilisé par la quasi-totalité des chefs de projet dans tous les

secteurs. Le diagramme de GANTT représente un outil pour le chef de projet, permettant de

représenter graphiquement l’avancement du projet, mais c’est également un bon moyen de

communication entre les différents acteurs d’un projet.

Figure 2. 2: Succession de tâches avec Gantt

Sources : Par nos soins

Evaluation des coûts

Le CFTL (Comité Français de Test Logiciel) et L’ISTBQ (International Software Testing

Qualification Board) évaluent les coûts des logiciels en fonction de la criticité de celui-ci :

Le coût d’un logiciel critique est estimé à plus de cinquante pour cent (50%) du coût

du développement du dit logiciel.

Le coût d’un logiciel standard est évalué à environ trente pour cent (30%) du coût total

de développement. [Glossaire CFTL/ISTQB]

5. DÉFINITION D’UNE STRATÉGIE