

Secteur Tertiaire Informatique  
Filière « Etude et développement »

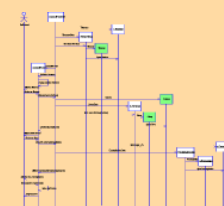
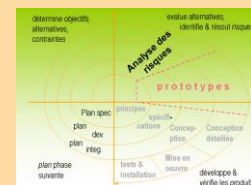
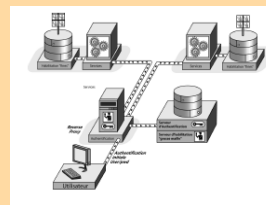
Séquence « Concevoir une application »

Identifier les exigences de sécurité de l'application

Apprentissage

Mise en situation

Evaluation



Version	Date	Auteur(s)	Action(s)
1.0	19/10/16	Lécu Régis	Création du document

Identifier les exigences de sécurité de l'application

Afpa © 2016 – Section Tertiaire Informatique – Filière « Etude et développement »

## TABLE DES MATIERES

Table des matières .....	2
1. Introduction .....	5
2. Enjeux.....	5
3. Principes de l'analyse de sécurité .....	6
3.1 Compléter l'analyse par une approche sécurité.....	6
3.2 Prendre le point de vue de l'attaquant.....	8
3.3 Repérer les biens essentiels .....	8
4. Le recensement des exigences ( <i>Requirements Elicitation</i> ) .....	10
4.1 <i>Misuse / Abuse Cases</i> .....	10
4.2 La modélisation des menaces .....	12
4.3 Glossaire.....	12
4.4 Etude de cas .....	13
5. Analyse de risque d'entreprise .....	16
5.1 Présentation succincte de la méthode EBIOS .....	16
5.2 Etude de cas .....	16

## Objectifs

A l'issue de cette séance, le stagiaire sera capable de :

- Identifier les exigences de sécurité de l'application dès la phase de spécification, en adoptant un regard sécurité, parallèlement à la spécification fonctionnelle.
- Prendre en compte une analyse de risque de l'entreprise, réalisée en amont par un expert sécurité (optionnel).

## Pré requis

Cette séance suppose que le stagiaire soit sensibilisé au développement sécurisé (par les séances précédentes du projet CyberEdu), et pratique une méthode d'analyse (Merise ou UML).

## Méthodologie

Ce document peut être utilisé en présentiel ou à distance.

Il précise la situation professionnelle visée par la séance, la situe dans la formation, et guide le stagiaire dans son apprentissage et ses recherches complémentaires.

## Mode d'emploi

Symboles utilisés :



Renvoie à des supports de cours, des livres ou à la documentation en ligne constructeur.



Propose des exercices ou des mises en situation pratiques.



Point important qui mérite d'être souligné !

## Ressources

- Guide d'introduction à l'analyse des exigences de sécurité, en anglais :  
*RequirementAnalysis\_PocketGuide.pdf*
- Article sur les *Misuse Cases* :  
*misuse.pdf*
- Documentation sur la méthode EBIOS :  
dossier EBIOS

## 1. INTRODUCTION

Cette séance fait partie du projet **CyberEdu**, qui prend en compte la sécurité dans tout le cycle de vie du logiciel, et dans toutes les couches des applications.

Les séances précédentes étaient centrées sur les étapes de conception et de codage. Cette séance se situe en amont : spécification (ou analyse) du projet.

L'identification des exigences de sécurité doit être effectuée le plus tôt possible dans le cycle de vie du projet, en même temps que l'analyse fonctionnelle. Dans le cas d'un développement agile, chaque itération comporte une phase d'analyse et de conception, et de nouvelles exigences de sécurité peuvent apparaître pendant toute la durée du projet.

Les exigences de sécurité de l'application sont souvent négligées dans l'analyse car :

- elles relèvent rarement d'exigences fonctionnelles explicites ;
- elles sont parfois difficiles à identifier ;
- elles ont un coût non négligeable.

Parallèlement aux spécifications fonctionnelles, l'analyste doit donc mener un questionnement spécifique, centré sur les besoins de sécurité :

- le cahier des charges définit-il des exigences explicites de sécurité : disponibilité d'un service critique, intégrité d'une information, confidentialité des données, traçabilité ?
- dans quel contexte doit-on protéger l'intégrité et la confidentialité des données : sur le réseau, dans leur stockage, pendant leur traitement ?
- le logiciel est-il multiutilisateur ? Existe-t-il des droits distincts selon les utilisateurs ? etc.

Dans cette séance, nous allons suivre le guide « *Requirements Analysis for Secure Software* » qui fait partie d'une série traitant de tous les aspects du développement sécurisé (libre de droit). Ce guide est un abrégé (*Pocket Guide*) qui renvoie à de nombreuses documentations en ligne complémentaires.

## 2. ENJEUX

Plus les erreurs sont découvertes tard dans le cycle de développement, plus elles coûtent cher. Les failles de sécurité ne font pas exception et peuvent être particulièrement coûteuses.

Il faut recenser toutes les exigences pour réussir le développement d'un système, mais trop souvent, ces exigences ne prennent pas en compte explicitement la sécurité.

En conséquence, des systèmes complètement fonctionnels sont rarement sécurisés et peuvent être victimes d'attaques. Inversement, les systèmes qui documentent soigneusement les exigences de sécurité réduisent la probabilité de succomber à ces attaques.

Les exigences de sécurité concernent les fonctionnalités qui doivent mettre en œuvre une politique de sécurité, dans tous les domaines du développement sécurisé :

- contrôle d'accès : identification, authentification, autorisation ;
- chiffrement et déchiffrement ;
- gestion des clés etc.

Ces fonctionnalités protègent les propriétés de sécurité du système et ses données, contre les accès non autorisés, modifications non autorisées, déni de service et divulgation des données.

Identifier les exigences de sécurité de l'application

Des exigences de sécurité complètes, explicites, mesurables et testables produiront des systèmes plus sûrs.

### 3. PRINCIPES DE L'ANALYSE DE SECURITE

Les exigences logicielles renvoient à des fonctionnalités ou des contraintes, définies positivement : « le système doit avoir cinq entrées ».

Une exigence logicielle est définie traditionnellement comme « une propriété du système qui répond et permet de résoudre un problème du monde réel ».

Il y a plusieurs définitions des exigences de sécurité, mais elles renvoient soit à des contraintes sur les exigences fonctionnelles, soit à des propriétés que le système doit conserver dans tous les cas (invariant) :

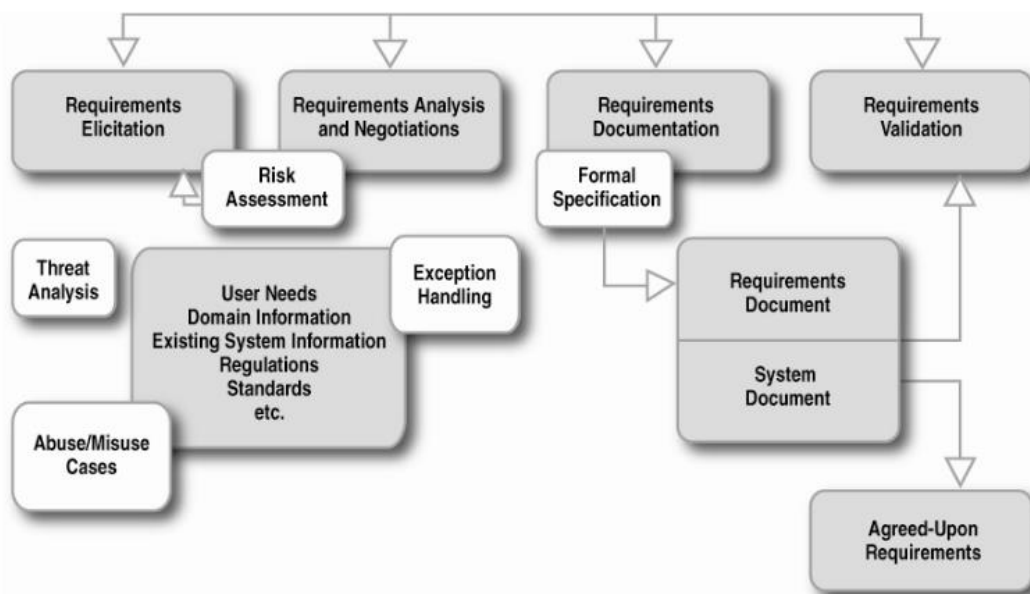
- dans le premier cas, elles découlent de l'analyse fonctionnelle et sont comparables aux autres contraintes, comme la sûreté ou le coût ;
- dans le deuxième cas, elles sont complémentaires et décrivent l'état du système au-delà de son utilisation normale (attaques etc.) : elles décrivent par des assertions, les propriétés qui restent vérifiées.

### 3.1 COMPLETER L'ANALYSE PAR UNE APPROCHE SECURITE

Il existe beaucoup de techniques et d'outils pour définir les exigences logicielles. Mais la plupart ne prennent pas en compte la sécurité : ils définissent les exigences du système en se concentrant sur ses fonctionnalités, sans considérer ce qu'il ne doit pas faire !

Hors, les utilisateurs font des hypothèses implicites sur leurs applications et leur système. Ils s'attendent à ce que leur application soit sécurisée et sont surpris si elle ne l'est pas. Les suppositions de l'utilisateur doivent se traduire en exigences de sécurité, dès le développement du système. Bien souvent, ces hypothèses implicites sont négligées, au profit des fonctionnalités évidentes.

Ce schéma décrit les tâches nécessaires pour améliorer la sécurité du logiciel, en les superposant au processus de recherche des exigences fonctionnelles : en gris les exigences fonctionnelles, en blanc l'approche sécurité.



## Identifier les exigences de sécurité de l'application

## Commentaires

Le **recensement des exigences** (*Requirements Elicitation*) est la première étape de la méthode UML : on recherche les exigences élémentaires à partir du cahier des charges, du système existant, des besoins de l'utilisateur, de la réglementation, des normes etc.

Dans la deuxième phase (*Requirements Analysis and Negotiations*), ces exigences vont être regroupées par intention d'acteur, en *Use Case*.



Pour approfondir: [https://en.wikipedia.org/wiki/Requirements\\_elicitation#Sequence\\_of\\_steps](https://en.wikipedia.org/wiki/Requirements_elicitation#Sequence_of_steps)

L'approche sécurité ajoute les tâches suivantes :

- **Evaluation du risque** (*Risk Assessment*) : estimation qualitative et quantitative des risques encourus, dans une situation bien définie et face à une « menace » connue (*Threat*).

Le risque est un « scénario qui combine un événement redouté et un ou plusieurs scénarios de menaces. On estime son niveau par sa gravité (hauteur des impacts) et sa vraisemblance (possibilité qu'il se réalise) »<sup>1</sup>.

Il faut déterminer le niveau de risque acceptable, en fonction de sa probabilité et de ses conséquences sur le système informatique et sur l'entreprise, et du coût d'une contre-mesure.



Pour approfondir : [https://en.wikipedia.org/wiki/Risk\\_assessment](https://en.wikipedia.org/wiki/Risk_assessment)

- **Analyse des menaces** (*Threat Analysis*) : le système est-il exposé sur un réseau public, sur Internet ? Quel serait l'intérêt de l'attaquer : fraude, vol de données confidentielles, image de l'entreprise etc. ?



Exemple de risque SSI et de menace dans la plaquette de présentation de la méthode EBIOS : [EBIOS-PlaquetteMetho.pdf](#)

- **Cas d'utilisation malveillante** (*Abuse / Misuse Case*) :

Avec le même formalisme que les *Use Case* d'UML, on modélise ce qui est attendu du système face à des événements imprévus, une utilisation malveillante, une attaque, en allant au-delà de l'utilisation normale du système.

- **Gestion des exceptions** :

Cela suit naturellement la définition des *Abuse/Misuse Cases* : la gestion des exceptions rend compte de la réaction du système face à l'imprévu et aux attaques.

Ces mécanismes d'exception doivent être définis dès la phase de spécification : sous forme de variantes, d'alternatives dans les scénarios UML. Le scénario dit « *nominal* » décrit ce que le système doit faire fonctionnellement, dans le bon cas. Les variantes décrivent comment il doit réagir à des situations imprévues, en gardant toujours le contrôle : reprise sur une erreur récupérable ou arrêt de l'application.

---

<sup>1</sup> Définition de l'ANSSI



- **Spécification formelle :**

Formalisation de toutes les exigences de sécurité, déduites des exigences fonctionnelles ou des attentes implicites de l'utilisateur. Cette formalisation peut faire appel à des assertions qui définissent certaines propriétés essentielles du système, au-delà de son utilisation normale.

### 3.2 PRENDRE LE POINT DE VUE DE L'ATTAQUANT

L'attaquant n'est pas particulièrement intéressé par les aspects fonctionnels du système, à moins qu'ils ne fournissent un boulevard pour son attaque. Il cherche au contraire des failles et d'autres comportements anormaux, qui rendront possible une attaque réussie. Lorsque l'on recherche les exigences de sécurité, il est important de prendre le point de vue de l'attaquant, et non pas seulement celui de l'utilisateur normal.

Il faut s'approprier la culture de l'attaquant et se familiariser avec les attaques possibles. Cette approche est facilitée par les catalogues en ligne d'attaques, classées par catégorie.



*Common Attack Pattern Enumeration and Classification* : <http://capec.mitre.org/>

Pour préciser la façon de voir de l'attaquant, on peut utiliser des *Misuse/Abuse Cases*, des arbres d'attaques et une modélisation des menaces.

### 3.3 REPERER LES BIENS ESSENTIELS

Les exigences de sécurité ont souvent une forme négative.

Les exigences de sécurité générales, comme « *Le système ne permettra aucune attaque réussie* » ne sont habituellement pas réalisables, car il n'y a pas de consensus sur la manière de les valider, à part appliquer des méthodes formelles (comme l'analyse statique de code) à tout le système.

Mais on peut identifier et protéger les services et les biens essentiels (*Assets*) qui vont motiver les attaquants.

Des **scénarios d'utilisation opérationnels** peuvent aider à comprendre quels sont ces services et biens essentiels. En fournissant des fils conducteur à travers tout le système, ils

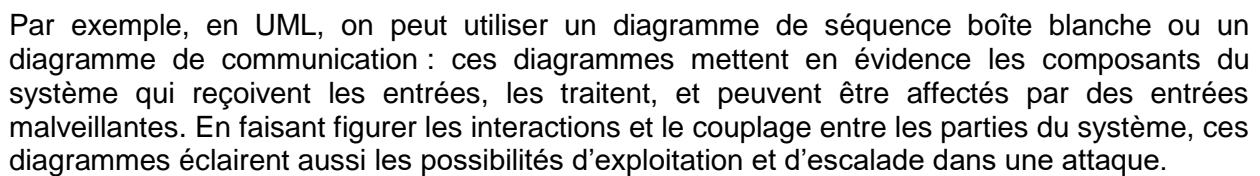
Identifier les exigences de sécurité de l'application

# UML : Diagramme de séquence

```

sequenceDiagram
    participant user1 as user 1
    participant bouton_etage as bouton étage
    participant bouton_ascenseur as bouton ascenseur
    participant controleur as contrôleur ascenseur
    participant ascenseur as ascenseur
    participant portes as portes ascenseur

    user1->>bouton_etage: 1: appui bouton étage
    activate bouton_etage
    bouton_etage->>controleur: 2: allumer
    deactivate bouton_etage
    bouton_ascenseur->>controleur: 4: eteindre
    deactivate bouton_ascenseur
    user1->>bouton_ascenseur: 6: appui bouton ascenseur
    activate bouton_ascenseur
    bouton_ascenseur->>controleur: 7: allumer
    deactivate bouton_ascenseur
    controleur->>ascenseur: 3: déplacer
    activate ascenseur
    ascenseur->>portes: 5: ouvrir
    activate portes
    portes->>ascenseur: 8: fermer
    deactivate portes
    ascenseur->>portes: 9: déplacer
    activate portes
    portes->>ascenseur: 11: ouvrir
    deactivate portes
    ascenseur->>controleur: 12: timer
    activate controleur
    controleur->>ascenseur: 13: fermer
    deactivate controleur
    deactivate ascenseur
  
```



On pourra aussi vérifier que le système gère correctement des menaces spécifiques identifiées par un modèle de menace, et répond correctement aux scénarios d'intrusion.

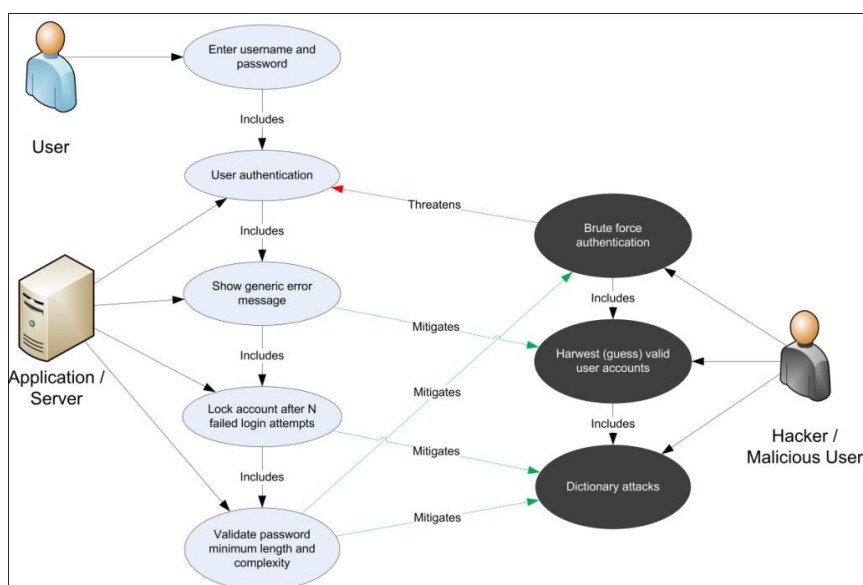
## 4. LE RECENSEMENT DES EXIGENCES (REQUIREMENTS ELICITATION)

Le recensement des exigences est le processus qui définit l'origine des exigences et la manière de les collecter.

L'utilisation d'une méthode de recensement va permettre de produire un ensemble cohérent et complet d'exigences de sécurité. Les méthodes de recensement fonctionnelles, basées sur des interviews avec les utilisateurs finaux et formalisées dans des scénarios, ne suffisent en général pas à identifier correctement les exigences de sécurité. A l'inverse, quand ces exigences sont identifiées en tant que telles de façon systématique, le système résultant sera moins exposé aux risques.

Comme les méthodes sont variées et en pleine évolution, nous allons nous limiter aux formalismes les plus courants.

### 4.1 MISUSE / ABUSE CASES



Les cas d'utilisation malveillante ou abusive (*misuse/abuse case*) sont semblables aux cas d'utilisation, sauf qu'ils concernent les utilisations malveillantes les plus courantes. Comme les *use case*, ils nécessitent une compréhension des services qui sont présents sur le système. Un *use case* décrit principalement le comportement attendu par le propriétaire du système. Un *misuse case* décrit un scénario négatif : c'est-à-dire une situation que le propriétaire du système ne voudrait jamais voir. Les *misuse cases* aident les entreprises à examiner leur logiciel avec le point de vue de l'attaquant.

Comme les *use case* sont pratiques pour spécifier, une combinaison de *misuse case* et de *use case* améliore l'efficacité du recensement des exigences. Le schéma ci-dessus (guide de test OWASP), superpose les fonctionnalités attendues du système (*use case* en bleu ciel) avec leurs acteurs licites (*User*, *Application/Server*) et les *misuse case* (en noir, avec l'acteur *Hacker/Malicious User*). Ce type de diagramme peut révéler de nombreuses failles de sécurité dans un système.

On remarque deux extensions au diagramme de *use case* : *Mitigates* et *Threatens*. Le *misuse case* **Brute Force Authentication** menace (*Threatens*) le *use case* **User Authentication**. A l'inverse, exiger des mots de passe d'une certaine longueur et complexité, limite (*Mitigates*) l'attaque de mot de passe par dictionnaire.

Identifier les exigences de sécurité de l'application

Le processus d'analyse des *misuse case* est le même que pour les *use case* : on part de la globalité du système et on détaille les *use case* principaux en sous *use case* (de type *includes* ou *extends*).

Par exemple dans le schéma, le *misuse case* **Brute force authentication** inclut le sous *misuse case* **Harvest (guess) valid user accounts**.

Dans un processus itératif et incrémental, les *misuse case* seront revus et précisés au fur et à mesure de l'avancement du projet :

- on part des *misuse case* de haut niveau (sans les scénarios) ;
- on écrit ensuite les *misuse case* de bas niveau, en précisant les scénarios d'attaque.

Dans des projets importants, les *misuse case* sont construits par une réflexion commune (*brainstorming*) entre les experts sécurité et les experts du domaine fonctionnel. Les experts sécurité questionnent les concepteurs du système, pour les aider à adopter le point de vue de l'attaquant et à identifier les points faibles de leur système.

Cette réflexion commune a trois points de départ possibles :

- **Une liste des problèmes de sécurité courants :**

On regarde si un attaquant peut exploiter ces vulnérabilités dans le système. Si c'est le cas, il faut décrire par des *misuse case* les possibilités d'exploitation et d'escalade dans l'attaque ;

- **Les ressources systèmes :**

On construit pour chaque ressource, des *misuse case* qui mettent en cause les propriétés de sécurité : authentification, confidentialité, intégrité et disponibilité ;

- **Les *use case* existants :**

C'est un moyen un peu moins structuré pour identifier les risques d'un système, mais il reste pratique pour les représenter, et vérifier que les deux premières méthodes n'ont pas laissé de côté des menaces évidentes.

Les *misuse case* obtenus par cette approche sont centrés sur l'utilisation normale (scénarios nominaux) et annotés avec les étapes malveillantes (variantes des scénarios nominaux)

OWASP (**CLASP** : *Comprehensive, Lightweight Security Process*) propose ce plan :

- Le système sera défini avec des rôles prédéfinis et l'ensemble des attaquants qui pourraient vraisemblablement agir contre lui : il faut considérer à la fois les acteurs normaux et les malveillants ;
- Comme dans les *use case* traditionnels, il faut établir qui fait quoi : quel est l'acteur principal d'un *misuse case* ?
- Les *misuse case* sont distingués graphiquement des autres *use case* (par un fond noir etc.)



Pour aller plus loin :

Article joint : [misuse.pdf](#)

## 4.2 LA MODELISATION DES MENACES

Une menace est un événement, malveillant ou non, qui pourrait endommager ou compromettre le système. La modélisation des menaces est un processus systématique qui identifie les menaces et les vulnérabilités dans le logiciel.

C'est une technique appréciée des concepteurs de système, qui leur permet de réfléchir aux problèmes de sécurité que le système pourrait rencontrer, et d'évaluer les risques pour le développement logiciel.

Elle permet au concepteur de développer des stratégies pour limiter les vulnérabilités et de concentrer ses ressources et son attention sur les parties les plus menacées du système.

### Plan type

- **Décomposer l'application** : déterminer comment l'application travaille, les biens qu'elle gère, ses fonctionnalités et la manière de s'y connecter ;
- **Définir et classer les biens** : en biens matériels et immatériels et les classer selon leur importance métier ;
- **Explorer les vulnérabilités potentielles** : qu'elles soient techniques, opérationnelles ou de gestion ;
- **Explorer les menaces potentielles** : construire une vue réaliste des vecteurs d'attaque, en prenant le point de vue de l'attaquant, par des scénarios de menace ou des arbres d'attaque ;
- **Créer des stratégies de réduction des menaces** : développer des moyens de défense pour combattre toutes les menaces réalistes.

## 4.3 GLOSSAIRE<sup>2</sup>

**Besoin de sécurité**  
(*sensitivity*)

Définition précise et non ambiguë des niveaux correspondant aux critères de sécurité (disponibilité, confidentialité, intégrité...) qu'il convient d'assurer à un élément essentiel.

**Exigence de sécurité**  
(*security requirement*)

Spécification fonctionnelle ou d'assurance sur le système d'information ou sur l'environnement de celui-ci, portant sur les mécanismes de sécurité à mettre en œuvre et couvrant un ou plusieurs objectifs de sécurité.

**Gestion du risque**  
(*risk management*)

Activités coordonnées visant à diriger et piloter un organisme vis-à-vis du risque. La gestion du risque inclut typiquement l'appréciation du risque, le traitement du risque, l'acceptation du risque et la communication relative au risque.

**Menace**  
(*threat*)

Attaque possible d'un élément menaçant sur des biens.

**Mesure de sécurité**  
(*security measure*)

Moyen destiné à améliorer la sécurité, spécifié par une exigence de sécurité et à mettre en œuvre pour la satisfaire. Il peut s'agir de mesures de prévision ou de préparation, de dissuasion, de protection, de détection, de confinement, de "lutte", de récupération, de restauration, de compensation...

---

<sup>2</sup> Repris du glossaire de la méthode GISSIP (libre de droit)

Identifier les exigences de sécurité de l'application

**Objectif de sécurité**  
(*security objective*)

Expression de l'intention de contrer des menaces ou des risques identifiés (selon le contexte) et/ou de satisfaire à des politiques de sécurité organisationnelles et à des hypothèses ; un objectif peut porter sur le système-cible, sur son environnement de développement ou sur son environnement opérationnel.

**Politique de sécurité de système d'information**  
(*information systems security policy*)

Ensemble formalisé dans un document applicable, des éléments stratégiques, directives, procédures, codes de conduite, règles organisationnelles et techniques, ayant pour objectif la protection du système d'information de l'organisme.

**Règle de sécurité**  
(*organisational security policy*)

Règle, procédure, code de conduite ou ligne directrice de sécurité qu'une organisation impose pour son fonctionnement.

**Risque**  
(*risk*)

Combinaison d'une menace et des pertes qu'elle peut engendrer, c'est-à-dire : de l'opportunité de l'exploitation d'une ou plusieurs vulnérabilités d'une ou plusieurs entités par un élément menaçant employant une méthode d'attaque ; et de l'impact sur les éléments essentiels et sur l'organisme.

## 4.4 ETUDE DE CAS

Nous en avons assez dit sur les principes et approches : il y a une littérature technique importante et c'est un domaine en pleine évolution.

Pour illustrer concrètement la démarche, nous allons suivre une étude de cas OWASP sur la modélisation des menaces :

[https://www.owasp.org/index.php/Application\\_Threat\\_Modeling#Introduction](https://www.owasp.org/index.php/Application_Threat_Modeling#Introduction)



### Guide de lecture

Cette étude de cas porte sur le site Web d'une bibliothèque de collège.

Parcourez-la, en suivant sa logique d'ensemble.

Commencez avant « *Decompose the application* ».

**Dépendances externes** : éléments externes à l'application, qui peuvent être source de menaces : typiquement des éléments qui font partie de l'organisation (accès réseau, autres serveurs etc.) sans être sous la responsabilité et le contrôle de l'équipe de développement.

Il est logique de commencer la modélisation des menaces, par la liste de ses dépendances externes (puisqu'il faudra faire avec, et sécuriser l'application en conséquence).

**Points d'entrée** : définissent les interfaces par lesquelles les attaquants potentiels peuvent interagir avec l'application et lui fournir des données.

Pour qu'un attaquant potentiel puisse réellement attaquer l'application, il faut qu'un point d'entrée existe.

A noter dans le tableau :

Identifier les exigences de sécurité de l'application

- la hiérarchisation des points d'entrée : par exemple, les différentes pages du site Web
- les niveaux de confiance (*Trust Level*) exigés pour accéder à un point d'entrée.

Ceux-ci seront listés par la suite de façon exhaustive.

### Biens (assets) :

Il ne suffit pas qu'il y ait un point d'entrée pour que l'attaque ait lieu. Il faut que quelque chose intéresse l'attaquant, un objet ou un emplacement (bien, *asset*). Les biens sont les cibles des menaces, c'est-à-dire les raisons pour lesquelles les menaces existent. Ils peuvent être physiques (par exemple la liste des clients et leurs informations personnelles) ou abstraits (réputation de l'entreprise).

### Diagrammes de flux (pour modéliser les menaces) :

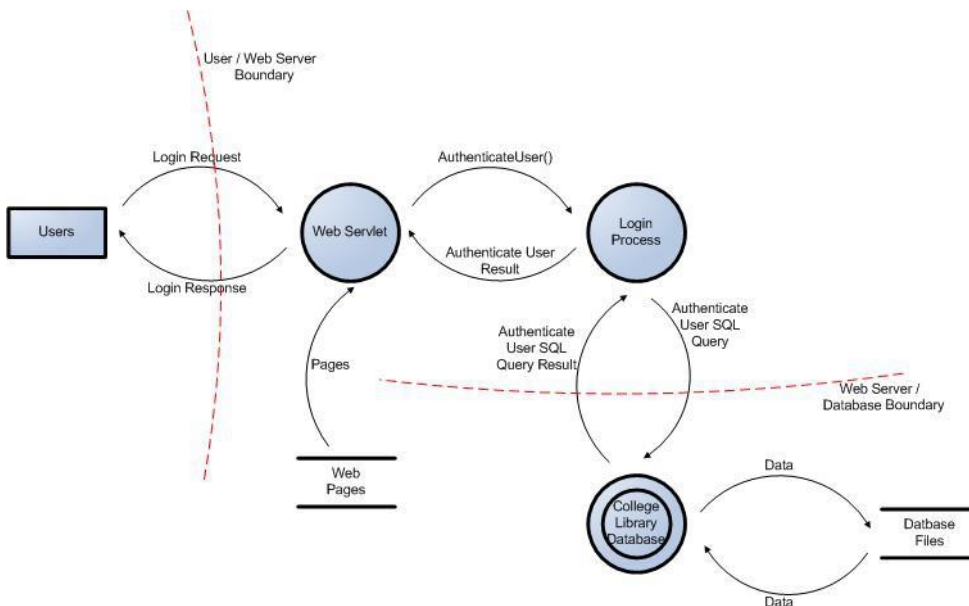
Une fois identifiés les dépendances externes, les points d'entrée et les biens, il faut les connecter dans un diagramme de flux, pour voir les chemins d'attaque possibles.

La méthode Merise comprend explicitement des diagrammes de flux.

En UML, on peut utiliser un diagramme de communication qui met en évidence les messages échangés par les objets.

Le diagramme proposé par l'OWASP distingue trois catégories : l'acteur (rectangle), le processus (cercle) et le stockage de données (deux barres horizontales).

Il représente par une ligne interrompue rouge les frontières entre les niveaux de privilège dans le système : par exemple, l'utilisateur doit se connecter pour accéder au site Web.



### Déterminer et classer les menaces

Pour déterminer les menaces, il faut s'appuyer sur une classification, qui fournit des catégories de menaces avec des exemples, afin que ces menaces puissent être identifiées systématiquement dans l'application, de façon structurée.

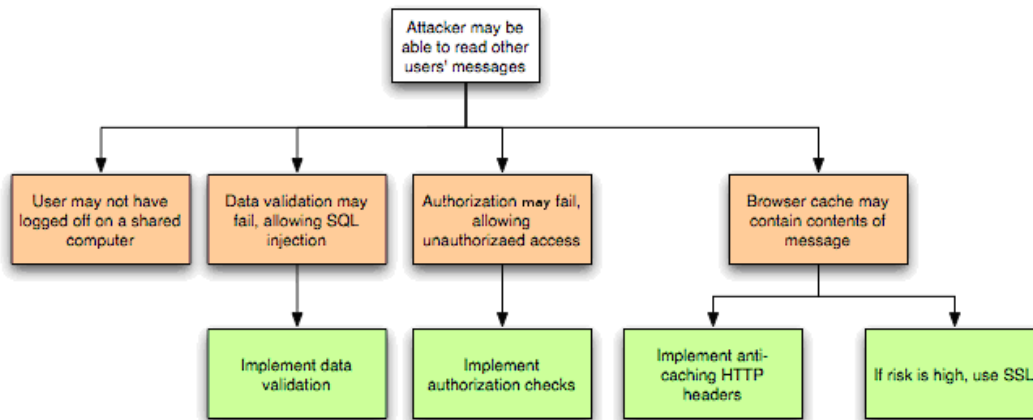
Une classification efficace s'appuie sur l'objectif de l'attaquant : usurpation d'identité (*spoofing*), falsification (*tampering*), répudiation etc.

Identifier les exigences de sécurité de l'application



Chaque catégorie de menaces enfreint un ou plusieurs principes de sécurité : authentification, intégrité, non répudiation, disponibilité, autorisation (listés dans la colonne de droite).

### Analyse des menaces (par arbre)



Chaque menace (en blanc) est décomposée en scénarios de menace (en orange), auxquels on associe les mesures correctives (en vert).

### Misuse case

Une fois que l'on a répertorié les menaces, vulnérabilités et attaques courantes, il faut réaliser une analyse de menace plus poussée, par des *misuse case*, qui font partie intégrante du recensement des exigences de sécurité.

On reprend les scénarios des *use case* fonctionnels, pour vérifier s'ils présentent des vulnérabilités, ou si les mesures de sécurité déjà en place peuvent être outrepassées.

### Classement des menaces

Parmi les méthodes de classement, retenons le « **Modèle de risque générique** », où le risque est calculé d'après sa probabilité et ses conséquences (impact) :

**Risque = probabilité \* impact**

En fonction du risque et des possibilités de mesures correctives, on décide finalement du niveau de correction souhaitable :

- 1 **Ne rien faire** : par exemple, espérer le meilleur ;
- 2 **Informé sur le risque** : par exemple, informer les utilisateurs du risque ;
- 3 **Limiter le risque** : par exemple, en mettant en place des mesures correctives ;
- 4 **Accepter le risque** : par exemple, en évaluant l'impact sur l'exploitation ;
- 5 **Transférer le risque** : par exemple, par des accords contractuels et des assurances ;
- 6 **Mettre fin au risque** : par exemple, en arrêtant ou en déconnectant le bien menacé.

Identifier les exigences de sécurité de l'application



## 5. ANALYSE DE RISQUE D'ENTREPRISE

(Ce chapitre est optionnel).

Nous venons de présenter les grandes lignes de l'analyse de risque d'application, qui permet de lister les exigences de sécurité et de modéliser les menaces, à l'échelle d'un projet de développement. Nous avons illustré la modélisation des menaces par une étude de cas de l'OWASP.

Avant de mener une approche sécurité sur une application donnée, il faut analyser les risques au niveau de l'entreprise et de son système d'information. De la même manière qu'en Merise, l'étude préalable d'une application est précédée par le schéma directeur de l'entreprise.

En tant que concepteur-développeur, nous n'aurons pas à réaliser d'analyse de risque au niveau de l'entreprise. Mais il est utile d'avoir la compétence de lecteur, pour s'approprier une analyse réalisée par des experts, et en extraire ce qui concerne notre application.

Nous allons donc suivre la méthode de l'ANSSI, **EBIOS** (Expression des **B**esoins et Identification des **O**bjectifs de **S**écurité), avec un point de vue de lecteur.

Documents fournis dans le dossier : [EBIOS](#)

### 5.1 PRESENTATION SUCCINCTE DE LA METHODE EBIOS



Utilisez la plaquette de présentation : [EBIOS-PlaquetteMetho.pdf](#)

### 5.2 ETUDE DE CAS

Pour fixer les idées, parcourez l'étude de cas (pp. 5 à 53, temps estimé : 1 heure)



[EBIOS-EtudeDeCas-Archimed.pdf](#)

Cela risque au départ de vous paraître rébarbatif.

Mais il faut se mettre dans la situation professionnelle du concepteur-développeur qui reçoit cette étude de cas, et cherche les informations concernant son activité.

Deux conseils :

- recherchez la logique d'ensemble, sans rentrer dans les détails des tableaux : on lira en diagonale tout ce qui concerne la sécurité physique (incendie, protection des locaux etc.) et la disponibilité du réseau et des équipements ;
- repérez dans les événements redoutés, les scénarios de menaces et les risques, ce qui définit des exigences de sécurité pour les applications informatiques.

Identifier les exigences de sécurité de l'application

Afpa © 2016 – Section Tertiaire Informatique – Filière « Etude et développement »

## **CRÉDITS**

### **OEUVRE COLLECTIVE DE L'AFPA**

Sous le pilotage de la DIIP  
et du centre sectoriel Tertiaire

### **EQUIPE DE CONCEPTION**

Chantal PERRACHON – IF Neuilly-sur-Marne  
Régis Lécu – Formateur AFPA Pont de Claix

Identifier les exigences de sécurité de l'application

Afpa © 2016 – Section Tertiaire Informatique – Filière « Etude et développement »