Secteur Tertiaire Informatique

Filière « Etude et développement »

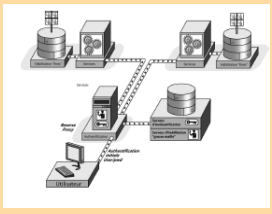
Séquence « Maquetter une application »

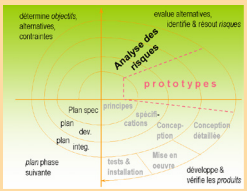
Sécuriser l'interface utilisateur

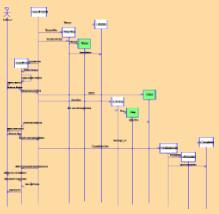
Apprentissage

Mise en situation

Evaluation







Version	Date	Auteur(s)	Action(s)
1.0	07/07/16	Lécu Régis	Création du document

TABLE DES MATIERES

Objectifs

A l'issue de la séance, le stagiaire sera capable, à partir du type d'application, de son environnement technique et du niveau de sécurité souhaité :

- De mettre en œuvre les bonnes pratiques de sécurisation d'interface utilisateur pour concevoir l'interface utilisateur la plus adaptée à l'application
- D'expliciter les contraintes de sécurité et leur traduction en terme graphique

Pré requis

Cette séance applique les principes du développement sécurisé à la conception d'interface utilisateur : elle a pour prérequis le chapitre « *Les principes du développement sécurisé* » dans la séance « *Coder de façon défensive en appliquant les bonnes pratiques de sécurité* ».

Méthodologie

Ce document peut être utilisé en présentiel ou à distance.

Il précise la situation professionnelle visée par la séance, la resitue 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

- Exposé sur la sécurisation des interfaces utilisateurs : *UI_vs_Security.pdf*

- Cours « *Usable Privacy And Security* » de l'université *Carnegie Mellon* qui fait une synthèse des contraintes de sécurité et de prise en compte des besoins de l'utilisateur, dans la conception d'interface utilisateur : *UsablePrivacyAndSecurity.pdf*

- Conception d'une interface utilisateur sécurisée : *MaquetteBOUM.doc*

- Guide Apple pour la conception d'interface utilisateur :
  iOS Human Interface Guidelines.pdf

1. INTRODUCTION

Cette séance sur la sécurisation des interfaces utilisateur suit l'objectif du projet **CyberEdu** : appliquer la démarche sécurité à toutes les étapes d'un projet informatique.

Dans la séance « *Coder de façon défensive en suivant les bonnes pratiques de sécurité* », nous avons listé les principes généraux du développement sécurisé, puis nous les avons appliqués dans le cas particulier du codage en Java, en suivant les règles de sécurité préconisées par le CERT.

Ces mêmes principes vont maintenant nous guider dans la conception de notre interface utilisateur, car la conception de la maquette qui précède le développement de l'interface n'est pas neutre, du point de vue de la sécurité.

Les principes de sécurisation vus dans cette séance seront appliqués dans la séance suivante *« Construire la maquette de l'application ».*

2. L'APPROCHE SECURITE DANS L'ERGONOMIE

2.1 ADAPTER SON INTERFACE UTILISATEUR AUX EXIGENCES DE SECURITE IDENTIFIEES

Une ergonomie sécurisée doit traduire les exigences de sécurité identifiées dans la phase d'analyse. Ces exigences ne seront pas toujours explicites et il faudra souvent refaire des entretiens client avec une approche sécurité, en se posant des questions du type :

- quel est le niveau de sécurité requis pour l'application, en fonction de son environnement technique et de son domaine professionnel : bancaire, installation industrielle critique, application pour l'utilisateur final, jeu etc. ?
- l'application est-elle multiutilisateur avec des autorisations et des privilèges sélectifs selon les utilisateurs ?
- l'application est-elle accessible physiquement depuis un environnement protégé ou dans un lieu public ?

2.2 APPLIQUER LES PRINCIPES GENERAUX DE LA SECURISATION D'INTERFACE UTILISATEUR

De même que les règles de sécurité du CERT rejoignent souvent les bonnes pratiques objet, l'application aux interfaces utilisateur des principes du développement sécurisé va rejoindre les préoccupations de l'ergonomie.

2.2.1 Keep UI Small and Simple

Rappelons qu'il n'y a jamais de « sécurité par l'obscurité ». Un logiciel difficile à utiliser décourage l'utilisateur de bonne foi, pas l'attaquant.

Une interface utilisateur sera d'autant plus facile à sécuriser qu'elle sera simple, fonctionnelle, facile d'accès pour l'utilisateur. Il faut donc :

- éviter la complexité inutile et le design agressif ; l'ergonomie n'est pas synonyme de tape à l'œil ;
- quel problème précis essaie-t-on de résoudre dans l'application ?

  Une application cohérente répondant complètement à un problème bien défini, sera plus facile à sécuriser qu'une grosse application qui veut tout faire et le fait mal. Il sera évidemment plus facile de rechercher les vulnérabilités, les menaces et les attaques possibles, en connaissant précisément l'objectif de l'application, son périmètre, ses

utilisateurs. Il ne faut donc pas hésiter à découper une application complexe en plusieurs petites applications ciblées et fonctionnelles.

- la cohérence exigée pour l'application l'est aussi pour ses différents éléments graphiques : chaque élément doit fournir une fonctionnalité bien délimitée et n'être accessible qu'à l'utilisateur ou au groupe d'utilisateurs autorisé.

2.2.2 Adopter une posture de méfiance

Nous avons vu dans la séance précédente qu'une couche logicielle, même interne, ne doit jamais accorder une confiance totale aux autres couches, et qu'elle doit donc valider systématiquement ses entrées.

Ce principe s'applique en premier lieu à l'interface utilisateur qui est une couche externe, directement exposée aux attaques. Pour reprendre la comparaison classique avec l'architecture militaire, l'interface utilisateur est le premier niveau de défense du château (fossés) qui devrait suffire à empêcher les attaques basiques.

Il faut se méfier à la fois de ce qui vient de l'extérieur (sécurisation des entrées) et de ce qu'un attaquant pourrait faire avec les informations qu'on lui renvoie (contrôle des sorties) :

1) Sécuriser ses entrées :

- pour minimiser les attaques, il vaut mieux prévenir les erreurs qu'y répondre après coup. Il faut éviter autant que possible de solliciter le code de contrôle de l'interface utilisateur (deuxième niveau de défense, le rempart extérieur) lorsque de bons choix d'ergonomie suffisent à empêcher l'erreur : éviter par exemple de saisir une date dans un champ texte, en confiant au code la vérification de la chaine saisie ;
- on choisira donc des composants graphiques spécialisés pour les entrées dont le type et l'intervalle sont connus à l'avance : valeurs entières ou réelles, dates, énumérations etc. On privilégiera les champs de saisie typés avec masque, les calendriers, les listes, les cases à cocher, les boutons radios ou tout composant graphique avec une question fermée, à un champ de saisie simple ;
- pour les données où aucun composant graphique spécialisé n'est disponible, on vérifiera la cohérence de la saisie dès que l'utilisateur quitte le champ. Cela facilite la vie de l'utilisateur qui n'accumule pas les erreurs de saisie, et évite aussi les effets de bord qui peuvent créer des vulnérabilités : par exemple, un champ « mesure » a un intervalle de validité différent selon le choix effectué dans le champ « unité » ; comment réagira l'application si le champ « unité » n'est pas saisi ou est s'il est incorrect ?
- on fera une nouvelle validation de toutes les entrées, par code, avant d'effectuer le traitement métier demandé (par exemple, à la validation du formulaire dans le cas du web). On vérifiera que tous les champs obligatoires ont bien été saisis, et que les champs saisis ont des valeurs correctes.

2) Contrôler ses sorties

On n'est pas tenté d'attaquer ce que l'on ne voit pas : dans le château-fort, l'assaillant voit le rempart extérieur, mais pas le trésor dans le donjon ; les positions des sentinelles et les heures de relève ne sont pas affichées à l'entrée !

Pour bien sécuriser une interface utilisateur, il faut penser aux conséquences des informations communiquées sur la sécurité de notre application, mais aussi sur celle des autres applications et de l'entreprise elle-même :

- **exposition minimale des données** : ne communiquer que les données réellement exigées par les fonctionnalités de l'application (elles-mêmes minimales et exigées par le cahier des charges). Toute fonctionnalité et tout affichage superflu ne sont profitables qu'à l'attaquant.

  Exemple : un annuaire d'entreprise doit filtrer les informations issues de la DRH (nom, prénom, nom du service, bureau, étage, téléphone) et éviter de communiquer l'âge du salarié ou son salaire.

  Mais cela peut aller plus loin : dans un environnement fortement sécurisé, l'application n'affichera que les numéros des services et pas leurs noms complets ; car ceux-ci renseignent déjà sur l'activité du service, et peuvent mettre en danger les salariés qui y travaillent.

  A partir d'un outil public (l'annuaire, l'affichage sur le rempart), on accède à une donnée confidentielle (la spécialité du salarié, l'emplacement du trésor).

- **messages d'information et d'erreur minimaux et neutres** :

  « Le diable est dans les détails » : un développeur peut vouloir guider au mieux son utilisateur, auquel il donne sa confiance, sans prendre en compte les cas d'attaques. Il faut guider l'utilisateur, mais en filtrant toutes les informations en sortie, de façon à ce qu'elle ne facilite pas les attaques (la position et l'heure de relève des sentinelles !).

  Exemples :

  - des plantages qui affichent la ligne en défaut, le nom du serveur, l'erreur SQL etc. (fréquents dans les sites Web développés en PHP) ;

  - des exceptions bien gérées mais qui affichent intégralement l'erreur système, ce qui revient au cas précédent. Dans les deux cas, l'utilisateur malveillant tirera profit de ces messages pour analyser votre architecture système et préparer ses attaques.

  Bonne pratique : utiliser des messages fonctionnels comme « utilisateur inexistant » et non pas :

```
com.microsoft.sqlserver.jdbc.SQLServerException: Login failed for user 'TITI'.
ClientConnectionId:e6335e64-ca68-4d72-8939-5b7ded951424 at
com.microsoft.sqlserver.jdbc.SQLServerException.makeFromDatabaseError etc.
```

  - des messages d'information fonctionnel mais trop précis :

  Suite à une erreur de connexion, l'application affiche sélectivement « Utilisateur inconnu » ou « Mot de passe incorrect ». Cela facilite la tâche du hacker, qui peut lancer une attaque par « force brute » d'abord uniquement sur le nom de l'utilisateur, puis lorsqu'il est trouvé, sur le mot de passe. (fr.wikipedia.org/wiki/Attaque_par_force_brute).

2.2.3 Séparer et minimiser les permissions et les privilèges

Puisque l'interface utilisateur est la couche logicielle au contact du monde extérieur, elle doit prendre en compte l'authentification des utilisateurs et le partitionnement de leurs tâches.

Chaque utilisateur ne verra que les données autorisées et n'aura accès qu'aux fonctionnalités qui le concernent.

Encore une fois, on n'est pas tenté par ce que l'on ne voit pas : il ne faut donc pas mettre en grisé ou invalider les fonctionnalités autorisées à d'autres utilisateurs. C'est presque demander à être attaqué ! Les zones visibles et invalidées doivent désigner des fonctionnalités permises à l'utilisateur connecté dans un autre contexte : par exemple, le champ « mesure » est invalide

quand le champ « unité » n'a pas encore été saisi, puisque l'unité doit être connue pour valider la cohérence de la mesure saisie.

2.2.4 Que valent les défenses de l'interface utilisateur ?

- une interface utilisateur est par définition côté client (pour un client lourd) ou accessible du client (pour un client web).
- elle peut donc être cassée : le *hacker* peut par exemple décompiler le code de l'interface utilisateur (ou simplement lire le code JavaScript dans le cas d'un client web) et rendre accessibles les champs et les menus cachés, pour les utilisateurs non identifiés.
- les défenses côté client sont donc en général assez faibles. Certains les considèrent comme un simple confort pour l'utilisateur de bonne foi : une interface bien conçue va le guider et lui éviter d'outrepasser ses droits, en créant des problèmes par inadvertance ; mais elle n'oppose qu'une faible résistance au vrai *hacker.*
- bien que cette opinion soit fondée, une interface utilisateur sécurisée reste un premier niveau de défense utile, comme le rempart dans le château-fort. Elle n'a jamais eu pour but d'arrêter toutes les attaques, mais de décourager les attaquants peu qualifiés et de retarder les autres.
- la sécurisation de l'interface utilisateur n'a de sens qu'associée à un système de « défense en profondeur » où chaque couche va adopter une posture de méfiance et tester à nouveau toutes ses entrées : le deuxième niveau de défense côté serveur (le donjon) sera bien sûr plus robuste.

2.3 CONCILIER L'APPROCHE SECURITE AVEC LES ATTENTES DE L'UTILISATEUR

En se centrant uniquement sur la sécurité, les deux points précédents considèrent tout utilisateur, normal ou *hacker*, comme un problème et jamais comme une solution.

Mais les études récentes dans le domaine des interfaces utilisateur concluent qu'une approche sécurité ne peut réussir que si elle respecte l'utilisateur normal, en prenant en compte ses demandes, son vocabulaire, son mode de travail et de pensée, afin de l'amener à participer activement à la sécurité de l'entreprise.

 Lire l'exposé UI_vs_Security.pdf en suivant ce Guide de lecture [1]:

p. 1-4 : il ne sert à rien de considérer l'utilisateur comme la source de tous les maux.

Le problème n'est pas l'utilisateur, mais le contexte dans lequel il va travailler. Nous ne pourrons jamais contrôler l'utilisateur, mais nous pouvons contrôler le contexte dans lequel il travaille, via l'interface utilisateur.

p. 7-8 : l'exemple des mots de passe montre qu'il peut y avoir contradiction entre les exigences de la sécurité informatique, et les capacités humaines.

Un utilisateur aura tendance à choisir des mots de passe simples pour pouvoir les mémoriser. Des exigences draconiennes sur la longueur et la complexité des mots de passe peuvent être contre-productives : l'utilisateur finira par écrire son mot de passe sur un Post-It collé sur son écran ! En réalité, ce n'est pas le mot de passe qui est faible, mais le système d'identification par mot de passe qui montre son insuffisance.

Pour une sécurité accrue, il faut passer par d'autres systèmes d'identification, carte, reconnaissance biométrique : http://www.biometrie-online.net.

[1] Traduit de l'exposé de Tom Vogt, original joint, libre de droit.

p. 9-11 : l'interface utilisateur ne doit pas se décharger de sa responsabilité sur l'utilisateur, en multipliant les boîtes de dialogue de confirmation, avec des formulations souvent confuses, qui interrompent le déroulement normal du travail de l'utilisateur.

Les choix présentés à l'utilisateur devraient parler d'eux-mêmes et il faudrait réserver les boîtes de dialogue et les messages pour des cas d'erreurs ou des choix risqués.

p. 11-13 : une interface utilisateur mal conçue et peu lisible a souvent une part de responsabilité dans les attaques par « ingénierie sociale » comme le hameçonnage (*fishing*).

Dans l'exemple, le formulaire web contient de nombreuses incohérences, mais elles sont masquées par la présentation.

p. 14-16 : toutes les interfaces utilisateur graphiques reposent sur des « métaphores » : un répertoire est représenté par un dossier, une clé de cryptographie par une clé réelle etc. Les métaphores sont pratiques et facilitent la compréhension et l'utilisation du logiciel. Mais elles sont souvent utilisées sans précaution, alors qu'elles peuvent être porteuses de significations fausses ou involontaires.

Le courriel est un exemple particulièrement mauvais de métaphore : un « message » est un « document » contenant des « données » ; message et données paraissent sans danger. Mais en réalité, un courriel peut aussi transporter du code dangereux, en pièce jointe.

p. 22-23 Conclusion

- Le problème n'est pas chez les utilisateurs, mais dans l'interface utilisateur.
- La conception d'une bonne interface utilisateur doit affecter une responsabilité à celui auquel elle revient, ne se fait pas remarquer (pas « tape à l'œil »), parle le langage de son destinataire et n'espère pas obtenir un comportement non humain de la part des humains.
- La prise en compte des facteurs humains va améliorer la sécurité en entraînant une meilleure acceptation du logiciel et en réduisant les erreurs.
- Respecter l'utilisateur et ses besoins permettra d'obtenir une bonne coopération.

p. 24 Les principes :

- **Profilage des utilisateurs** : connais ton utilisateur, parle son langage
- **La métaphore** : emprunte des comportements à des contextes familiers pour les utilisateurs
- **Visibilité** : laisse l'utilisateur voir clairement ses options
- **Cohérence** : le comportement (de l'application) devrait être cohérent (constant)
- **Contexte et gestion des processus métier** (*workflow*) : adapte-toi au mode (à l'état) dans lequel l'utilisateur est actuellement.
- **Test par les utilisateurs** : demande de l'aide pour repérer les problèmes inévitables.

Le tableau qui suit résume clairement les objectifs propres de la sécurité informatique, de la conception d'interface utilisateurs (HCI = *Human-Computer interaction*) et de leur synthèse indispensable (*Usable Security*) :

- Il faut prendre en compte à la fois, comme des contraintes fondamentales, les facteurs humains et la sécurité ;

- Il faut s'intéresser autant aux utilisateurs normaux qu'aux adversaires et aux attaquants potentiels ;
- Il faut inclure à la fois des modèles de menaces, des modèles métier et des modèles cognitifs ;
- Pour rendre compte de la performance de l'interface utilisateur, il faut mesurer à la fois la facilité d'utilisation (*usability*) et la sécurité de l'application ;
- Il faut faire un double retour d'expérience utilisateur : sur l'utilisation courante, mais aussi sur les adversaires actifs et les cas d'attaque.

Usable security research bridges security and usability

Security	Usability/HCI	Usable Security
Humans are a secondary constraint to security constraints	Humans are the primary constraint, security rarely considered	Human factors and security are both primary constraints
Humans considered primarily in their role as adversaries/attackers	Concerned about human error but not human attackers	Concerned about both normal users and adversaries
Involves threat models	Involves task models, mental models, cognitive models	Involves threat models AND task models, mental models, etc.
Focus on security metrics	Focus on usability metrics	Considers usability and security metrics together
User studies rarely done	User studies common	User studies common, often involve deception + active adversary

2.4 COMMENT PROCEDER DANS UN PROJET ?

Ces trois contraintes sont indissociables : pendant toute la conception de l'interface utilisateur, il faut concilier les principes généraux de sécurisation, les exigences de sécurité propres à l'application et la prise en compte des demandes utilisateur en tant que telles.

Les exigences de sécurité peuvent être listées avant la construction de la maquette mais elles devront être affinées de façon itérative en cours de construction, en particulier dans un projet piloté en méthode agile, où la conception graphique de la maquette se fait en étroite collaboration avec le client.

L'approche sécurité participe à la qualité de la conception et à l'activité de conseil vis-à-vis du client, en lui faisant prendre conscience des conséquences de ses choix de présentation sur la sécurité de son application : vulnérabilités issues de choix marketing, ludiques, tape à l'œil etc.

Il est clair que l'interface utilisateur est un argument de vente pour certains types d'application, dont la présentation ne pourra être conduite seulement par des critères de sécurité. Comme on vient de le voir, négliger les attentes et les besoins de l'utilisateur va même à l'encontre d'une sécurité efficace.

Mais à l'inverse, une approche sécurité doit au minimum évaluer les risques encourus dans chaque choix, et le faire savoir au client.

2.5 QUID DU DEVELOPPEUR DANS LA CONCEPTION D'INTERFACE UTILISATEUR SECURISEE ?

L'ergonomie et la sécurisation des interfaces utilisateurs est un métier à part entière.

Comme dans les séances qui suivront sur le réseau et la cryptographie, l'objectif fixé par le projet **CyberEdu** n'est pas de former des spécialistes, mais de sensibiliser tous les développeurs informatiques à des activités proches et transverses, qui ont un impact direct sur la sécurité globale de leur application. Mais ceci n'interdit pas à des développeurs motivés par ce domaine de se perfectionner.

Nous suivrons donc la même démarche que dans la séance « *Coder de façon défensive en suivant les bonnes pratiques de sécurité » :*

- Acquisition des réflexes de base du développement d'interface sécurisée, à partir d'une étude de cas ;
- Présentation de sites et de guides d'éditeur (Apple, Microsoft) sur la sécurisation des interfaces utilisateur, pour aller plus loin.

3. ETUDE DE CAS

Pour illustrer les principes de la sécurisation d'interface utilisateur, nous allons partir du cahier des charges d'une application de productique et de gestion de stock : un peu schématique[2], mais suffisant pour se poser les questions importantes sur la sécurité et justifier ses choix.

Vous mettrez vous-mêmes en pratique cette démarche dans la séance suivante *« Construire la maquette de l'application ».*

3.1 CAHIER DES CHARGES

La société BOUM qui fabrique des moteurs, voudrait améliorer la qualité de sa production en :

- informatisant le processus de contrôle des pièces fabriquées ;

- effectuant des statistiques sur l'évolution de la qualité des pièces produites.

La première itération du projet concerne uniquement la production des pistons, le processus de mesures des pistons et les statistiques qui les concernent.

Les pistons sont fabriqués en aluminium moulé, par des presses à injection sous pression.

Sur chaque piston on mesurera quatre cotes : en haut, dans le sens longitudinal du moteur (cote HL) ; en haut, dans le sens transverse (cote HT) ; en bas, dans le sens longitudinal (cote BL) ; en bas, dans le sens transverse (cote BT).

Comme pour toute fabrication, une dispersion des diamètres obtenus est inévitable.

Afin d'assurer une meilleure qualité de la production, les pistons seront triés en trois catégories: petit, moyen, gros.

Chaque catégorie est définie par un intervalle de tolérance : [- 0.1, -0.01] pour petit, [-0.05, +0.05] pour moyen, et [+0.01, +0.1] pour gros.

Pour qu'un piston appartienne à l'une de ces catégories, toutes ses cotes devront être dans l'intervalle de confiance. Dans tous les autres cas, le piston sera rebuté.

L'entreprise fabrique divers modèles de piston (R4, R8, R10...) définis par leur nom et leur diamètre nominal.

---

[2] Résumé du cahier des charges d'un projet fil rouge de la formation CDI, Pont de Claix

Quotidiennement, le responsable d'atelier consulte les stocks. S'il constate qu'un modèle de pièces, pour une catégorie donnée, est en quantité insuffisante (Stock < Seuil minimum), il peut déclencher le lancement d'un lot.

La production est lancée par lot. Chaque lot est pris en charge par une et une seule presse à injecter. Un lot ne concerne qu'un seul type de piston. A la sortie des presses, les pistons sont déposés sur un tapis roulant et acheminés vers les postes de contrôle. Sur le tapis roulant à destination des contrôleurs, les lots ne sont jamais mélangés. Chaque changement de lot est matérialisé sur le tapis par une étiquette. Un lot ne peut être traité que par un seul poste de contrôle.

Lorsque la fabrication d'un lot est terminée, la presse redevient disponible pour fabriquer un autre lot, et ceci même si toutes les pièces du lot ne sont pas encore mesurées.

Les employés en charge du contrôle examinent les pistons un par un et en fonction des mesures établies rangent chaque piston dans une des 4 caisses : Gros, Moyen, Petit, Rebut. Un contrôleur peut toujours rebuter une pièce sur un défaut visuel (cassure, état de surface) même si les dimensions sont correctes.

Lorsqu'une caisse est pleine, elle est conduite au magasin où elle est stockée et enregistrée. Une caisse peut contenir des pièces provenant de lots différents. Une caisse ne contient qu'un seul type de piston et pour une seule catégorie.

Dans l'atelier d'assemblage, les ouvriers disposent de 3 caisses de pistons : Gros, Moyen ou Petit. Lorsqu'une caisse est vide, ils vont en chercher une autre au magasin.

L'un des objectifs de cette itération sera d'obtenir les statistiques (réduites et détaillées), destinées au responsable qualité, pour chacun des lots :

- Statistiques réduites :
    - Pour chaque cote : moyenne, maximum, minimum, écart type.
    - Nombre de pistons obtenus dans chaque catégorie.
- Statistiques détaillées : courbes d'évolution des cotes, un point par pièce, une courbe par cote.

Enfin, le responsable d'application est en charge de la gestion des machines, des modèles de piston et des seuils.

3.2 LISTE DES FONCTIONNALITES [3]

Quoi	Qui	Quand	Comment
Consultation des stocks par modèle et catégories	Resp. Atelier	A la demande	Informatisée
Lancement d'un lot	Resp. Atelier		Informatisée
Démarrage d'un lot	Resp. Production	Presse libre	Informatisée
Production des pièces	Resp. Production		Manuel
Fin du lot : mise en place étiquette	Resp. Production		Manuel

[3] Extrait de l'analyse Merise du même projet fil rouge

Fin du lot : enregistrement libération de la presse	Resp. Production		Informatisée
Contrôle des visuels des pièces	Contrôleur		Manuel
Mesure des cotes	Contrôleur		Manuel
Saisie des mesures	Contrôleur		Informatisée
Mise en Caisse	Contrôleur		Manuel
Arrêt du lot : calcul des moyennes, max, min et écarts-type pour le lot.	Contrôleur	Etiquette fin de lot	Informatisée
Entrée des caisses produites en magasin	Magasin	Caisse pleine	Manuel
Enregistrement des entrées de stock	Magasin		Informatisée
Sortie des caisses pour assemblage	Magasin	Besoin atelier	Manuel
Enregistrement des sorties de stock	Magasin		Informatisée
Consultation des Stat. Réduites par lot	Resp. Qualité		Informatisée
Consultation des Stat. Détaillées par lot	Resp. Qualité		Informatisée
Gestion des modèles	Responsable de l'application		Informatisée
Gestion des machines	Responsable de l'application		Informatisée
Gestion des seuils	Responsable de l'application		Informatisée

3.3 CONCEPTION DE L'INTERFACE UTILISATEUR SECURISEE DE L'APPLICATION BOUM

Nous allons relire le cahier des charges en gardant en mémoire ces trois axes :

- Les exigences de sécurité propres à notre cahier des charges
- Les principes du développement sécurisé
- Les attentes et besoins de l'utilisateur, son vocabulaire etc.

Comme pour les *patterns* objet, il n'y a pas d'ordre dans l'application de ces points : il faudra respecter à la fois la demande du client et les principes de sécurisation.

3.3.1 Identifier les exigences de sécurité propres à l'application

1) Un bon point de départ est de prendre en compte les spécificités de l'application, pour déterminer le niveau de sécurité requis :

- l'application n'est pas dans un lieu public, ni publiée sur internet. Elle n'est accessible physiquement que dans un environnement protégé (usine), en principe isolé d'internet. Donc la probabilité d'une attaque par un *hacker* opérant de l'extérieur est faible.

  Une attaque par un salarié opérant de l'intérieur est toujours possible, mais il faut examiner ses motivations : notre entreprise n'est pas une banque ni un site industriel critique (barrage, centrale etc.) et le seul mobile d'une attaque serait une vengeance contre l'entreprise. Ce risque doit être pris en compte mais il peut être considéré comme faible ;

- mais cette attaque, dans un environnement de production, pourrait avoir des conséquences graves : mauvais ordonnancement de la production conduisant à une rupture de stock et à l'arrêt de la production des moteurs ; statistiques faussées conduisant à un mauvais réglage des presses et à une dégradation de la production.

Le niveau de sécurité exigé par l'application peut donc être estimé à : **Moyen**.

2) Le cahier des charges montre qu'il s'agit d'une application multiutilisateur, où chaque utilisateur a des droits distincts décrits dans la liste des fonctionnalités.

Comme une attaque interne par un salarié de l'entreprise qui voudrait outrepasser ses droits n'est pas à exclure, il faudra mettre en œuvre un mécanisme d'authentification fiable et des permissions sélectives selon les utilisateurs connectés.

Reprenons la liste des fonctionnalités, en ne conservant que les tâches informatisées. Cette liste nous fournit les catégories d'utilisateur (Responsable d'atelier, Responsable de production, Contrôleur, Magasin, Responsable Qualité, Responsable d'application) et les fonctionnalités auxquelles ils ont accès. Complétons la liste avec le lieu où doit être déployée l'application :

Quoi	Qui	Quand	Où
Consultation des stocks par modèle et catégories	Resp. Atelier	A la demande	Administration
Lancement d'un lot	Resp. Atelier		Administration
Démarrage d'un lot	Resp. Production	Presse libre	Administration
Fin du lot : enregistrement libération de la presse	Resp. Production		Administration
Saisie des mesures	Contrôleur		Atelier
Arrêt du lot : calcul des moyennes, max, min et écarts-type pour le lot.	Contrôleur	Etiquette 'fin de lot'	Atelier
Enregistrement des entrées de stock	Magasin		Magasin
Enregistrement des sorties de stock	Magasin		Magasin

Consultation des Stat. Réduites par lot	Resp. Qualité		Administration
Consultation des Stat. Détaillées par lot	Resp. Qualité		Administration
Gestion des modèles	Responsable de l'application		Administration
Gestion des machines	Responsable de l'application		Administration
Gestion des seuils	Responsable de l'application		Administration

3) Comme les différentes fonctionnalités et les différents utilisateurs partagent les mêmes données, l'application devra être client-serveur. Le nombre de tiers n'est pas à déterminer dans cette analyse de sécurité centrée sur l'interface utilisateur.

3.3.2 Appliquer les principes du développement sécurisé (sans oublier son utilisateur)

1) *Keep UI Small and Simple*

2) Adopter une posture de méfiance

3) Séparer et minimiser les permissions et les privilèges

4) === Sans négliger l'utilisateur normal, qui n'est pas notre ennemi ! ====

Quel périmètre doit-on définir pour notre (ou nos) application ?

Les deux critères à prendre en compte sont : les lieux d'utilisation de l'application ; les processus métier et leurs acteurs.

- 3 lieux : administration, atelier et magasin
- 6 processus métier associés chacun à un seul acteur : application, qualité, magasin, ordonnancement/lancement, production, contrôle.

Si l'on applique strictement les principes (1) et (3), on aboutit à 6 petites applications dédiées chacune à un seul processus métier :

- du point de vue de la sécurité, c'est la solution la plus robuste ;
- mais cette granularité ne tient pas compte des lieux d'utilisation et n'est pas pratique pour l'utilisateur, qui doit se rendre sur un poste dédié à son type d'activité.

Comme le niveau de sécurité attendu de l'application est **Moyen**, on choisira donc un compromis :

- une application bureautique, multiutilisateur, qui implémente tous les processus métier, à l'exclusion du contrôle des pièces ;
- une application d'atelier, spécialisée dans le contrôle des pièces et l'arrêt des mesures d'un lot ;

- cette division respecte les attentes des différents utilisateurs et les contraintes des lieux d'utilisation : la première application aura une ergonomie bureautique ; la deuxième un design d'atelier, accessible par un non informaticien et permettant un travail à la chaîne.

3.4 MAQUETTE COMMENTEE DE L'INTERFACE UTILISATEUR DE BOUM

📖 Lire le document **MaquetteBoum.pdf** qui propose une maquette de l'interface, répondant le plus simplement possible au raisonnement précédent.

Les choix motivés par la sécurisation de l'interface sont indiqués en rouge.

4. POUR ALLER PLUS LOIN

- Le chapitre *d'Apple* sur la sécurisation des interfaces utilisateur (très clair) :

  https://developer.apple.com/library/ios/documentation/Security/Conceptual/SecureCodingGuide/Articles/AppInterfaces.html

- Le guide d'Apple sur le développement d'interface utilisateur sous iOS :

  iOS Human Interface Guidelines.pdf

  Ce guide fournit de nombreux conseils pour bien construire ses interfaces, qui dépassent le cas particulier du système iOS.

- Voir aussi le site Microsoft sur les bons usages dans le développement d'interface utilisateur, qui vise à « mettre l'utilisateur au centre du processus de développement » (*usability*)

  https://msdn.microsoft.com/en-us/library/windows/desktop/ff728829(v=vs.85).aspx

  et la page Microsoft sur le développement sécurisé (pour les développeurs C#) :

  https://msdn.microsoft.com/en-us/library/windows/desktop/ee663293(v=vs.85).aspx

- L'université *Carnegie Mellon* met en ligne un cycle de cours, sur la sécurisation des interfaces utilisateur (*usable security*) qui met en évidence l'importance de cette discipline intermédiaire entre l'ergonomie et la sécurité

  http://cups.cs.cmu.edu/courses/ups-sp14/

5. CONCLUSION

Cette séance a mis en évidence que la qualité du code, aussi importante soit-elle, n'était pas le seul moyen pour sécuriser une application : avant toute vérification par code, il faut anticiper les vulnérabilités et tenter de les éviter, en sécurisant l'interface utilisateur.

Cette sécurisation s'appuie d'abord sur une bonne conception qui limite la surface d'attaque, en réduisant l'exposition des données et les fonctionnalités accessibles, en partitionnant les grosses applications en petites applications ciblées, en authentifiant correctement les utilisateurs et en leur attribuant des droits avec parcimonie.

Mais elle passe aussi dans le détail des écrans par le choix de composants graphiques appropriés qui préviennent certaines erreurs ou attaques, en évitant donc au code de les corriger.

Dans tous les cas, l'interface utilisateur n'est que le premier niveau de défense de l'application (le rempart extérieur) qui doit jouer son rôle dans une stratégie globale (« défense en profondeur »).

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