



Sécurité dans les développements d'applications :
Méthodes de gestion des exigences et outils ...
Sécurité dans le cycle de vie des applications
Gestion des patchs (diffusion...)

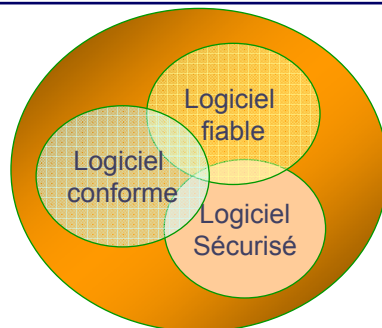
Clusir RA : Présentation du 25 avril 2007
Eric Deronzier - Ysosecure



État des lieux

- La sécurité dans les développements dans le domaine de la gestion est hétérogène :
 - Généralement « plutôt bien traitée » chez de grands éditeurs et certaines SSII
 - Plutôt « oubliée » pour les développements « in house » dans ses différentes dimensions :
 - **Disponibilité** : robustesse, intégration dans le PRA, mode dégradé
 - **Confidentialité** : anonymisation des bases de tests, cryptage des flux ou des bases, confidentialité des certificats
 - **Traçabilité** ou gestion de la preuve
 - **Intégrité** : contrôles de saisie, contrôles de cohérence, contrôles aux limites, saisie des paramètres ...

Qu'est ce qu'un logiciel sécurisé ?



- Qui garantit :
 - la confidentialité des informations qu'il contient
 - l'intégrité des données
 - les performances et la disponibilité requises

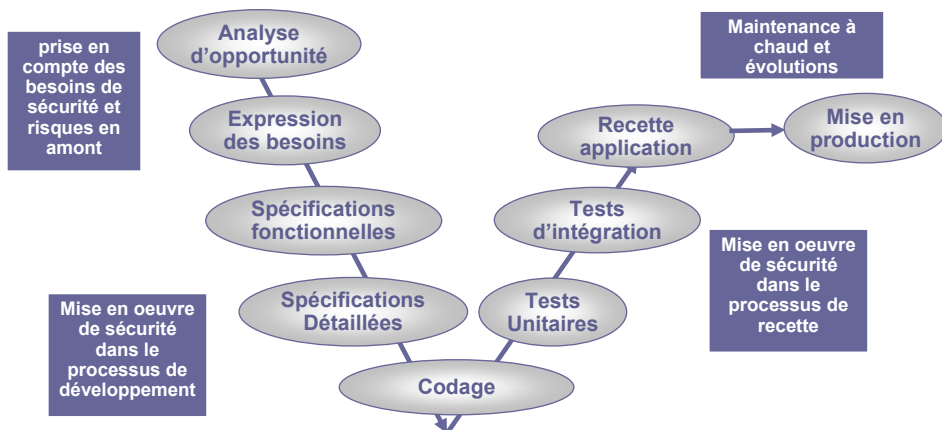
Validité
Extensibilité
Réutilisabilité
Compatibilité
Efficacité
Portabilité

Vérifiabilité
Intégrité
Facilité d'emploi
Economie
Documentation
Compréhensibilité

Flexibilité
Interopérabilité
Modularité
Modificabilité
Disponibilité
Performances

Adaptabilité
Utilisabilité
Clarté
Maintenabilité
Testabilité
...

- La sécurité dans les développements peut être :
 - La gestion des risques dans le cycle de vie des développements
 - La prise en compte des besoins de sécurité (DIC) dans le cycle de vie des développements (gestion des exigences)
 - La sécurisation du code
 - L'anonymisation des données de tests
 - Le cryptage des bases et des échanges
 - La maintenance à chaud
 - L'organisation des tests sécurité lors du passage en production
 - Diffusion des corrections, patches
 - ...



Tentative de regroupement de la sécurité dans un cycle en V

Identification des
besoin de sécurité de
l'application

Conception
de l'application

Développement
de l'application

Recette
de l'application

Mise en
exploitation

Maintenance
applicative, à chaud

- Apprécier les risques liés au projet :
 - Les risques projet,
 - Les risques organisationnels,
 - Les risques technologiques
- Identifier les exigences de sécurité
 - Liés aux risques fonctionnels, d'architecture
 - Liés aux besoins juridiques/réglementaires,
- En déduire des contraintes :
 - de sauvegarde, d'archivage, de continuité, ...
 - de sécurisation de code
- En déduire des protocoles de recette sécurité :
 - Audit sécurité (et éventuellement tests intrusifs)
 - Tests des fonctions sécurité et Recette
- Procédure de vérification des paramètres de sécurité avant mise en exploitation (PRA, paramètres,)
- Procédure de gestion des maintenances à chaud
- Gestion des évolutions
- Diffusion des corrections

7



**Comment est traité la sécurité des développements
dans les normes et méthode**

Mehari , Ebios,
ISO 2700x CMMi

- La base de toute l'analyse des risques est l'analyse des enjeux des métiers
 - Dédit une classification des **processus** et des **informations**
 - Donc des applications en DICP
 - Donc potentiellement des programmes
 - Mais attention, difficile de demander aux études une criticité en D et C

10 Sécurité des projets et développements applicatifs

A - Organisation des développements, de la maintenance et de la gestion des changements

10A01	Prise en compte de la sécurité dans les projets et développement
10A02	Gestion des changements
10A03	Externalisation du développement logiciel
10A04	Organisation de la maintenance applicative
10A05	Modification des progiciels

B - Sécurité des processus de développement et de maintenance

10B01	Sécurité des processus des développements applicatifs
10B02	Protection de la confidentialité des développements applicatifs
10B03	Sécurité relative aux traitements internes des applications
10B04	Protection des données d'essai
10B05	Sécurité de la maintenance applicative
10B06	Maintenance à chaud

12.1 Exigences de sécurité vis à vis des systèmes d'information	12.1.1 Analyse et spécifications des exigences de sécurité
12.2 Fonctionnement correct des applications	12.2.1 Validation des données d'entrée 12.2.2 Contrôle interne du traitement 12.2.3 Intégrité des messages 12.2.4 Validation des données de sortie
12.3 Moyens cryptographiques	12.3.1 Politique d'utilisation des mesures cryptographiques 12.3.2 Gestion des clés
12.4 Sécurité des fichiers systèmes des applications	12.4.1 Contrôle des logiciels opérationnels 12.4.2 Protection des jeux d'essai des applications 12.4.3 Contrôle d'accès aux bibliothèques de programmes sources
12.5 Sécurité des processus de développement et de support	12.5.1 Procédures de contrôle des modifications 12.5.2 Examen technique des modifications apportées au système d'exploitation 12.5.3 Restrictions sur les modifications apportées aux progiciels 12.5.4 Fuite d'informations 12.5.5 Externalisation du développement de logiciel
12.6 Gestion des vulnérabilités	12.6.1 Contrôle des vulnérabilités

■ Les vulnérabilités génériques d'EBIOS V2

MAT_ACT 33	Le matériel utilisé permet un autre usage que celui qui est prévu (développement de logiciels non destinés à l'organisme...)
LOG 26	Possibilité d'existence de fonctions cachées introduites en phase de conception et développement
LOG 30	Absence de qualification des développements dans un contexte représentatif de l'exploitation
LOG 24	Possibilité d'installer des correctifs, mises à jours, patches, hotfixes...
PER_DEC 32	Existence de composants désuets dans l'infrastructure de traitement de l'information (développement dans des langages plus utilisés...)
...	



CMMi et la sécurité



De quoi s'agit-il ?

**CMMI est un modèle d'évaluation,
Ce n'est pas une méthodologie Projet !**

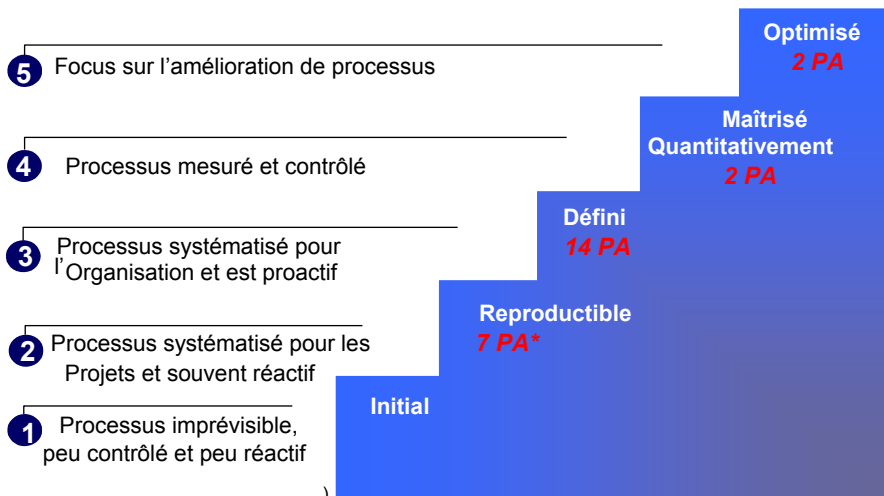
■ Intérêt de CMMI :

- Démarche permettant de mesurer de façon objective les améliorations des processus
- Souplesse / Sur mesure
 - Liberté du choix des mises en application
 - Progression selon besoin
- Coordinations entre département
 - Uniformité du langage
 - Adaptations
- Comparaisons possibles

25 Process Area (Macro Processus)

2	Gestion des exigences (REQM) Planification de projet (PP) Suivi et contrôle de projet (PMC) Mesures et Analyses (MA) Assurance Qualité (PPQA) Gestion de configuration (CM) Gestion des fournisseurs (SAM)
3	Développement des exigences (RD) Solutions techniques (TS) Intégration de produits (PI) Vérification (VER) Validation (VAL) Focalisation sur processus organisationnel (OPF) Définition des processus de l'organisation (OPD) Formation de l'organisation (OT) Gestion de projet intégrée (IPM) Gestion des risques (RSKM) Analyse et prise de décision (DAR) Environnement organisationnel pour intégration (OEI) Coordination des équipes internes (IT) Coordination des fournisseurs (ISM)
4	Performance du processus organisationnel (OPP) Gestion de projet quantitative (QPM)
5	Déploiements et changements organisationnels (OID) Analyse des causes et résolution (CAR)

Niveau de maturité par étage



* PA : Process Area (Macro Processus)

- **Secteur clef (PA)**

Gestion des exigences

- **Objectif spécifique**

SG1: Gérer les exigences

- **Pratiques spécifiques**

SP 1.1 Obtenir une compréhension des exigences

SP 1.2 Obtenir l'engagement sur les exigences

SP 1.3 Gérer les changements des exigences

SP 1.4 Maintenir la traçabilité bidirectionnelle des exigences

SP 1.5 Identifier les incohérences entre les produits du projet et les exigences

- CMMi n'aborde pas directement les questions d'urbanisme, de qualité de la formulation des exigences, de besoins de sécurité de l'information :
 - CMMI suppose ces questions-là déjà traitées,
 - il ne s'interroge pas sur la pertinence de l'expression de besoin, sa cohérence mais se limite à organiser la réalisation.



Prise en compte des besoins de sécurité et risques en amont

➤ Gestion des risques

Gestion des exigences de sécurité



Principes de la méthode

■ La méthode INCAS a pour objectif d'inclure la notion de sécurité dans un projet en cours

- Se compose d'une suite d'actions de sécurité courtes (1 h à 1 j)
répartie sur tout le cycle de vie de conception et de
développement de l'application
 - S'intègre dans toute méthode de conduite de projet de
développement
- 4 facteurs de base DICP et classification des risques identique à
celle de MEHARI

- La méthode INCAS est applicable dans les cas suivants
 - Conception et développement de projets traditionnels de développement
 - Acquisition de logiciels ou progiciels
 - Développement sur micro-ordinateurs (mais avec enjeux métiers forts ...)
 - Maintenance évolutive lourde ou migration

- Elle traite l'amont (risque et exigences de sécurité) mais également (de façon peu approfondie) la partie réalisation et recette

■ Sur les aspects risques projet

	Domaines étudiés	Note	Pond	Note * Pond
1	Risques liés à la taille du projet			
11	Importance du développement (charges/délais)		5	0
12	Nombres de directions, fonctions concernées			
13	Planification et suivi budgétaire		5	0

Charges / Délais :	3 à 9 m	9 à 12 m	12 à 24 m	+24m
3H*mois à 6H*m	1	2	3	4
6H*m à 30H*m	2	1	2	3
30H*m à 6H*années	3	2	1	2
+ de 6H*années	4	3	2	3

Directions	Note
1 à 2	1
3 à 4	2
5 et +	3

Suivi régulier de la planification et du budget	1
Suivi de temps à autre et sans contrôle	2
Planification et/ou suivi budgétaire rare	3

Exemples issus de la méthode INCAS

■ Sur les aspects risques organisationnels

Domaines étudiés	Note	Pond	Note * Pond
2 Risques liés à l'organisation			
21 Impact (%) des fonctions à automatiser		3	0
22 Impact des modifications de procédures		3	0
23 Impact des modifications de structures Utilisateurs		5	0
24 Appréciation de l'Utilisateur (Direction et final)		5	0
25 Participation des fonctions utilisatrices		5	0
26 Existence d'une équipe de projet		4	0
27 Définition et fonctionnement des structures du projet		5	0
28 Expérience du management du projet		6	0
Total des notes pondérées			

Impact	Note
0% à 25%	3
25% à 50%	2
> 50%	1

Modifications de procédures	Note
Formation nécessaire des personnels	3
Légères modifications	2
Sans conséquences	1

Modifications de structures	Note
Déplacement de personnel	3
Réorganisation partielle du service	2
Modifications sans conséquence	1

Structures ou principes non définis	Note
en partie définis	3
clairement définis	2
	1

Participation des fonctions	Note
Peu ou pas impliquées	3
Attitudes concernées	2
Attitudes responsables ou fortement impliquées	1

Appréciation de l'Utilisateur	Note
Réservé sur l'apport du projet	3
D'accord avec quelques réserves	2
Positive ou demandeur	1

Equipe de projet	Note
Non existante	3
Représentant Utilis. à temps partiel	2
Représentant Utilis. conforme au projet	1

Expérience	Note
équipe faible	3
équipe moyenne	2
équipe forte	1

■ Sur les aspects risques technologiques

Domaines étudiés	Note	Pond	Note * Pond
3 Risques technologiques			
31 Aspects Matériels (matériels, sites nouveaux, etc.)		3	0
32 Aspects Applicatifs (logiciels nouveaux ou modifiés)		5	0
33 Connaissance informatique des Utilisateurs		4	0
34 Connaissance du domaine des Utilisateurs		6	0
35 Connaissance du domaine par les Informaticiens		3	0
Total des notes pondérées			

Notes cumulatives	Note
Pas de matériel nouveau	1
Ajout de nouveaux types de périphériques	2
Ajout d'un CPU ou serveur	3
Ajout de PC en environnement hétérogène	3
Ajout de réseaux (WAN, LAN, etc.)	3
Création d'un nouveau site	4

Notes cumulatives	Note
Pas d'outil logiciel nouveau	1
Nouveau langage de développement	2
Nouveau SGBD	2
Nouveau protocole de télécommunication	2
Nouvelle technologie	2
Nouveau logiciel d'édition	2
Nouveau logiciel de communication	2
Nouveau progiciel	2
Nouvelle technique de développement	3

Connaissance des Utilisateurs	Note
N'utilise pas d'outils informatiques	3
Travail sur un micro en local	2
Connait une méthode et la programmation	1

Utilisateurs connaissant	Note
Peu le sujet traité dans le projet	3
Le sujet qui fait partie du travail courant	2
Bien le sujet traité	1

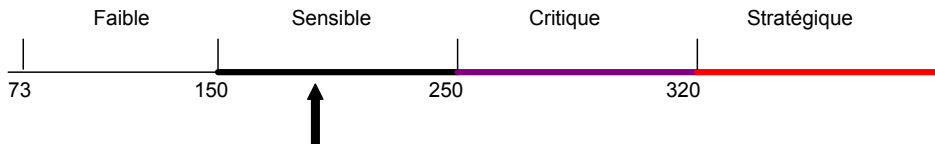
Informaticiens connaissant	Note
Pas le sujet traité ou débutants	3
Déjà traité le sujet	2
Bien le sujet traité ou expérimentés	1

■ Règles d'analyse

Au-delà du seuil de notation 250 des mesures doivent être prises en ce qui concerne les risques liés :

- à la taille du projet
- à l'organisation du projet
- aux risques technologiques & fonctionnels

Echelle de notation du niveau de risque



Niveau de risque lié au développement du projet :

Faible 1 Sensible 2 Critique 3 Stratégique 4

■ Intérêt de cette étape :

- Par une analyse enjeux/risques partagée savoir graduer l'effort en terme de :

Mesures de sécurité (classifiées DIC)

Exigences en terme de sécurité du code

- Revue (interne/externe)
- Tests de robustesse

2 ou 3 niveaux d'applications

Exigences en terme de recette

Phase : Etude Préalable		Type de préjudice	Valorisation du risque	Prise en compte de la mesure	Responsable de la mesure	Coût des mesures de sécurité
Classification Préalable	Nature du Risque					
	D I C A E M					
1						
2						



Prise en compte des besoins de sécurité et risques en amont

Gestion des risques

➤ Gestion des exigences de sécurité



Sécurité dans la gestion des exigences

- Dans l'ingénierie des besoins, la **méthode VOLERE** est une des plus utilisées en Europe :
 - Très adaptée si développement Interne
 - Combine les exigences, leur gestion et la gestion de projet
 - Chaque Item est détaillé (Instructions, justifications, description, exemple)
 - Liste détaillée d'exigences non fonctionnelles et notamment de sécurité
 - Structure de description des exigences
 - Pratique et Facile à utiliser

<http://www.volere.co.uk/>

- **Contenu** : Liste des personnes ou systèmes qui peuvent accéder au système (aux fonctionnalités et aux données), dans quelles circonstances et pour quelles parties du produit.
- **Objectif de la section** : Comprendre les attentes en matière de confidentialité en ce qui concerne le produit.
- **Exemples**
 - Le produit doit contrôler que des données qui pourraient endommager les données existant déjà dans le produit ne peuvent être introduites.
 - Le produit doit se protéger lui-même d'abus intentionnels.
 - Les utilisateurs recevront des fichiers clients mis à jour toutes les 24 heures.
- **Exemples de questions**
 - Y a-t-il des données que la direction souhaite protéger ?
 - Y a-t-il des processus qui pourraient causer des dégâts ou qui pourraient être utilisés pour un profit personnel ?
 - Y a-t-il des gens qui ne devraient pas avoir accès au système ?
 - Quelles sont les permissions d'accès ?
 - Qu'y a-t-il à protéger (données, logiciels, lancement de traitements, matériel) ?
- **Critère de satisfaction** :
 - Nom de la fonction ou des données faisant l'objet des exigences de confidentialité.
 - Rôles ou noms des utilisateurs qui peuvent utiliser certaines fonctions ou données

- **Contenu** : Liste des personnes ou systèmes qui peuvent accéder au système (aux fonctionnalités et aux données), dans quelles circonstances et pour quelles parties du produit.
- **Objectif de la section** : Comprendre les attentes concernant l'intégrité des données du système, spécifier ce que le produit fera pour assurer son intégrité dans le cas où un événement indésirable comme une attaque de l'extérieur ou une utilisation erronée du produit par un utilisateur non autorisé advenaient.
- **Exemples**
 - Le produit doit contrôler que des données qui pourraient endommager les données existant déjà dans le produit ne peuvent être introduites.
 - Le produit doit se protéger lui-même d'abus intentionnels.
 - Les utilisateurs recevront des fichiers clients mis à jour toutes les 24 heures.
- **Critère de satisfaction** :
 - RTO/RPO de l'application.
 - ...

■ Exemples de questions

- Sauvegardes et restauration des données et de l'application
 - Comment doivent être faites les sauvegardes des données de production ? Seront-elles effectuées périodiquement ?
 - La procédure de sauvegarde des données de production doit-elle être automatique ?
 - Toutes les données stockées sur les disques devront-elles pouvoir être sauvegardées, archivées, restaurées, purgées sans avoir à arrêter les applications ?
 - Le système devra-t-il générer une alarme en cas de dépassement des tailles maximales des fichiers de données ou des bases définis dans une liste spécifique ?
- Restauration des connexions réseau :
 - Que souhaitez-vous en matière de sécurité réseau ? Souhaitez-vous exclure cette partie du projet ?
 - Comment imaginez-vous un crash réseau ? Une reprise à chaud ? Une reprise à froid ?
- Obsolescence et mise à jour des informations
 - Comment l'information sera-t-elle utilisée ?
 - Quel est l'impact sur le métier de ceux qui achètent le produit si l'information est erronée ?
 - Y aura-t-il des problèmes si différents utilisateurs ont des versions différentes du système ?
 - Quand les acheteurs du produit doivent-ils être informés des modifications ? Tous les mois ? Cinq minutes au plus tard après chaque modification ? En temps réel ? Chaque fois qu'ils le demandent ?
 - Tous les acheteurs ont-ils besoin de la même fréquence de mise à jour ?
 - Quel est l'impact sur les utilisateurs du produit si les informations sont obsolètes ?

31

- **Contenu** : Spécification de ce que le produit doit faire pour assurer la protection de la vie privée des personnes sur lesquelles des informations sont stockées. Le produit doit aussi être conforme aux lois sur la protection des données à caractère personnel..
- **Objectif de la section** : S'assurer que le produit est conforme à la loi, et protéger les données à caractère personnel stockées. Les gens sont actuellement peu indulgents envers les organismes qui ne respectent pas leur vie privée.
- **Exemples**
 - • Le produit doit informer les utilisateurs de l'usage qui sera fait des informations les concernant avant de collecter ces informations.
 - • Le produit doit informer les personnes sur lesquelles des données à caractère personnel sont stockées des changements de politique concernant les données à caractère personnel.
 - • Le produit ne divulguera des informations d'ordre privé que conformément avec la politique de l'entreprise concernant les données à caractère personnel.
 - • Le produit protégera les données à caractère personnel conformément aux lois applicables concernant ces données / à la politique de l'entreprise concernant les données à caractère personnel.

32

- **Contenu** : Spécification de ce que le produit doit faire (habituellement conserver des enregistrements) afin de permettre les contrôles nécessités par les audits.
- **Objectif de la section** : Construire un système en accord avec les règles d'audit applicables.
- **A prendre en compte**
 - Cette section peut avoir des implications légales. On vous conseille de faire valider les exigences de ce chapitre par les auditeurs de votre organisme.
 - Vous devez aussi vous demander si le produit doit conserver des informations sur qui l'a utilisé, quand, pourquoi ; qui a modifié telle information, quand, pourquoi... L'objectif est d'augmenter la sécurité : une personne ne pourra pas nier avoir utilisé le produit ni avoir participé à telle transaction en utilisant le produit.



La sécurisation du code

➤ Base des malveillances

Exemple de méthode (Stride)

Outil de revue de code



CWE List

Full Dictionary View
Classification Tree
Other Views

About

Sources
Process
Documents

Community

Related Activities
Discussion List

News

Calendar
Free Newsletter

Compatibility

Program
Requirements
Declarations
Make a Declaration

Contact Us

Search the Site

Individual CWE Dictionary Definition (draft 5)

Hard-Coded Password	
CWE ID	259
Description	Hard coded passwords may compromise system security in a way that cannot be easily remedied. It is never a good idea to hardcode a password. Not only does hardcoding a password allow all of the project's developers to view the password, it also makes fixing the problem extremely difficult. Once the code is in production, the password cannot be changed without patching the software. If the account protected by the password is compromised, the owners of the system will be forced to choose between security and availability.
Likelihood of Exploit	Very High
Weakness Ordinality	Primary (Weakness exists independent of other Weaknesses)
Causal Nature	Explicit (This is an explicit weakness resulting from behavior of the developer)
Common Consequences	Authentication: If hard-coded passwords are used, it is almost certain that malicious users will gain access through the account in question.
Potential Mitigations	Design (for default accounts): Rather than hard code a default username and password for first time logins, utilize a "first login" mode which requires the user to enter a unique strong password. Design (for front-end to back-end connections): Three solutions are possible, although none are complete. The first suggestion involves the use of generated passwords which are changed automatically and must be entered at given time intervals by a system administrator. These passwords will be held in memory and only be valid for the time intervals. Next, the passwords used should be limited at the back end to only performing actions valid to for the front end, as opposed to having full access. Finally, the messages sent should be tagged and checksummed with time sensitive values so as to prevent replay style attacks.
Demonstrative Examples	The following code uses a hardcoded password to connect to a database: <pre>... DriverManager.getConnection(url, "scott", "tiger"); ...</pre>



La sécurisation du code

Base des malveillances

➤ Exemple de méthode (Stride)

Outil de revue de code



STRIDE de Microsoft

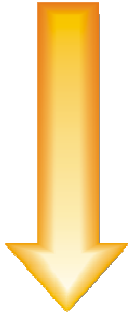
Acronyme de :
Spoofing (usurpation),
Tampering (falsification),
Repudiation (répudiation),
Information Disclosure (divulgarion d'informations),
Denial of Service (refus de service)
Elevation of Privilege (élévation du privilège)



Modèle STRIDE des catégories de menaces

Terme	Définition
Usurpation d'identité	Obtention par des moyens illicites et utilisation des informations d'authentification d'une tierce personne, telles qu'un nom d'utilisateur ou un mot de passe.
Falsification de données	Modification malveillante des données.
Répudiation	Fait que des utilisateurs nient avoir exécuté une action, alors qu'il n'existe aucun moyen de prouver le contraire. (La non répudiation fait référence à la capacité d'un système à contrer les menaces de répudiation et comprend des techniques telles que la signature d'un bordereau de réception d'un colis pouvant servir de preuve.)
Divulgarion d'informations	Exposition d'informations à des individus qui ne sont pas sensés y avoir accès, par exemple l'accès à des fichiers sans possession des droits appropriés.
Refus de service	Tentative explicite visant à empêcher des utilisateurs légitimes d'utiliser un service ou un système.
Élévation des privilèges	Obtention de privilèges d'accès par un utilisateur n'en possédant pas, par exemple en trouvant un moyen d'être ajouté au groupe Administrateurs.

Processus de modélisation des menaces

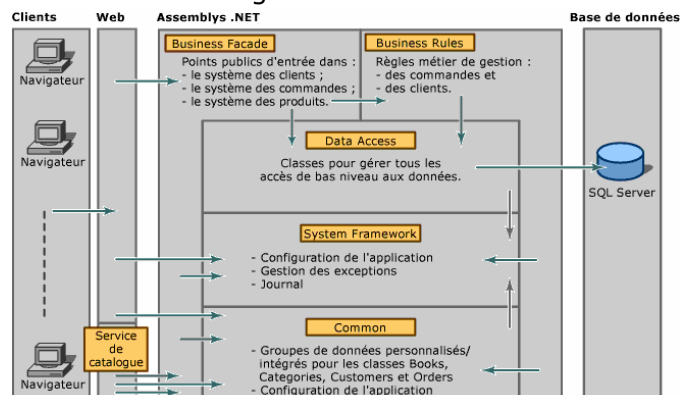


- 1 Identifier les ressources (actifs)
- 2 Créer une vue d'ensemble de l'architecture
- 3 Décomposer l'application
- 4 Identifier, documenter, évaluer les menaces

Conception de l'application

■ Étape 2 : Créer une vue d'ensemble de l'architecture

- Identifiez ce que fait l'application
- Créez un diagramme de l'architecture
- Identifiez les technologies utilisées



■ Etape 3 : Décomposez l'application

- Créez un profil de sécurité basé sur les domaines de vulnérabilité classiques
- Examinez les interactions entre les différents sous-systèmes
- Utilisez des diagrammes de flux de données (DFD) ou diagrammes UML

Identifier les frontières sécurisées



Identifier le flux de données



Identifier les points d'entrée



Identifier le code privilégié



Documenter le profil de sécurité

43

■ Etape 4 : Identifier les menaces et les mesures de sécurité

- **développer des applications qui résistent mieux aux attaques (STRIDE) :**

Types de menaces	Exemples
USurpation	<ul style="list-style-type: none"> • Falsification de messages électroniques • Rediffusion de paquets d'authentification
FalsificaTion	<ul style="list-style-type: none"> • Modification des données lors d'une transmission • Changement des données dans des fichiers
Répudiation	<ul style="list-style-type: none"> • Suppression d'un fichier important et déni de l'action • Achat d'un produit et déni de l'action
DivulgaTion d'Informations	<ul style="list-style-type: none"> • Exposition d'informations dans des messages d'erreur • Exposition de code sur des sites Web
Deni de service	<ul style="list-style-type: none"> • Submersion d'un réseau avec des paquets SYN • Submersion d'un réseau avec des paquets ICMP falsifiés
Elévation de privilèges	<ul style="list-style-type: none"> • Exploitation du débordement de mémoire tampon pour obtenir des privilèges système • Obtention illégale des privilèges d'administrateur

44



La sécurisation du code

- Base des malveillances
- Exemple de méthode (Stride)
- Outil de revue de code



Techniques de revues/sécurisation de code - Principes

- La plupart des « exploits » sont liées à des bugs de programmation.
- En respectant des principes simples de développement et en utilisant des outils spécifiques, il est possible de minimiser les erreurs résiduelles lors de la mise en production.
- Cela nécessite une chaîne de production rigoureuse.

- **Trois grandes catégories de solutions existent :**
 - La première concerne les **tests sur le code source** : Ils vérifient que la construction du code est conforme aux règles de programmation du langage utilisé (dead code, uninitialized variable etc.)
 - langage Java, C, C++ ou tout autre langage actuellement disponible
 - La deuxième concerne les **tests fonctionnels** : Ils détectent les erreurs qui ne se déclenchent que lors de l'exécution (buffer overflow, dangling pointers etc.)
 - concerne aussi bien les interfaces utilisateurs, les API, la gestion des bases de données, la sécurité, que le réseau
 - La troisième catégorie concerne **les tests de performances des logiciels développés**
 - Concerne les applications Web, intranet ou des Web services
- Plus de 278 outils référencés sur le site spécialisé www.stickminds.com

- **Analyse statique :**
 - Détecter les défauts le plus tôt possible.
 - Assurer une meilleure lisibilité : maintenance et réutilisation.
 - S'assurer que le code soit testable.
 - Contrôler l'architecture du projet.
- **Analyse dynamique :**
 - Tester la fonction du module.
 - Tester la structure du module.
 - Tester l'interfaçage entre les modules.
 - Tester la robustesse du code.

- Les outils **PRQA** sont destinés à être mis au service des différents acteurs du process de développement logiciel:
 - Développeurs, Ingénieurs méthode, Ingénieurs qualité logicielle, Chef de projet...
 - Dispose pour chaque langage de plus de 1000 règles standard.
 - Moteur d'analyse développé spécifiquement pour chaque langage
 - Permet d'implémenter ses propres règles de vérification.
 - Navigation simple et efficace.
 - Analyse module par module.
 - Pas de vérification dynamique



Panorama de tests de code source (issue de JD Net solutions)

Solutions	Caractéristiques
BullseyeCoverage	Pour C++ et C. Compatibilité étendue avec les mondes Windows, Unix et embarqué.
Rational Purify	Proposé en version Windows et Linux/Unix, pour le langage C/C++ (détection de corruption en mémoire) et Java, C/C++, Visual C++, C# et VB.Net (détection des fuites mémoire).
Cantata++	Permet de tester ANSI C, ISO C++ et EC++, en tests unitaires et d'intégration, sur systèmes hôtes ou systèmes embarqués. Compatible Windows XP, Solaris (2.x), HP-UX et Linux.
JCover, JStyle, JVerify	Société indienne spécialisée dans les outils pour Java.
C++Test, .TEST, JTest, Insure++	C++Test est un produit de prévention automatique des erreurs pour C et C++ tandis que .TEST s'applique à l'environnement .Net de Microsoft et que JTest concerne Java. Insure++ est un outil automatique de détection des problèmes de mémoire pour C/C++.
QuickAnt Test Pro, QuickAnt,	Permettent d'automatiser les processus de test en Java. La version QuickAnt est alléguée par rapport au produit QuickAnt Test Pro. Pour C, C++ et C#, la solution PreciCode est adaptée. Compatibilité Windows NT, 2000, XP et Linux (sauf Precilog).
QA·C, QA·C++, QA·J,QA·FORTRAN	Les solutions permettent de détecter les erreurs de code dans, respectivement, les langages C, C++, Java et Fortran. Compatibilité plates-formes Windows, Sun, HP, Redhat Linux et Slackware Linux.
CTA++, CTC++, CMT++, CMTJava	Société finlandaise. Propose une gamme complète d'outils de test pour les langages C, C++ et Java, au niveau des classes, bibliothèques et sous-systèmes. L'outil CTA++ s'intègre à Visual Studio.



La sécurité dans les processus de recette

- Implication fonction sécurité



Sécurité dans les processus de recette

- Il s'agit de vérifier avec le concours de la fonction RSSI (si elle existe) ou de la Direction Technique :
 - De vérifier l'implémentation des mesures de sécurité
 - De préparer les mesures de sécurité nécessaires pour la mise en production
 - De réaliser d'éventuels tests de performance
 - De préparer les évolutions du PRA
- Cette tâche est très **IMPORTANTE** ; car elle doit **consolider toutes** les mesures de sécurité ainsi que les mesures de sécurité planifiées dans les phases précédentes.

- En environnement de pré production :
 - A l'issue de cette phase, la synthèse sécurité doit valider et vérifier :
 - la mise en relief des services de sécurité technique à prendre en compte pour le système d'information futur ;
 - d'autre part, des mécanismes et moyens techniques complémentaires à consolider ou à mettre en oeuvre,
 - l'actualisation de la cible (disponibilité, intégrité, confidentialité) de sécurité visée,
 - A ce niveau du projet, des éléments clés :
 - contrat de service,
 - le fonctionnement dégradé avec des consignes utilisateurs opérationnelles,
 - les plans de sécurité (PRA, PCM, sauvegardes, ...)
 - doivent être vérifiés, voire contrôlés par un audit (interne ou externe) avant la signature d'un document contractuel.

- Mise en place d'une check-list type par niveau d'application
 - Audit sécurité

Aspect de sécurité à traiter		Oui	Non	Commentaires
1	Le dossier de recette contient-il les résultats de tests liés à la sécurité :			
	? tests d'accès illicites à l'information en modification, ou en copie ?			
	? tests de plantage et reprise à chaud ?			
	? examen de la lisibilité des traces par un non informaticien ?			
	? accès aux transactions utilisateurs interdit à la fonction d'administration du serveur ?			
	? trace systématique de toutes les actions effectuées par l'administration ?			
2	Le propriétaire a-t-il donné son aval sécurité pour :			
	? les outils de contrôle et de suivi du bon fonctionnement de l'application ?			
	? l'organisation du contrôle (qui, quoi, comment) ?			
3	Est-ce que les règles de gestion (paramétrage UNIX, NT) retenues pour			
	? couple identifiant/authentifiant unique et propre à chaque utilisateur ?			
	? mot de passe non trivial ?			



La diffusion des patches

- Exploitation des failles de sécurité dans les développements
 - Outils



Les enjeux de réactivité

- Vulnérabilité dans l'interface RPC de la LSA des systèmes Windows (MS04-011)
 - Rapportée par eEye à Microsoft : 8 Octobre 2003
 - Correctif MS04-011 publié par Microsoft : 13 Avril 2004 (+ **188 jours**)
 - Publication par eEye des détails techniques sur la vulnérabilité : 13 Avril 2004
 - Exploitation par le vers Sasser : 1er Mai 2004 (+ **17 jours**)
- Vulnérabilité dans le module de décodage du protocole ICQ des sondes de détection d'intrusion (IDS) d'ISS
 - Rapportée par eEye à ISS : 8 Mars 2004
 - Correctif publié par ISS : 18 Mars 2004 (+ **10 jours**)
 - Publication par eEye des détails techniques sur la vulnérabilité : 18 Mars 2004
 - Exploitation par le vers Witty : 19 Mars 2004 (+ **1 jour**)

Source Hervé Schauer Consultants

- Chaque éditeur propose son propre outil de diffusion plus ou moins automatisé ou automatisable.
L'inconvénient c'est pour une configuration complète il faut utiliser et superviser parfois plus d'une dizaine d'outils différent. Ces outils sont rarement capable de faire du reporting et d'indiquer quel est le nombre de machines qu'ils ont touchées.
- Les outils de télédistribution permettent d'avoir un outil homogène pour tous les éditeurs une majorité de plates-formes, mais au prix d'un effort de « repackaging ». Ils ne prennent pas en compte toutes les plates-formes composant le SI (notamment les appliances).
- Aucun outil ne propose d'automatiser le retour arrière si une anomalie bloque le système d'information (Et pour cause, puisque seul l'éditeur du patch peut permettre sa désinstallation). Il existe nécessairement une présence et un plan de secours complémentaire.

- Outils de patch virtuels (virtual patching)
 - Ils simulent en ligne le fonctionnement des correctifs de sécurité des éditeurs en intervenant sur les flux réseau en temps réel.
 - **Le correctif virtuel** est une technique consistant à filtrer les flux destinés à un serveur (serveur de fichier, de messagerie, serveur de bases de données Oracle) en interceptant les trames non-conforme avant leur interprétation par le système central
 - C'est une solution qui évite à l'administrateur d'avoir à immédiatement « appliquer les rustines », action qui peut toujours provoquer des régressions dramatiques.
 - Exemple de Virtual patching : « Blue Lane
 - Intérêt de cette technique en virtualisation

■ En synthèse :

- Évidemment partir des enjeux métiers et tenter de classer les applications
- Faire formaliser les exigences de sécurité par vos Utilisateurs en interne et par vos Clients (mais capacité à limiter leurs rêves)
- Définir les mesures de sécurité dès la phase de conception générale
- Suivant les langages et la maturité des équipes, organiser une revue de code interne ou externe, ou des tests d'intrusion applicatifs
- Penser que l'intégration des exigences sécurité coûte quelques points dans le cycle de vie mais coûte **11** fois moins cher qu'après mise en production



La certification des développeurs la solution miracle

Une initiative récente du SANS

- **Introduction :** Governments, companies, and educational institutions are doomed to deal with endless streams of software vulnerabilities unless programmers learn to write much more secure code.
- Several initiatives are underway to improve secure programming skills and knowledge.
 - Symantec, Oracle, Microsoft, and a few other software companies are conducting short courses for their programmers;
 - software firms like SPI Dynamics and Fortify Technology are working with universities to provide automated, real-time feedback to student programmers; and dozens of universities are creating elective courses on secure programming.
 - Yet, even if all of those initiatives are successful, they are unlikely to affect even two percent of the existing 1.5 million programmers already in the work force or those who will be entering the work force over the next five years.

- Le SANS Institute (SysAdmin, Audit, Network, Security), propose une certification GIAC Secure Software Programmer (GSSP) pour enseigner aux développeurs les bonnes pratiques pour programmer de manière sécurisée :
 - 4 langages de développement sont couverts : C/C++, Java/J2EE, Perl/PHP et .NET/ASP.
 - Les postulants apprendront ainsi à identifier et corriger les erreurs de programmation générant des failles de sécurité, comme le dépassement de mémoire tampon ...