SDLC (secure development life-cycle)

Sécurité applicative / sentiment de sécurité / Risques / SDLC

Auteur
Jean-François Casquet
Editeur
AZERTY Microsystem

www.65120.net/stage/

INTRODUCTION. Définition de la Sécurité avec un grand 'S'

1. Gérer les pannes et défaillances	Souvent appelé « PCA » Plan de Continuité d'activité. Il s'agit des moyens mis en place pour s'assurer de la continuité de service. **La guerre contre la défaillance matérielle et attaques virales.**
2. Droits aux ressources	Il s'agit des droits en lecture, modification, création, suppression, impression, sauvegarde, maintenance. Tous ces droits sont liés à une ressource par rapport à des usagers. **Guerre contre les pirates.**
3. Cohérence des informations	Il s'agit de moyens mis en place pour s'assurer de la cohérence des informations stockées. Souvent, ce chapitre est relégué aux développeurs ou aux DBA (DataBase Administrator). **Guerre contre les Crackers.**
4. En cas de sinistre grave	Souvent appelé « PRA » Plan de Reprise d'Activité. Il s'agit des moyens mis en œuvre pour reprendre une activité après sinistre grave. **Guerre contre le Crash.**
5. Sauvegarde	Il s'agit des moyens mis en place pour Sauvegarder, Archiver, Stocker mais surtout : « les moyens de Restaurer » le SI. **Guerre contre l'erreur de manipulation**.
6. Authentifier	Il s'agit de sécuriser les transports d'information avec les moyens techniques qui permettent une confidentialité des informations**. Guerre contre le Flooding**.

1. Pourquoi de la sécurité applicative ?

Le modèle ISO amélioré pour la sécurité

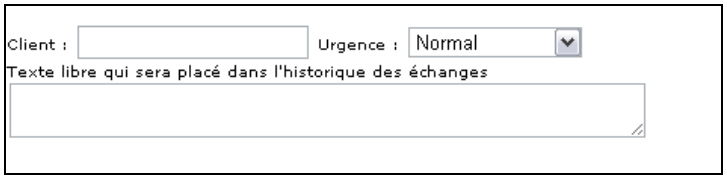
Couche			Moyens de sécuriser
10	*VIP*	*Souhaitent faire comme les autres avec le minimum de sécurité. ihone, TSE …*	*Pas de sécurité automatisée à part : sonde d'intrusion (IDS), antivirus, NAC …*
9	*Techniciens*	*Logiciels installés inutilement, codes secrets affichés, spyware …*	
8	*Utilisateur*	*Actions sur un mail, laisser entrer un pirate, faire un chèque à un inconnu …*	
7	Application	ex. HTTP, HTTPS, SMTP, SNMP, FTP, Telnet, NFS	
6	Présentation	ex. XDR, ASN.1, SMB, AFP, ZIP, RAR, ASCII	
5	Session	ex. ISO 8327 / CCITT X.225, RPC, Netbios, ASP	Mots de passe / Cookies / IDSession … Antivirus (?!)
4	Transport	ex. TCP, UDP, RTP, SPX, ATP	Firewall / DMZ
3	Réseau	ex. IP (IPv4 ou IPv6), ICMP, IGMP, X.25, CLNP, ARP, OSPF, RIP, IPX, DDP	Routage / VLAN / classes IP
2	Liaison	ex. Ethernet 802, Token Ring, PPP, HDLC, Frame relay, RNIS (ISDN), ATM, Wi-Fi, Bluetooth, ZigBee, irDA (Infrared Data Association)	Switchs et commutation (ponts)
1	Physique	ex. techniques de codage du signal (électronique, radio, laser, …) pour la transmission des informations sur les réseaux physiques (réseaux filaires, optiques, radioélectriques …)	Sécurité des locaux - câbles / appareils …

a. Pourquoi sécuriser les applications WEB

Aujourd'hui beaucoup de pirates s'attaques aux sites WEB puisqu'ils sont libres 24h/24. Avec un maximum de failles !

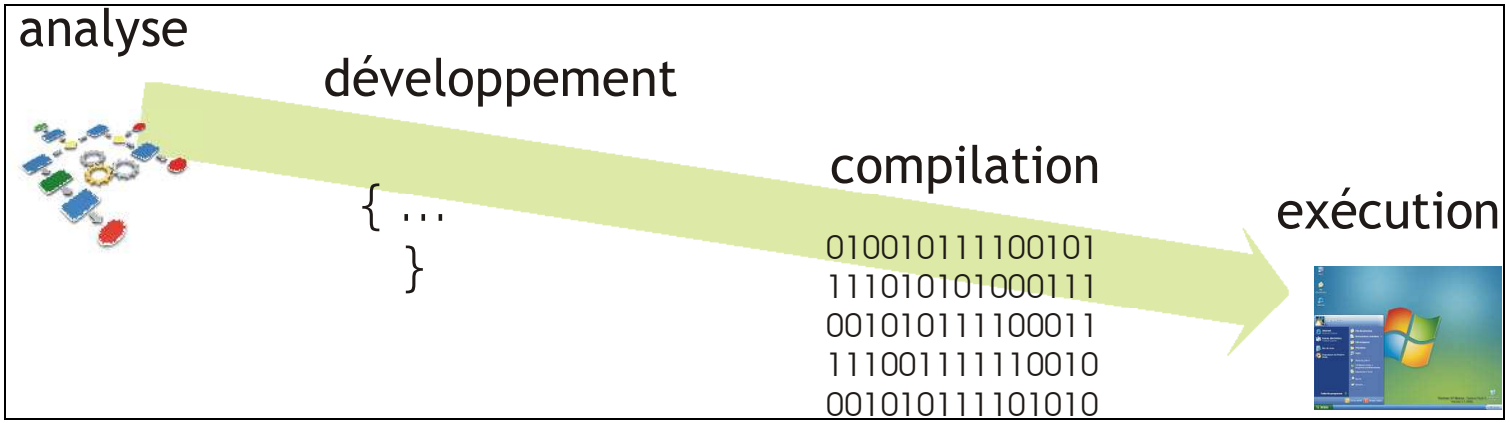
http://www.crazyws.fr/securite/plus-de-10-outils-pour-tester-la-securite-de-votre-site-web-IO8AX.html

Les attaques de sites passent par des procédés simples et dangereux :

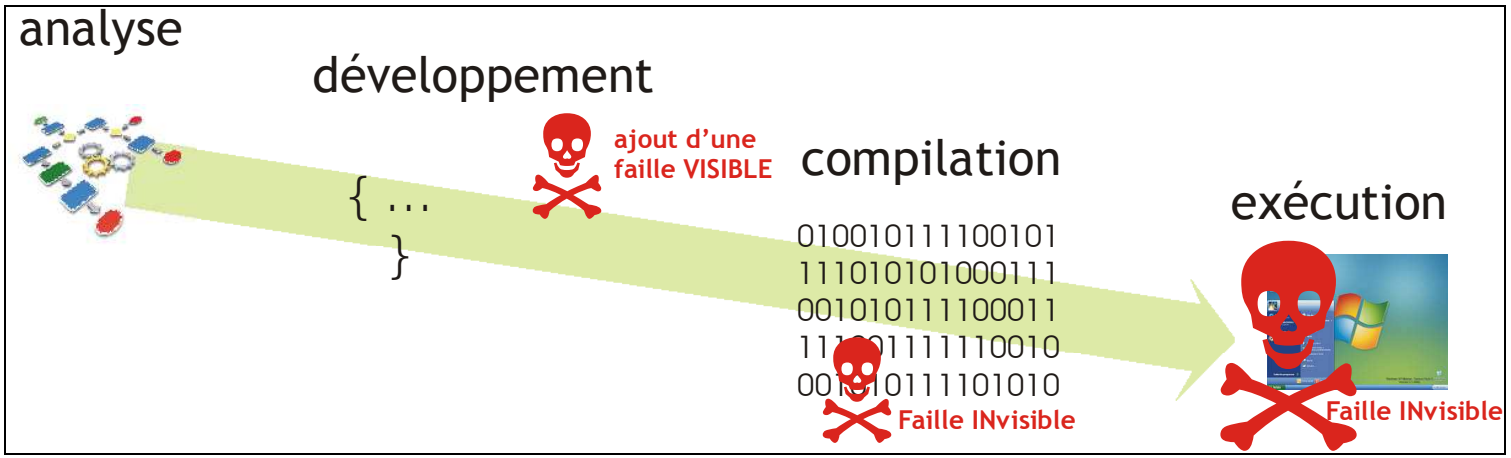
 solution possible : enlever les caractères « (» tout simplement	Puisque la plupart des sites font une exécution du type : « select * from … where client = … » En plaçant « (drop table … ;) Ca provoque l'exécution de la requête dans la requête.
Solution possible : enlever les caractères « < »	Des pirates placent du texte dans les zones : « <a hreh= > » transmettant des liens vers sites pirates pour les destinataires du mail – pour qu'ils cliquent dessus
Solution possible : ajouter une champ invisible – car, s'il est rempli, c'est qu'il s'agit d'un Spammeur	Pour éviter que les ROBOTS internet n'envoient pas de Spam
Solution possible : ajouter un lien invisible. Donc, si l'on passe dessus, c'est qu'il s'agit d'un aspirateur	Repérer les aspirateurs des sites internet
Les programmeurs d'application internet doivent faire attention à leurs codes JAVASCRIPT qui pourraient lancer des commandes directes. Il ne doit servir qu'à l'affichage ! Même remarque pour JSON / AJAX …	Interdire l'injection de code au travers de la lecture du code HTML

b. Pour les applications compilées

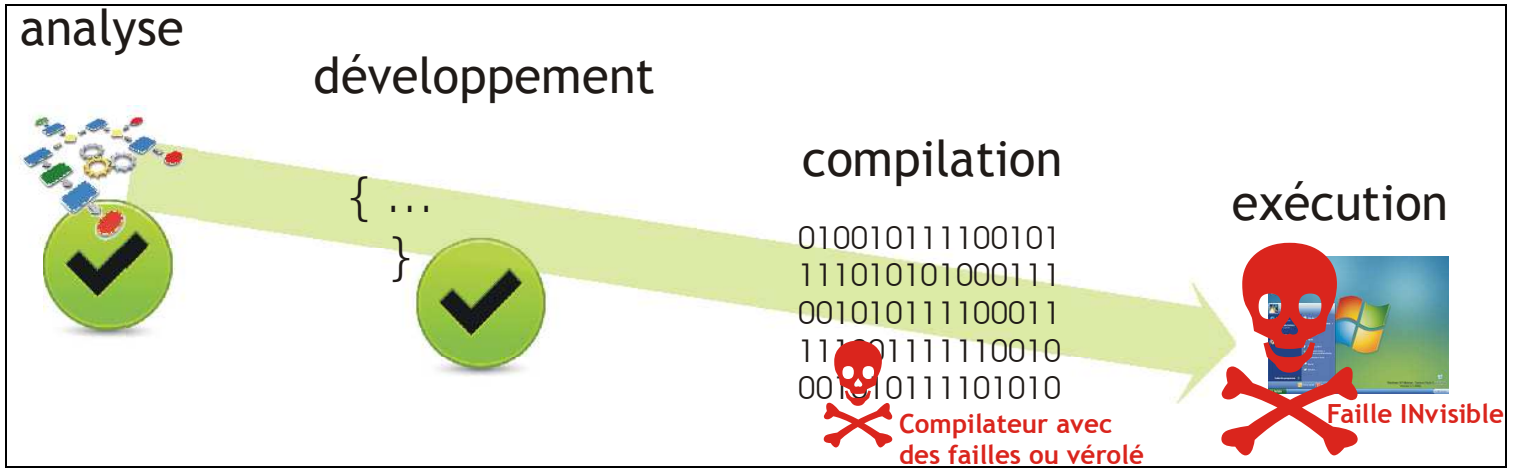
Normalement, les analystes pensent à tout, les développeurs font ce qui leur est demandé sans ajouter de Bug … Dans ce monde parfait, la compilation n'aurait pas de faille non-plus et l'exécution serait réalisée sans anomalies ! ???
Mais, tout ça : sur quelle planète ?



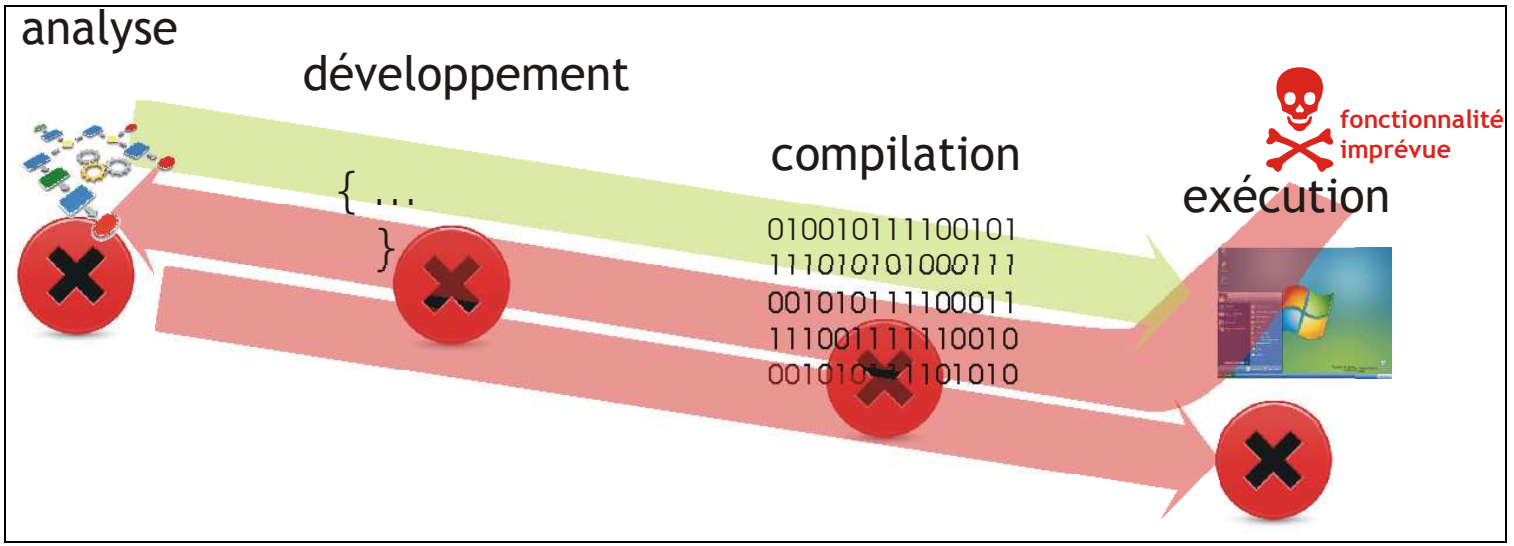
Dans la réalité, les développeurs ajoutent souvent des zones de tests pour le développement. Zones qui restent - c'est ce qui est appelé : la couche 9 ! Dans un programme compilé, la faille devient invisible dans presque tous les cas !



Il arrive même d'avoir des soucis avec le compilateur qui ajoute des failles ou qui a des faiblesses

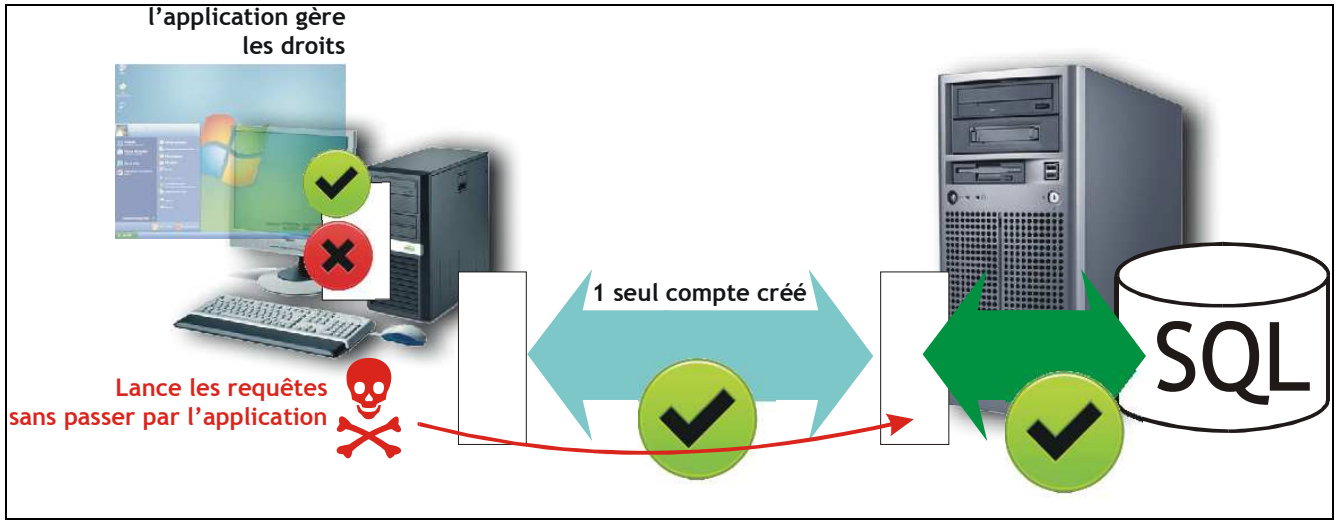


Il est même possible d'être vulnérable par une utilisation imprévue de l'application !

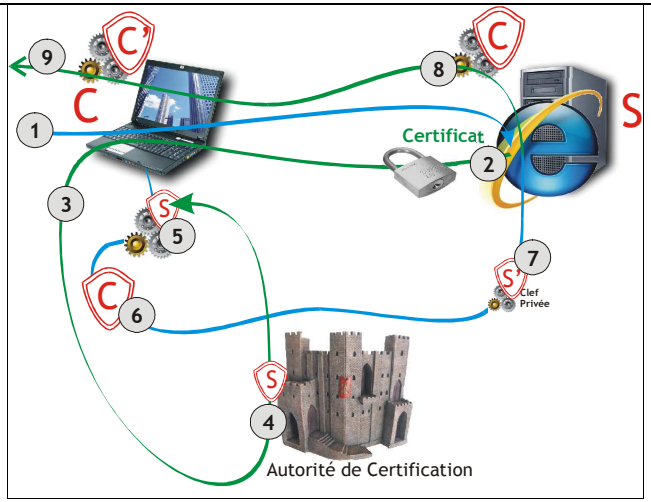


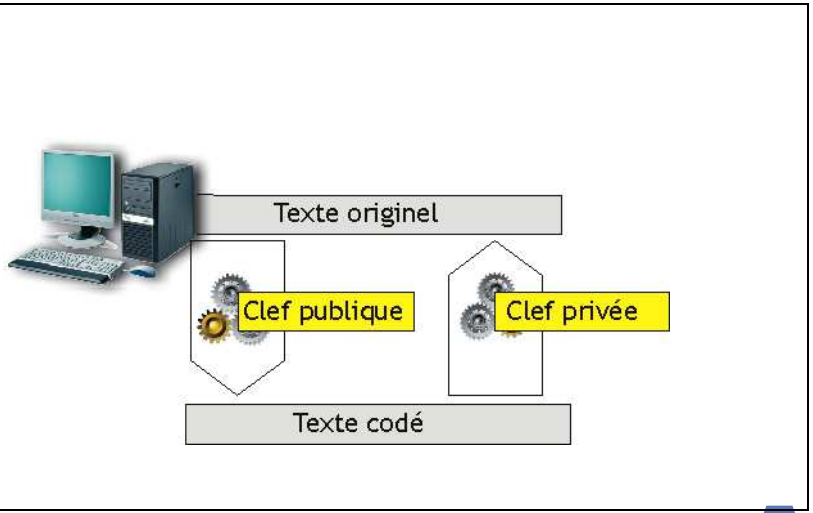
c. Pourquoi la sécurité applicative au niveau des données gérées ?

Avant tout, parce que beaucoup d'application gère les droits sur les données par l'application elle-même.



Dans le risque de "données" se trouvent aussi les paramètres applicatifs - tout ce qui est "variable" … certificats …





d. Niveaux de sécurité exigés

Nous retrouvons le modèle ISO qui représente bien les actions suivant le niveau souhaité - mais, cette fois-ci le modèle se lit de haut-en-bas.

Couche		
10	*VIP*	*Niveau déclaré sur les publicités - vue Grand-Public (et enfance)*
9	*Techniciens*	*Validation de ce qui a été programmé par nos acteurs - règle de sécurité*
8	*Utilisateur*	*S'assurer que les usagers peuvent s'en servir sans se mettre en danger*
7	Application	L'application rend servir et répond aux exigeances CNIL / HADOPI / NSA…
6	Présentation	Choix des niveaux de chiffrement ou de codage
5	Session	Définition des règles de mots de passe et authentification
4	Transport	Définition des services utilisés et co-habitation applicative
3	Réseau	Application est routable et connectable à Internet
2	Liaison	Risque sur les protocoles réseaux et séparation des flux
1	Physique	Qu'en est-il pour la sécurité des câbles / chassis …

La réglementation ou la certification nous oblige éventuellement à un niveau de sécurité. Malheureusement, ce niveau est rarement le niveau réellement suffisant. Il s'agit d'un niveau minimum pour être certifié.

2. Ce qui pourrait donner un sentiment de sécurité

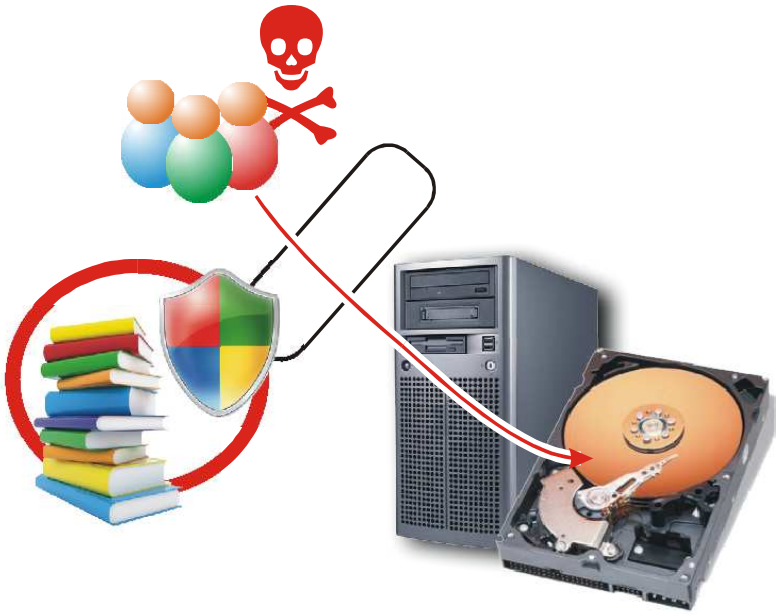
a. Les outils de sécurité systèmes et leurs périmètres (Firewall …)

Couche			OUTILS
10	*VIP*	*ihone, TSE …*	
9	*Techniciens*	*Logiciels installés inutilement, codes secrets affichés, spyware …*	
8	*Utilisateur*	*Actions sur un mail, laisser entrer un pirate, faire un chèque à un inconnu …*	
7	Application	Logiciels	Antivirus
6	Présentation	Correction des CR/LF	Anti Malware
5	Session	Netbios	Proxy
4	Transport	TCP/UDP	Firewall
3	Réseau	IP	Routeur / NAT
2	Liaison	Ethernet	switch
1	Physique	Câbles / ondes	

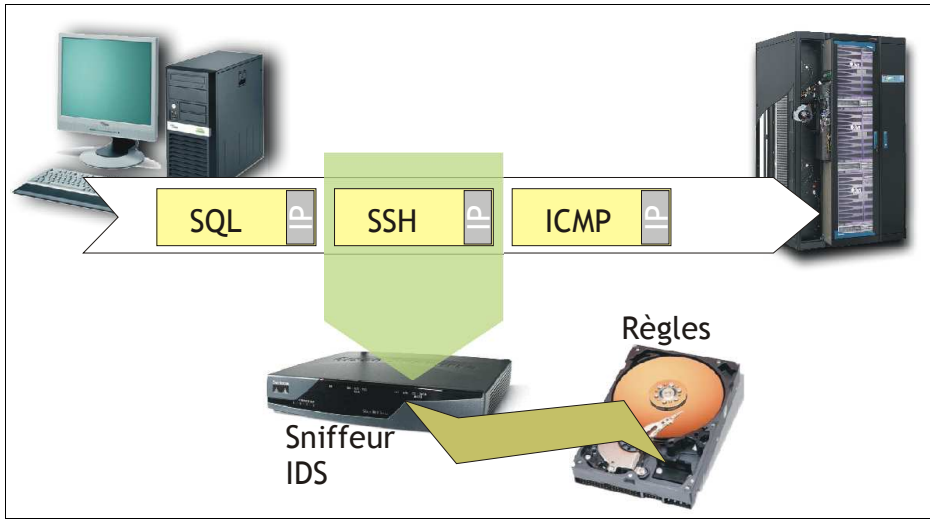
b. Les antivirus

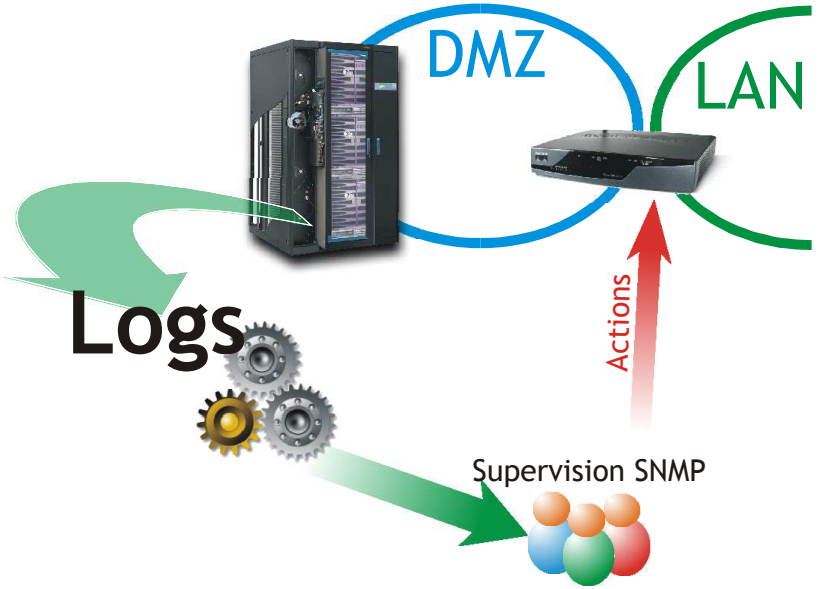
Les antivirus fonctionnent généralement par reconnaissance de signature.
En plaçant la chaine de caractères suivante dans un fichier "notepad" puis en mettant l'extension ".com" au fichier, il sera reconnu comme un virus … et ce n'est pas le cas !
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*

Une autre méthode est celle appelée : heuristique. Par reconnaissance de comportement. Certains fichiers systèmes ouverts, ou des accès au registres …
Les antivirus nécessitent forcément des mises à jours car ils ont besoin de connaître les nouvelles formes d'attaque.
Ils nécessitent aussi de scanner l'ordinateur régulièrement car il a pu laisser passer des virus inconnus…



c. Les statistiques de connexion





3. Définition des risques applicatifs

a. risques d'intrusion

Très souvent, nous pensons à la sécurité d'intrusion contre la récupération d'informations … mais ce n'est pas le seul cas !

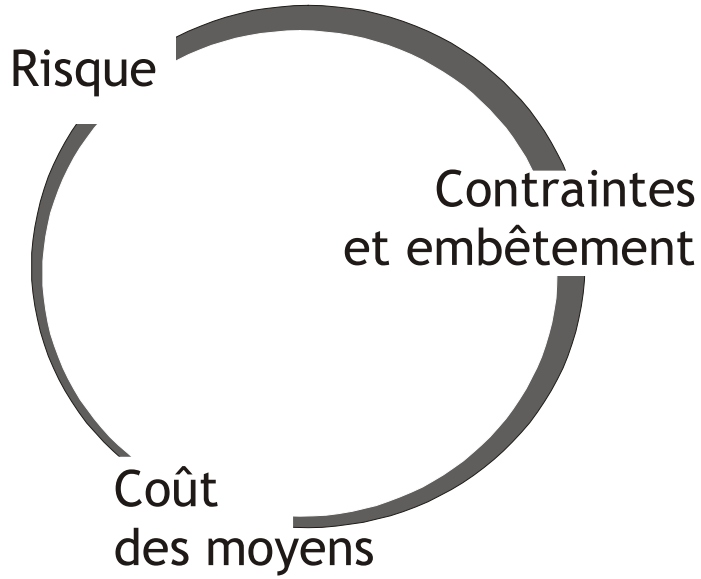
- Le premier est la non diffusion d'information
  - Protéger l'information contre la « copie »
  - Protéger l'information contre l' « écoute »
  - S'assurer qu'un support à détruire est réelle vide
  - S'assurer qu'un support d'information ne « voyage »

- La protection de l'information (contre sa disparition)
  - S'assurer que l'information ne soit pas supprimée ou détériorée
  - S'assurer que l'information originelle ne soit pas dissimulée par une fausse

- Le contrôle par rapport à l'usager
  - S'assurer que « seuls » les personnes autorisées ont accès aux informations
  - Contrôler les accès et s'assurer que les méthodes de contrôle sont fiables

- La notification d'incident
  - Avant ou après les alertes ou les incidents, il est nécessaire de notifier les soucis rencontrés, les moyens mis en place pour le résoudre et notifier les ressources préventives à mettre en place.

Trouvons l'équilibre entre les 3 points de la sécurité

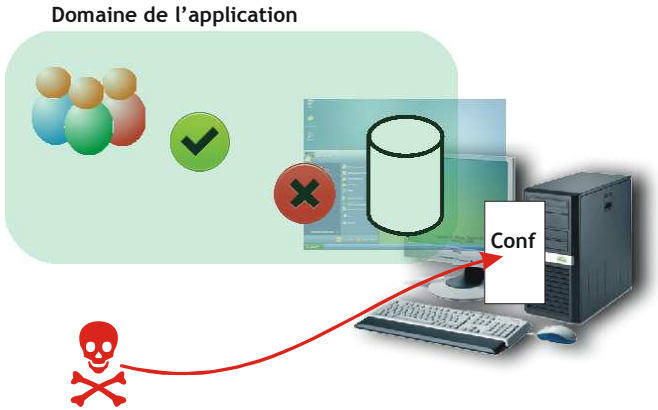


Risque : de perte / destruction / divulgation / altération / inaccessibilité …

Contraintes : les moyens mis en place auront-ils de pratiques qui gêneront les usagers ? Si oui , ils contourneront les moyens mis en place !

Coût : nous pouvons toujours TOUT mettre en place. Pourtant, le coût est souvent le phénomène bloquant. Coût financier / Ressources / Moyens humains …

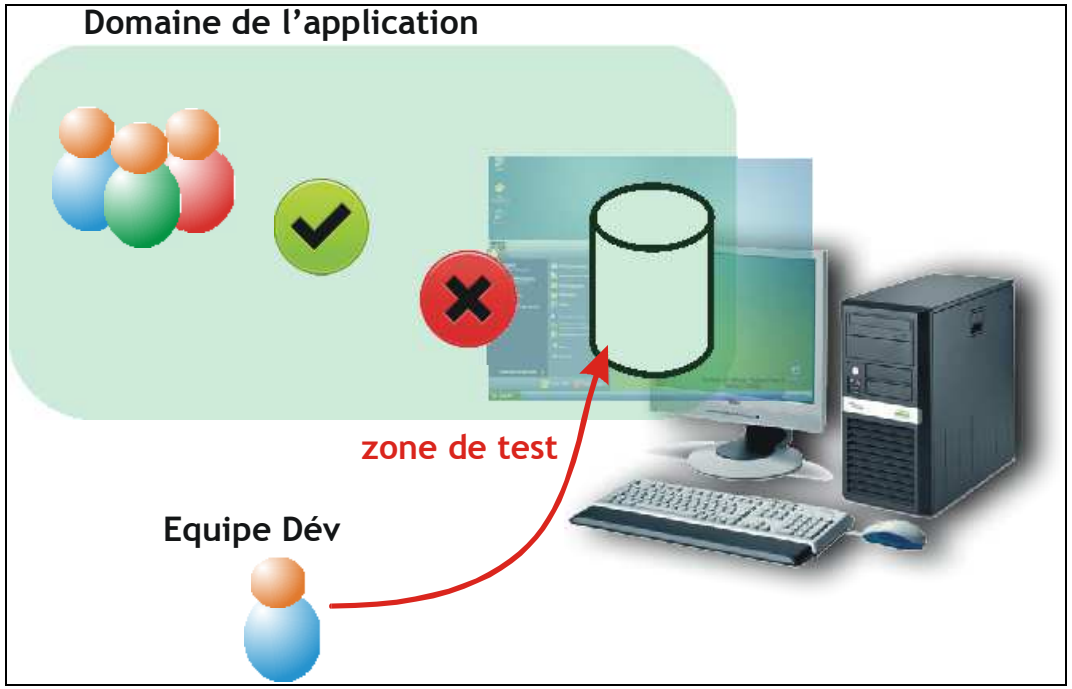
b. risques de manipulation



La sécurité est souvent présentée comme "observant" le domaine de l'application réalisée. Mais le risque se trouve aussi sur les configurations, sur le système. Ainsi, le pirate peut manipuler les données, les traitements sans se faire repérer.

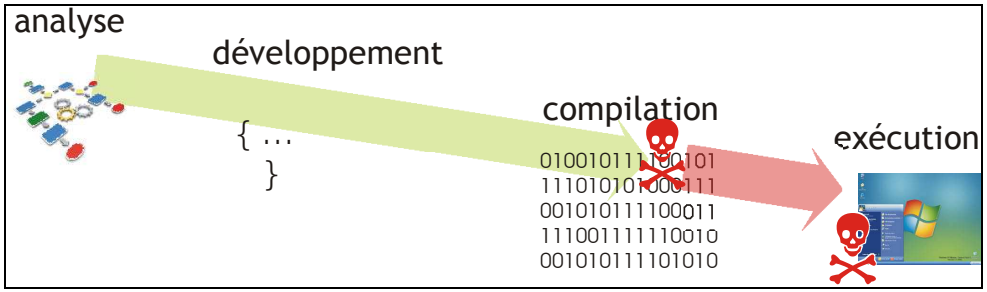
c. risques internes

Nous avons déjà parlé de la "couche 9". Risque de zones laissées ouvertes, connexions possibles, YES codes.



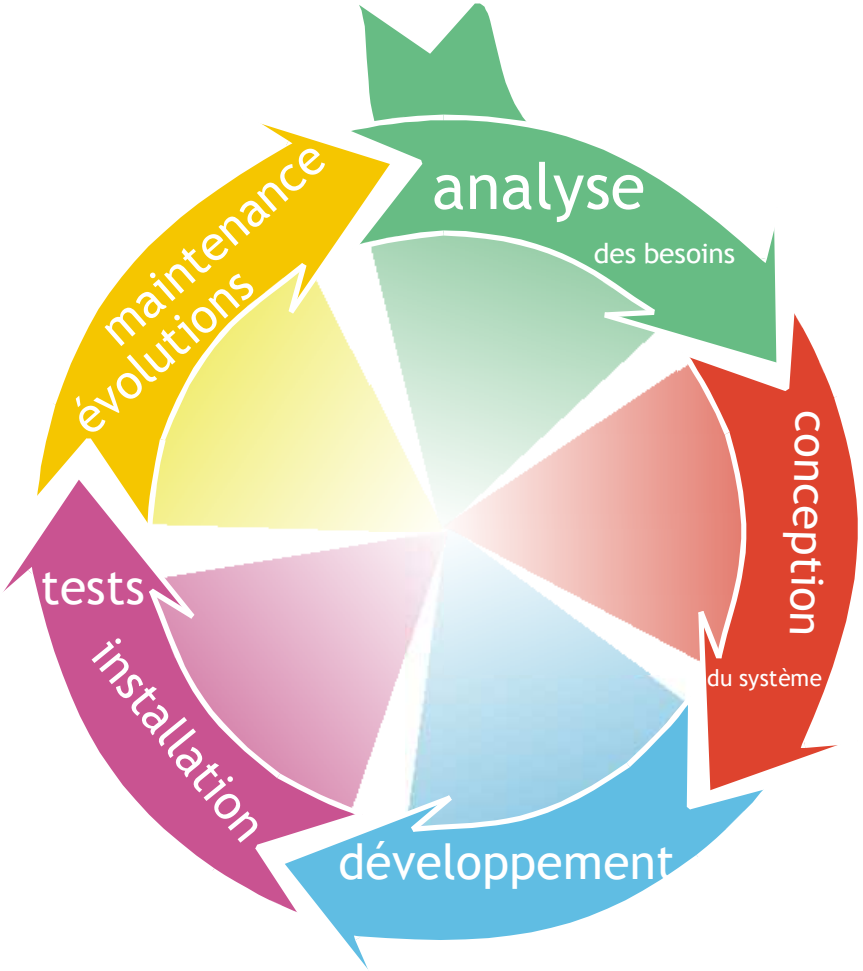
d. risques de compilateurs espions

Il s'agit d'une contamination du logiciel de compilation sur le PC du développeur. Ainsi, tous ses réalisations sortent avec un virus "maison" intégré dans l'application !



4. SDLC (secure development life-cycle)

Qu'est-ce que l'SDLC



Plusieurs phases font parti du SDLC

1. analyse des besoins et cadrage - analyse initiale
2. conception du système
   - conception système + base de données
   - intégration des processus métiers + règles de gestion
   - choix plateforme de déploiement
   - choix des librairies utilisables (et certifiées)
   - choix interface IHM
   - ouverture pour réutilisation du modèle
3. Développement
   - développement + intégration des processus métiers
   - utilisation des librairies
   - suivi des tâches
4. tests et implémentation
   - tests et recettage avec les usagers
   - tests de performances / déploiement
   - tests et traçage client/serveur
5. Maintenance pertinente et améliorations
   - maintenance système - maj sécurités
   - amélioration à prévoir sur version n+1

a. analyse du besoin et planification

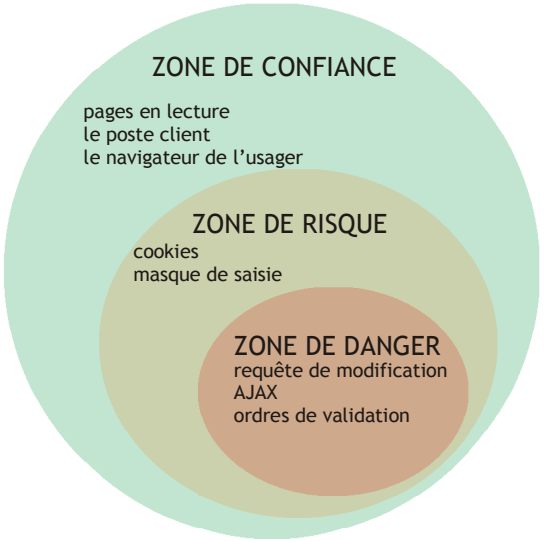
Durant cette phase, nous déterminons les besoins … comme toute application ! Mais, ici, nous ne parlons pas de "Cycle de vie d'un développement informatique" mais "de SECURITE dans le cadre d'un cycle de vie d'un développement".
Donc, dans cette phase apparaît d'autres points :

- nommer un "monsieur sécurité" concerné par le projet - c'est lui (elle) qui fera la veille technologique pour que le projet soit sécurisé jusqu'au niveau demandé.
- Voir les plans et recommandations liés au projet (ou la fonction demandée)
- Définition des objectifs sécurité à mettre en place - voire même prévoir les versions n+1 … n+x
- Définition des obligations réglementaires et respect des publicités annoncées

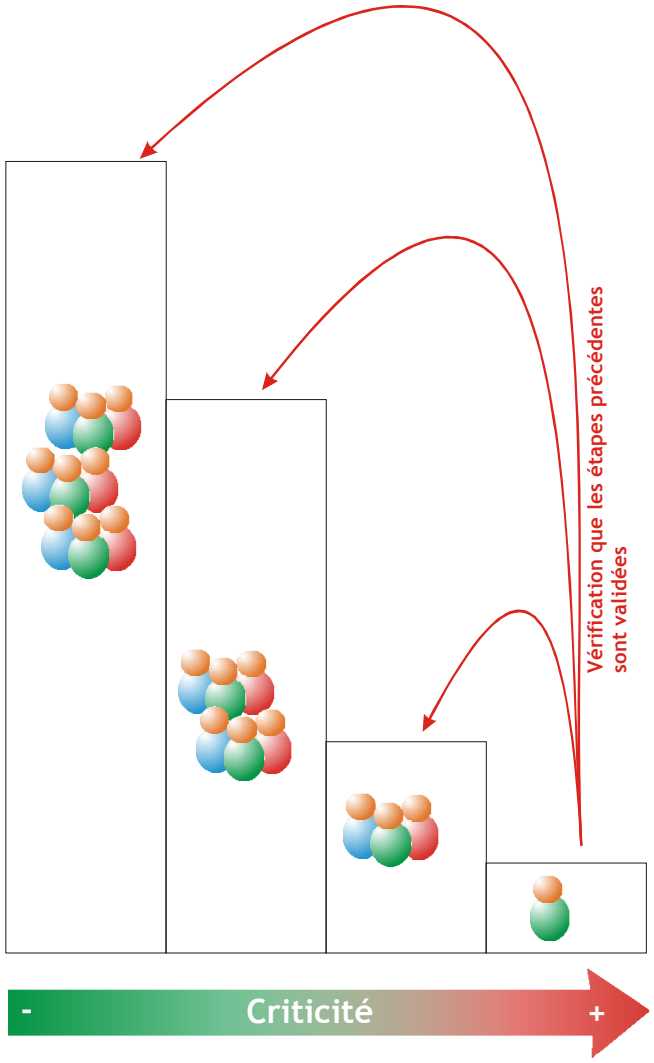
b. Conception

Généralement, dans cette phase, la sécurité s'arrête aux "droits des usagers" !

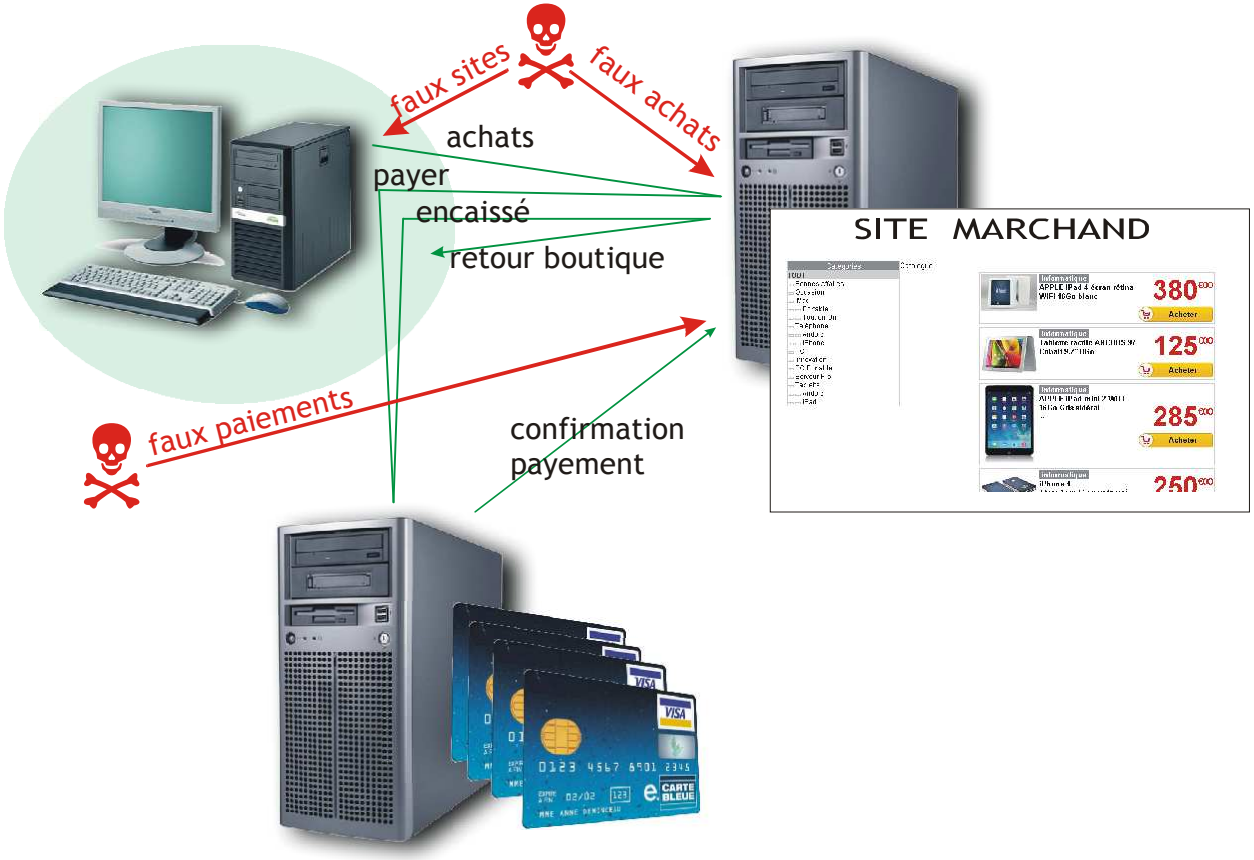
- Définir l'architecture de sécurité - définition des zones de confiance, des zones de danger et des zones à risque. Attention, les périmètres sont directement liés à l'Entreprise, la réglementation, les obligations, l'application et de mes risques.



- Définition des fonctions attaquables ou vulnérables. Et définition de l'architecture empirique des authentifications. La fonction n+1 (avec pluS d'autorisations) ne sera activable QUE si la fonction 'n' a été vérifiée. Et, à l'étape 'n+1' mettre en place le moyen de revérifier que les paliers 'n' … 'n-1' … 'n-2' ont été validés.

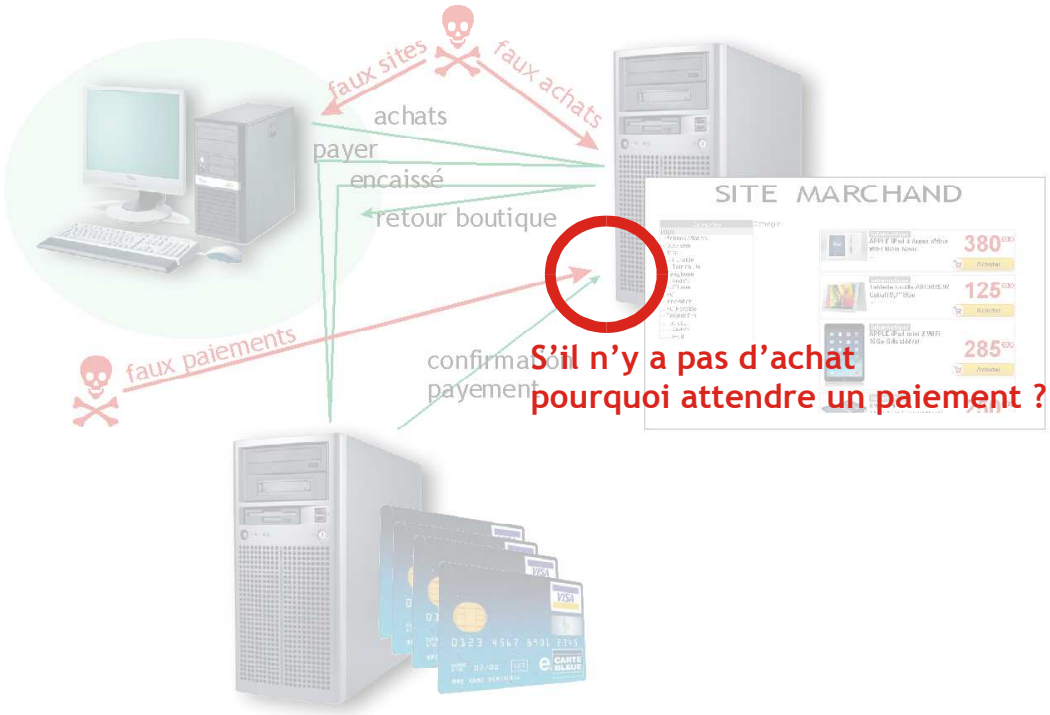


- Modélisation des menaces. Définitions des axes à risque et à danger. Les développeurs pourront anticiper les moyens de sécurité.



- Définir un niveau de log et le niveau de sécurité à donner aux logs. Ils seront nécessaires en cas d'attaques ou dysfonctionnement.

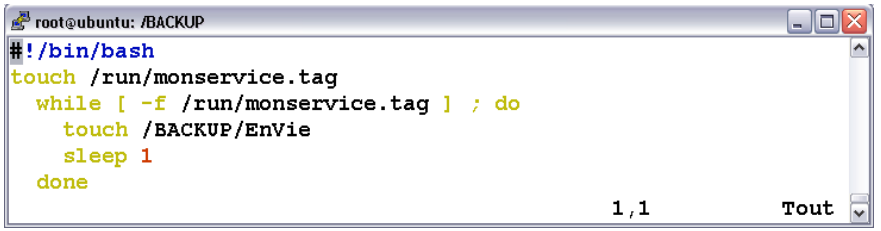
- Définition des "fenêtres" de validité. Espace-temps durant lequel la fonctionnalité est sécurisé et espace-temps durant lequel le service est inexistant. A-t-on besoin d'un service à tout instant ? Définissons les espaces-temps.



- Définition des zones de test réclamées par les développeurs. **Ces zones de test sont OBLIGATOIRES** ! Sinon, les développeurs les créeront et oublieront peut-être de les fermer ! Sans compter sur leur imagination pour les "yes codes", ports réseaux ouverts, URL administrateur, fonctions de reboot ou DoS (Dénie de service) …
- C'est dans cette phase que l'on détermine les niveaux de sécurité couche 1 des sources et de leurs compilations. Il est défini si la procédure de paiement codée en C++ sera stockée et sauvegardée avec un niveau de cryptage. Seuls les exécutables seraient diffusés … par exemple …
- C'est dans cette phase que l'on définit le niveau de documentation des sources émis - exemple CMMI

Niveau 1 : CMMI Initial

Le niveau le plus bas montre que l'organisation n'est pas prête, et le projet pas stable. Ce dernier dépend d'une poignée de personnes, qui ne font pas appel à des processus éprouvés. Il se peut cependant que le projet aboutisse, mais en dépassant certainement le budget et le temps alloués. Le projet ne construit pas sur les succès passés.

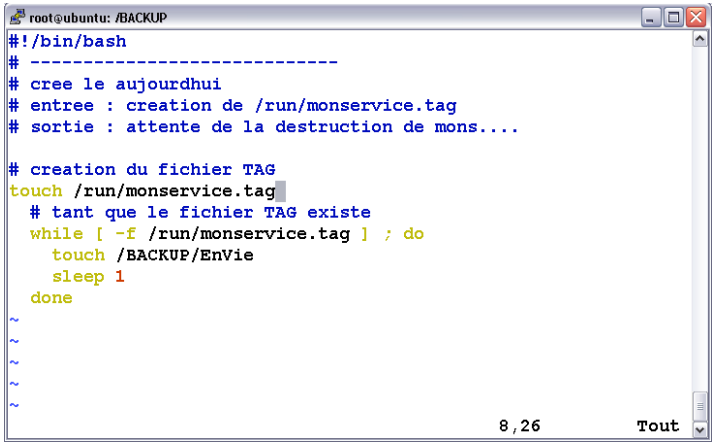


Niveau 2 : Reproductible

Le projet construit sur ce qui a été appris précédemment, en faisant appel à une certaine discipline et à une gestion de projet basique. De fait, le projet est géré selon les plans, avec étapes-clefs et vérification des coûts et des fonctionnalités. En sources, nous ajoutons un descriptif au début

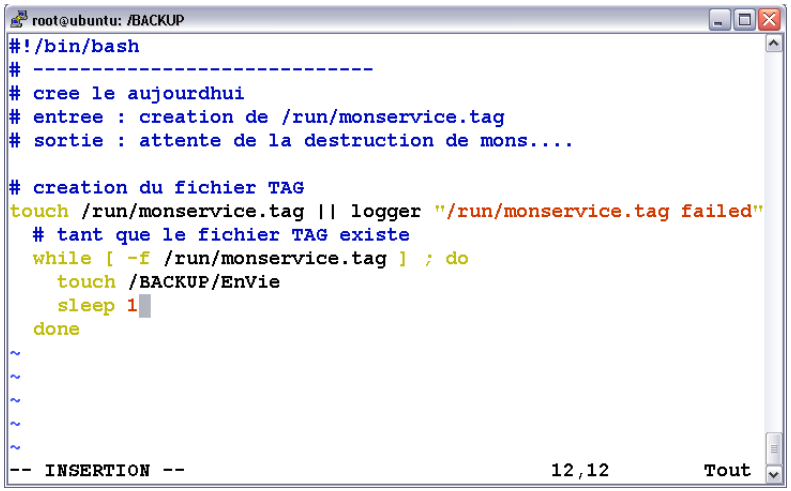
Niveau 3 : Défini

Ce n'est plus le projet qui dispose d'une bonne discipline, mais l'ensemble de l'organisation, de manière cohérente. Tous les projets s'en trouvent améliorés.



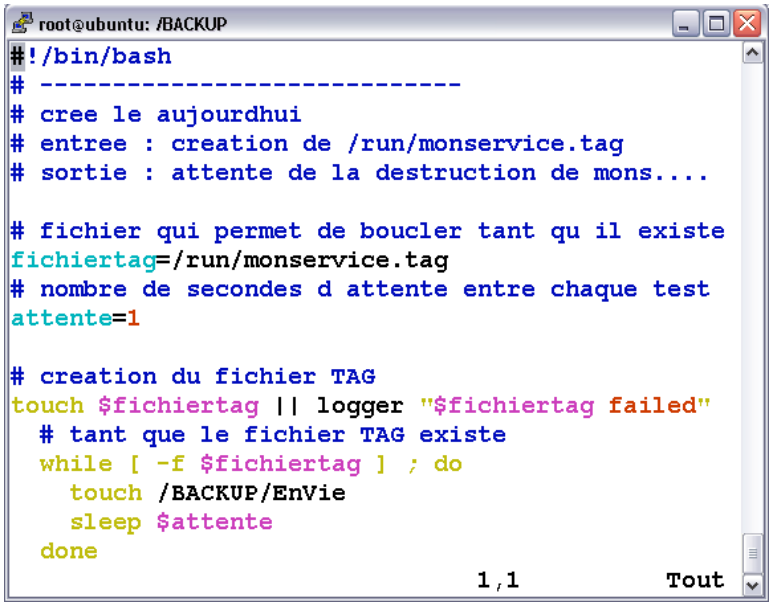
Niveau 4 : Maîtrisé

Les efforts de mesure et de gestion autorisent un contrôle sans effort du développement, avec capacité d'ajuster et adapter des projets précis sans troubler les autres. Les performances des processus sont prévisibles en quantité comme en qualité.



Niveau 5 : optimisation

Les processus sont constamment améliorés de manière incrémentale et innovante. Les objectifs sont revus en permanence pour rester proches des besoins du marché. Les évolutions sont anticipées et gérées de bout en bout.



c. implémentation

il s'agit de la phase : code / test / intégration
Les éléments du SDL qui s'appliquent à la phase d'implémentation sont les suivants :

- Application des normes de codage et de test. Les normes de codage aident les développeurs à éviter d'introduire des défauts pouvant se traduire par des failles de sécurité. Par exemple, l'utilisation de fonctions de gestion de chaînes et de gestion des tampons plus cohérentes et plus sûres évite d'introduire des failles de saturation de tampon. Les normes et les recommandations en matière de test permettent de s'assurer que les **tests sont axés sur la détection des failles** de sécurité potentielles et ne se limitent pas au bon fonctionnement du logiciel.

- Application d'outils de test de la sécurité y compris des outils aléatoires. Ces outils aléatoires fournissent des entrées structurées mais invalides aux interfaces de programmation d'applications (API) et aux interfaces réseau afin de maximiser la probabilité de détecter des erreurs pouvant se traduire par des failles du logiciel.

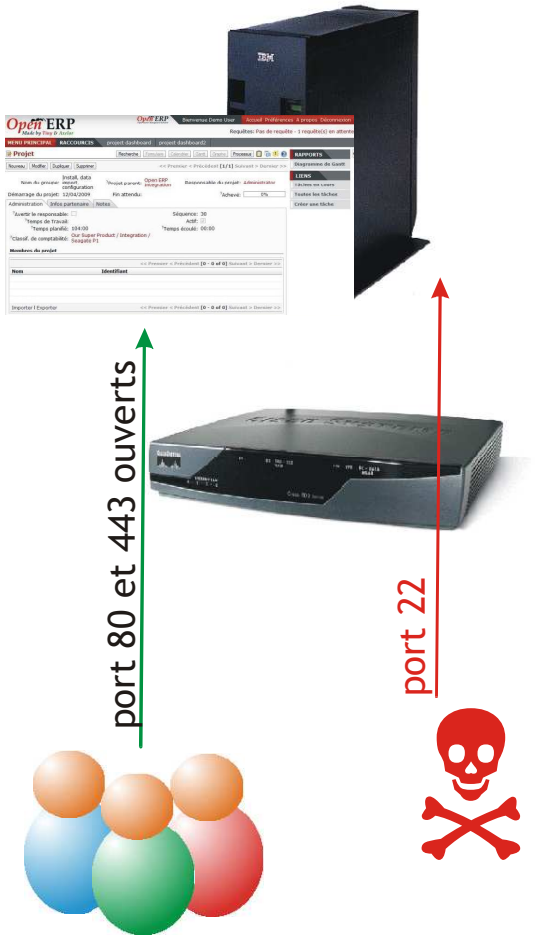
- Application d'outils d'analyse de code statique. Les outils peuvent détecter certains types de défauts qui se traduisent par des failles, tels que les saturations de tampon, les dépassements sur les entiers et les variables non initialisées. Microsoft a investi massivement majeur dans le développement de ces outils (lPREfix et PREfast sont les deux outils utilisés depuis le plus longtemps) et poursuit leur amélioration à mesure que l'on découvre de nouveaux défauts de codage et failles de logiciels.

- Réalisation de révisions du code. Les révisions du code complètent les outils automatisés et les tests en dirigeant les efforts des développeurs formés vers la révision du code source, ainsi que la détection et la suppression des failles de sécurité potentielles. Elles constituent une étape essentielle du processus de suppression des failles de sécurité des logiciels au cours du processus de développement.

- S'assurer que les zones de test sont utilisées et refermables le temps voulu.

d. méthodes de test et recette (vérification)

Très souvent, la DSI demande aux usagers de faire des tests (recettes) pour s'assurer que ce qui a été produit répond aux besoins. Toutefois, il est rarement fait le test : "est-ce que l'application ne répond pas TROP aux besoins". C'est à dire que le niveau de sécurité permet de faire bien plus de chose que prévu.



Peut-on penser à tout ??

Méthode pratique et efficace :

1. observation / mesure / alerte
2. action de protection
3. audit régulier de la protection
4. auditer les effets de bord : auditer ce que nous n'avons pas protégé.

Il est souvent conseillé de faire les tests utilisateurs en même temps que les tests sécurités.
Il est souvent demandé aussi aux usagers de tester ce qui leur semble interdit - au niveau fonctionnel.

Peut-on prévoir des "pot de miel" ?
Il s'agit de préparer un piège afin de détecter une tentative d'intrusion. Par exemple, le nom et mot de passe "admin" / "admin" déclenche un blocage de l'émetteur, qui est , de toute évidence, un pirate ou en "force brute".

e. Diffusion et documentation

Microsoft a découpé cette phase en 2 : Phase de diffusion (correction des bugs de sécurité) et Phase dépannage + support

- Il est nécessaire de prévoir une révision Finale de Sécurité (FSR). Et d'indiquer ce qui a été fait.
- Définir les documentations avec leur niveau de diffusion :
    - Diffusion aux usagers (rassurer les usages) ex: "vous entrez dans une zone sécurisée"
    - Diffusion aux clients (diffusion commercial) ex: "vos données sont chiffrées" - "certifié T.R.U.C."
    - Diffusion aux développeurs (technique) ex: "les forces brutes sont repérées"
    - Diffusion aux agents RSSI (sécurité) ex: "en cas d'intrusion, nous sommes prévenus"
    - Diffusion à la Direction (secret de fabrication) ex: "les sources sont dans le local A code 1234"
- C'est dans cette phase qu'il nous faut prévoir du temps pour les analyses post-mortem d'attaques abouties ou pas.
- Il s'agit d'une phase veille technologique.

f. déploiement et mise en production

Le déploiement (ou mise en Prod) est souvent fait par étapes afin de ne pas rencontrer de dysfonctionnement. Toute l'énergie est souvent mise sur le "il faut que ça marche" et il est souvent oublié "la sécurité".

Voici donc des phases de déploiements sécurisés :

- Avant toute forme de déploiement, il faut arrêter les formes de logs verbeux, les "failles pour les tests", s'assurer que le niveau de sécurité est toujours le même.
- S'assurer qu'il n'y a pas de code-source ou documentation de développement embarqués avec le déploiement. 99% des attaques passent par ce moyen-là.
- Vérifier que les commentaires applicatifs ne soient pas visibles des usagers
- Vérifier le versionning et le fonctionnement de patchs de sécurité
- S'assurer que le recettage fonctionne encore

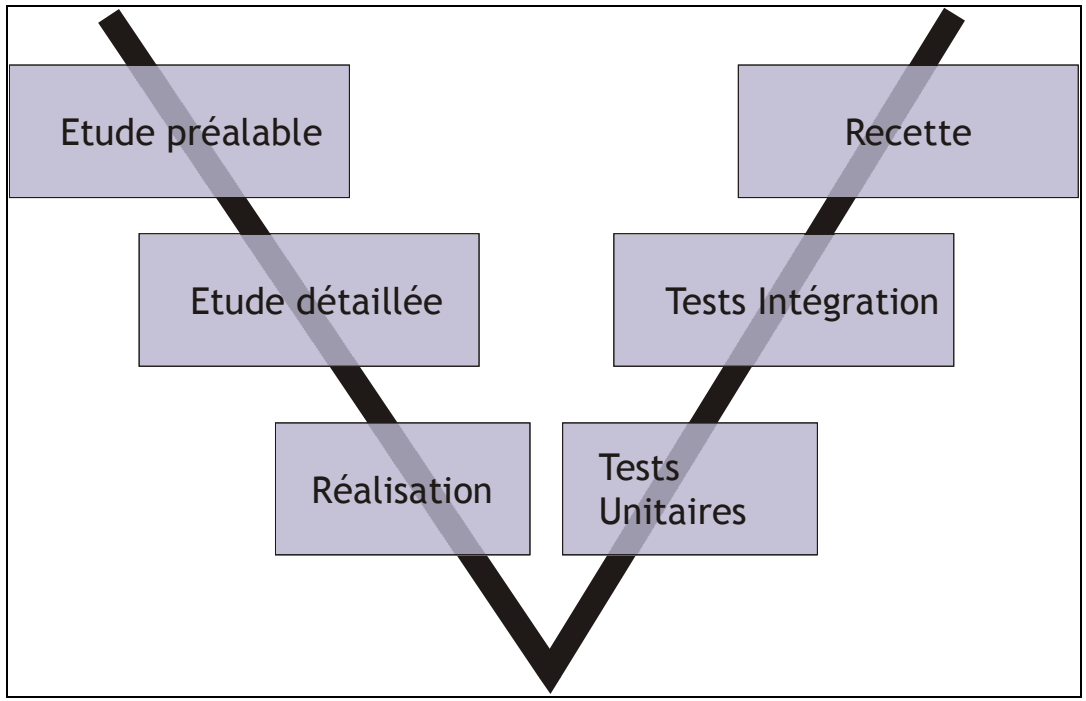
g. solutions de maintenance et Risques

Les solutions de maintenance ne sont pas seulement celles de déploiements automatiques. La maintenance, est une chose plus ouverte en sécurité :

- formation continuelle sur les risques et les moyens utilisés par les pirates
- mises à jour des risques sur tous les points de vues : risque information / risque à la diffusion d'information individuelle / risque financier / risques matériels / risque de pérennité ou de perte de confiance des usagers…

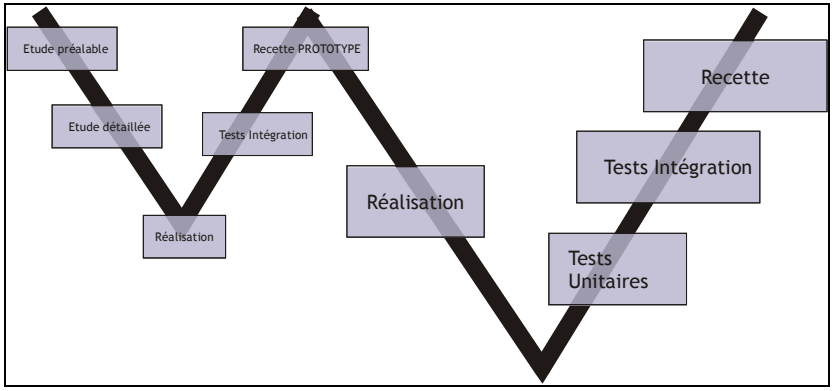
h. différentes méthodes (agile, V, W …)

Les cycles linéaires



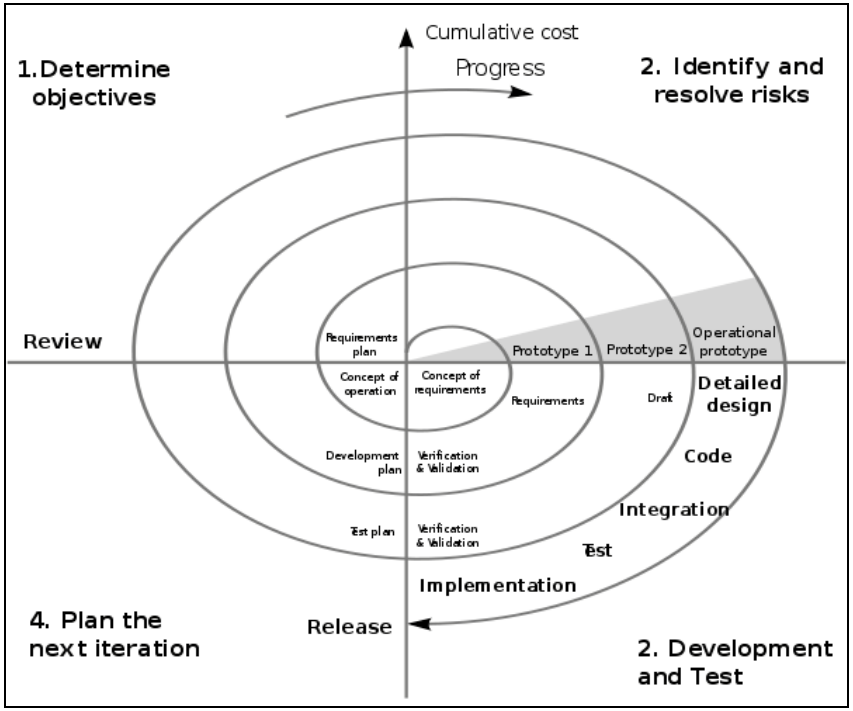
Cycle en V

Adapté pour des projets qui présentent des fonctionnalités dites « habituelles ».
Développement maîtrisé d'applications.



Cycle en W

Adapté pour des projets qui nécessitent des prototypes.



Cycle en spirale

Il s'agit d'une méthode qui permet de valider chaque étape par un prototype avant de la développer réellement.
Ainsi, beaucoup moins de développement est attendu et donc, beaucoup moins de BUG !

L'usager sait, avant le développement, ce qu'il aura - mais il sait aussi qu'il l'aura pas maintenant dans sa conception finale.
Il faut que le projet soit évolutible par accoues..

Dans ce but, elles prônent 4 valeurs fondamentales (entre parenthèses, les citations du manifeste)[2] :

- **L'équipe** (« Personnes et interaction plutôt que processus et outils ») : Dans l'optique agile, l'équipe est bien plus importante que les outils (structurants ou de contrôle) ou les procédures de fonctionnement. Il est préférable d'avoir une équipe soudée et qui communique, composée de développeurs (éventuellement à niveaux variables), plutôt qu'une équipe composée d'experts fonctionnant chacun de manière isolée. La communication est une notion fondamentale.
- **L'application** (« Logiciel fonctionnel plutôt que documentation complète ») : Il est vital que l'application fonctionne. Le reste, et notamment la documentation technique, est une aide précieuse mais non un but en soi. Une documentation précise est utile comme moyen de communication. La documentation représente une charge de travail importante, mais peut pourtant être néfaste si elle n'est pas à jour. Il est préférable de commenter abondamment le code lui-même, et surtout de transférer les compétences au sein de l'équipe (on en revient à l'importance de la communication).
- La **collaboration** (« Collaboration avec le client plutôt que négociation de contrat ») : Le client doit être impliqué dans le développement. On ne peut se contenter de négocier un contrat au début du projet, puis de négliger les demandes du client. Le client doit collaborer avec l'équipe et fournir un feed-back continu sur l'adaptation du logiciel à ses attentes.
- **L'acceptation du changement** (« Réagir au changement plutôt que suivre un plan ») : La planification initiale et la structure du logiciel doivent être flexibles afin de permettre l'évolution de la demande du client tout au long du projet. Les premières releases du logiciel vont souvent provoquer des demandes d'évolution.

Méthode AGILE (Adaptive)

Le principe est d'utiliser une méthode propre aux développements informatiques.

C'est une méthode qui part du principe que l'informaticien fait bien son travail et que l'usager sait bien ce qu'il veut.
Dans ce monde de « bisounours » la méthode AGILE se veut plus simple.

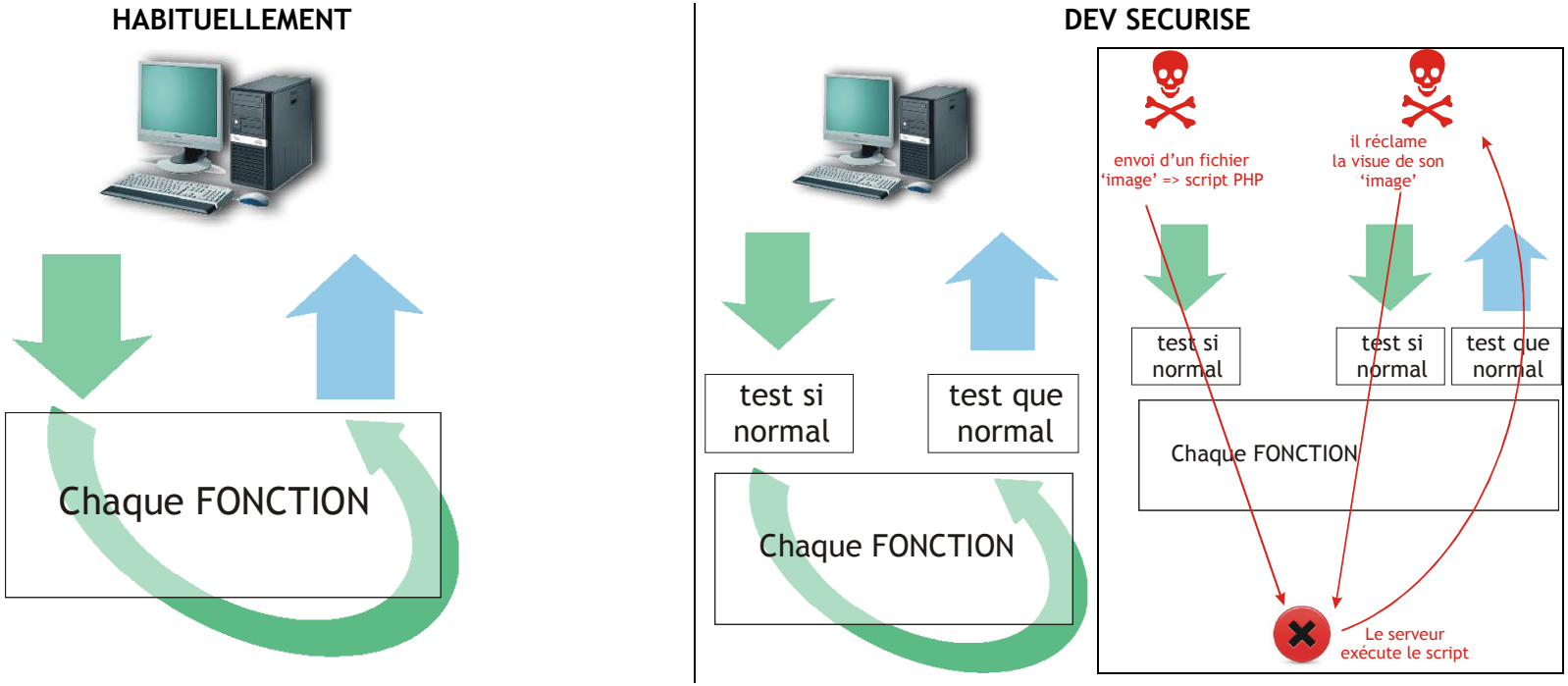
5. Audit de code

a. introduction

Un audit de code se fait en plusieurs phases, avec plusieurs compétences différentes.
L'analyse peut se faire d'une manière automatique, mais cela suppose que notre développement fasse partie des solutions connues (par exemple, l'utilisation d'un CMS pour une application WEB).

b. audit + test d'intrusion (Penetration-test)

L'Audit (contrairement à ce que ça veut dire) se fait avec les yeux. Le découpage de l'audit est assez simple au premier abord.



Chaque fonction, chaque appel, chaque entrée sortie est regardé pour voir si un test est fait avant exploitation.

Voici un exemple sur un CMS très connu :

```php
<?php
include("../configuration.inc.php");

if (!isset($_GET['rubid'])) {   $rubid = 0;} else {$rubid = $_GET['rubid'];}

$qid_r = recupere_sous_rubrique($rubid);

$DOC_TITLE = "$site";

include("$repertoire_modele/haut.php");
?>
```

```php
function recupere_sous_rubrique($rubnewsid=0) {

    $qid = mysql_query("SELECT id, nom, image, parent_id  FROM peel_rub_news WHERE
     parent_id = $rubnewsid AND id > 0 AND etat = 1 ORDER BY nom") or
     DIE('Une erreur de connexion □ la base a est produite ' . __LINE__ . '.<p>' . MYSQL_ERROR());;

        return $qid;
}
```

c. solutions et acteurs

http://www.rapid7.com/ Open Web Application Security Project

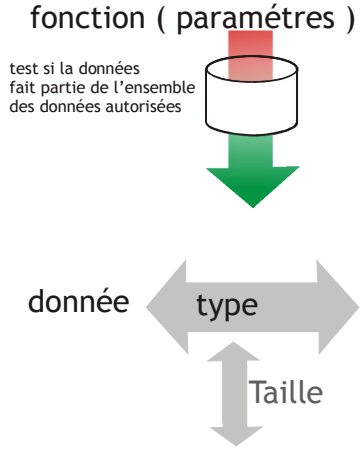
http://www.metasploit.com/ Payment Card Industry Data Security Standard (PCI DSS)

http://www.xmco.fr/ Systems Development Life Cycle (SDLC)

Automated Penetration Testing with White-Box Fuzzing (Microsoft)

d. méthodologie

l'approche de test de source se fait d'une manière itérative :



Pour chaque fonction nous regardons les types de données et nous devons faire une test de cohérence. A savoir si la donnée entre bien dans l'ensemble des données autorisées.

Chaque donnée déclarée doit avoir son spectre de résultats et une taille régulée.

S'il s'agit de dimensionnement dynamique, il faut établir un maximum afin d'éviter les débordement (fuite mémoire)

e. résultat et champs d'attaques

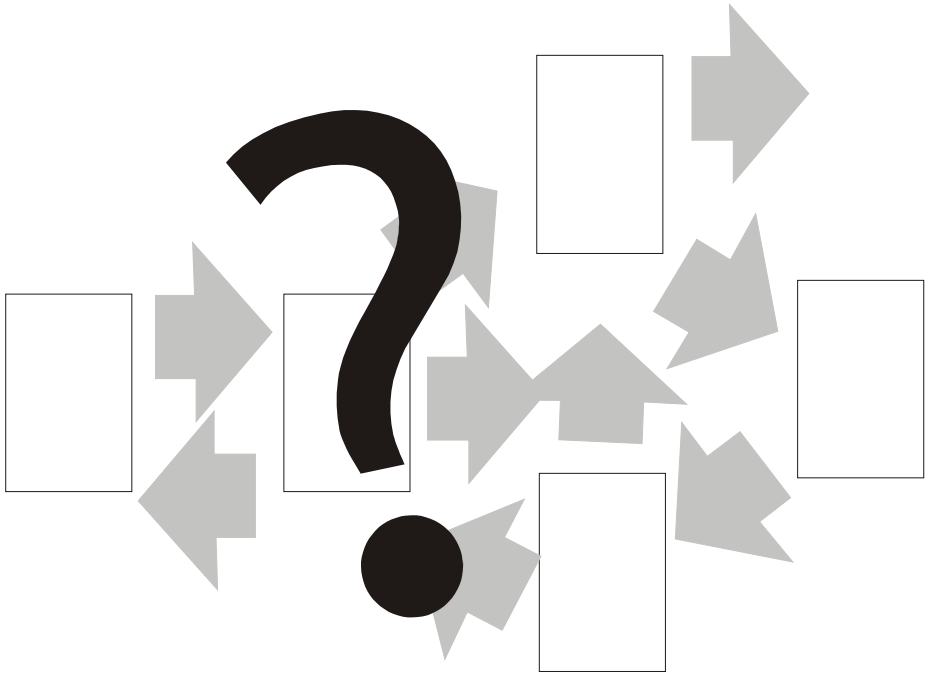
Toute fonction qui est lancée doit gérer les exceptions sur une donnée incorrect en la logguant …

6. Design Review (amélioration de conception)

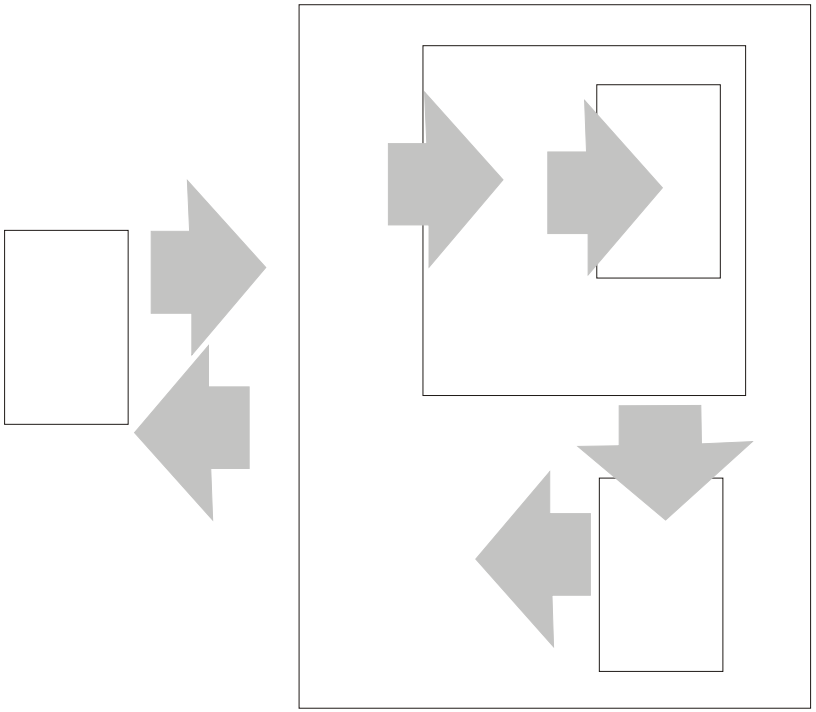
a. introduction

Lors de la mise à jour de solutions développés, il nous faut partir du principe que nous pouvons ajouter des défaillances imprévues.

b. comment construire le plan



Avec une programmation linéaire, il nous est très compliqué de trouver les impacts d'une mise à jour sur les fonctions déjà créées.



Une organisation des traitements orientés "objets" permet de faciliter très sérieusement les mises à jour car chaque objet "vit sa vie" d'une manière autonome.

Donnons donc une directive de programmation en objet ou en Classes.

c. validation des données / traitement / service / journalisation

Chaque donnée modifiée (ou consultée), l'usager avec sa demande peut être archivée (logguée / journalisée).

1. Les journaux, s'ils sont bien faits, peuvent être "rejoués" en cas de sinistre
2. Les journaux montrent ce qu'il s'est passé en cas d'intrusion
3. les journaux nous donnent des indications sur l'usage réel des traitements développés et leurs risques pour mise à jour

1. Est-ce que l'usager, à cette heure-là, sur cette machine, avec ce certificat, peut-il modifier la données ? Si, non : Alerte !
2. est-ce que le Batch peut lancer la mise à jour + nombres de lignes concernées

d. comment définir les données sensibles

Données sensibles NE VEUT PAS dire "données privées".
Données sensibles = données qui a un impact sur le fonctionnement / sur les résultats (calculs fiables) / sur la législation / sur mon risque de perdre un contrat / qui ne possède pas de sauvegarde / qui a besoin de rapidité

e. règles CNIL

Toute information qui est liée à un traitement réclame une déclaration CNIL.
Les usagers ont un droit à l'oubli : 1 mois
La comptabilité à un devoir de : 10 ans (au moins)
Les données nominatives n'ont pas le droit d'être croisées avec une autre application

1. Adopter une politique de mot de passe rigoureuse
2. Concevoir une procédure de création et de suppression des comptes utilisateurs
3. Sécuriser les postes de travail
4. Identifier précisément qui peut avoir accès aux fichiers
5. Veiller à la confidentialité des données vis-à-vis des prestataires
6. Sécuriser le réseau local
7. Sécuriser l'accès physique aux locaux
8. Anticiper le risque de perte ou de divulgation des données
9. Anticiper et formaliser une politique de sécurité du système d'information
10. Sensibiliser les utilisateurs aux « risques informatiques » et à la loi "informatique et libertés"

f. cryptographie et autres méthodes

Couche		Type de chiffrage Suivant la couche ISO concernée
7	Application	Affichage chiffré ou brouillé (quand on n'est pas en face de l'écran)
6	Présentation	Chiffrement applicatif - données stockées sur le disque sont chiffrées
5	Session	Mot de passe en MD5 ou SHA1 ou AES …
4	Transport	Modification de numéros de port à la volée
3	Réseau	VPN ipsec ou VPN SSL
2	Liaison	802.x (authentification niveau 2) - OSD (chiffrage avec certificat pour WIFI)
1	Physique	Coder les fréquences (WIFI) / chiffreurs

Ne pas oublier qu'une donnée non chiffrée sera retrouvable même sur un disque HS ou formaté.