LE GUIDE DU DÉVELOPPEUR

Sécurité dans les développements

Véritable aide-mémoire, ce guide du développeur vous permettra de retrouver les notions fondamentales du parcours **SÉCURITÉ DANS LES DÉVELOPPEMENTS** : gestion des erreurs, guides et référentiels, mécanismes de protection, etc.

PROGRAMME

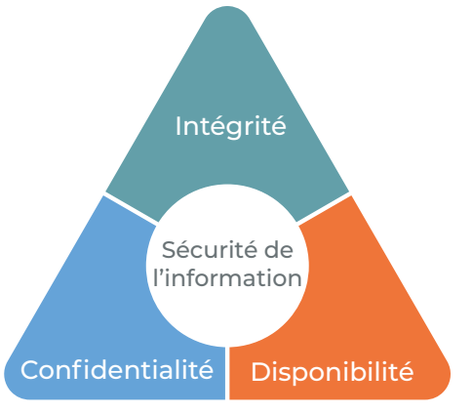
1. Concepts de base
2. Principes génériques
3. Guides et référentiels
4. Authentification
5. Gestion de session et contrôle d'accès
6. Protection des données et cryptographie
7. Logique applicative
8. Sécurité dans les projets et dans le processus de développement
9. Analyse de risques
10. Tests de sécurité et maintien en condition de sécurité

phosforea

+33 (0)5 61 17 08 54

contact@phosforea.com

1 CONCEPTS DE BASE



La sécurité doit être intégrée dans le développement des applications pour assurer :

• **LA DISPONIBILITÉ** : garantit l'accessibilité pour une entité autorisée au moment demandé et selon le temps de réponse défini (ex : garantir l'accès à une application 24h/24h).

• **L'INTÉGRITÉ** : garantit la fiabilité de l'information
(ex : garantir un canal de communication « intègre » entre un destinataire et un expéditeur).

• **LA CONFIDENTIALITÉ** : garantit l'accessibilité de l'information aux personnes autorisées uniquement
(ex : utiliser la cryptographie)

Quelles peuvent être les vulnérabilités dans le développement d'une application ?

Une **VULNÉRABILITÉ** (ou une faille applicative) est une faiblesse de la sécurité pouvant permettre à un attaquant de porter atteinte à la disponibilité, l'intégrité ou la confidentialité de l'application, des données qu'elle contient ou du système hébergeant cette application.

LES DIFFÉRENTS TYPES DE VULNÉRABILITÉS

01 **LOGICIELLE**
Ex : faille créée par le développeur de l'application

02 **CONFIGURATION**
Ex : erreur lors du déploiement

03 **SYSTÈME**
Ex : vulnérabilité intrinsèque au système d'exploitation

04 **MATÉRIELLE**
Ex : utilisation d'un composant matériel vulnérable

Comment interpréter le niveau de risque d'une vulnérabilité ?

NIVEAU DE RISQUE = impact d'une vulnérabilité **x** facilité d'exploitation
FACILITÉ D'EXPLOITATION = niveau d'expertise des attaquants **x** moyens déployés pour exploiter une vulnérabilité

• Une vulnérabilité facile à exploiter ayant un impact élevé sera évaluée comme **CRITIQUE**.
• À l'inverse, une vulnérabilité difficile à exploiter ayant un impact faible sera évaluée comme **MINEURE**.

La liste OWASP référence les bonnes et mauvaises pratiques de développement et communique un ensemble de ressources pour renforcer la sécurité des applications.

2021

TOP 10 OWASP

A1 – Broken Access Control
A2 – Cryptographic Failures
A3 – Injection
A4 – Insecure Design
A5 – Security Misconfiguration
A6 – Vulnerable and Outdated Components
A7 – Identification and Authentication Failures
A8 – Software and Data Integrity Failures
A9 – Security Logging and Monitoring Failures*
A10 – Server-Side Request Forgery (SSRF)*
** From the Survey*

2| PRINCIPES GÉNÉRIQUES

Notons qu'il est essentiel de bien évaluer et maîtriser cette surface d'attaque lors du développement afin de la minimiser autant que possible pour réduire le nombre de contrôles de sécurité nécessaires.

Il existe 3 types de surface d'attaque : logicielle, réseau et humaine.

Zone de confiance : espace dans lequel l'ensemble des flux qui transitent sont considérés comme sûrs.
Objectif : isoler un système avec des mesures de protection adaptées.
Défense en profondeur : successions de barrières de « contrôles de sécurité » appliquées afin d'augmenter le niveau de sécurité globale.

SURFACE D'ATTAQUE = points d'entrée d'un système

VECTEURS D'ATTAQUE = ensemble de flux d'entrée ou sortie

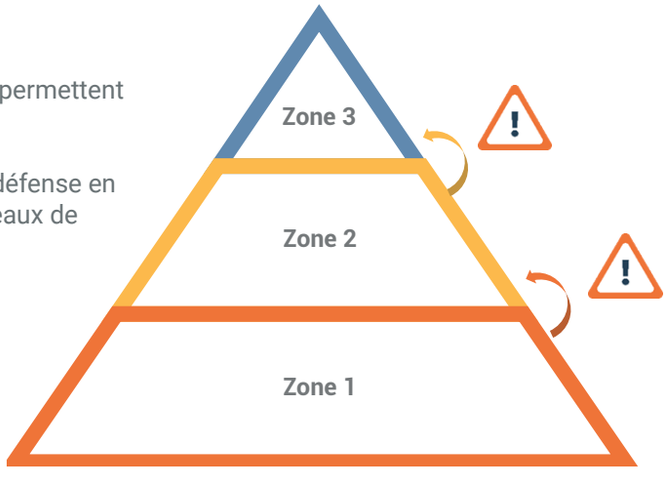
ATTAQUER = utiliser, détourner, corrompre un des flux, afin de l'exploiter

Les principes associés à la défense en profondeur sont les suivants :

- Les biens à protéger sont entourés de plusieurs lignes de défense ;
- Chaque ligne de défense participe à la défense globale ;
- Chaque ligne de défense à un rôle à jouer : affaiblir l'attaque, la gêner, la retarder ;
- Chaque ligne de défense est autonome : l'objectif est d'éviter l'effet « château de cartes ».

Ces successions de barrières assurent un "contrôle de sécurité" et permettent de constituer une ligne de défense adaptée au risque.

Il est possible d'associer les concepts de zone de confiance et de défense en profondeur pour définir plusieurs zones de confiance avec des niveaux de contrôles différents.



Zone de confiance + défense en profondeur = amélioration de la sécurité

3| GUIDES ET RÉFÉRENTIELS

Les référentiels peuvent être utilisés de différentes façons :

- Connaître les vulnérabilités d'un produit
- Connaître les bonnes pratiques
- Connaître les outils disponibles
- Utiliser les moyens de protection adaptés

Comment les utiliser efficacement ?

Les référentiels permettent d'identifier les **principales vulnérabilités** des logiciels du marché, il est essentiel de connaître leur version ainsi que le nom de leur éditeur.

> De nombreux organismes publics et privés recensent des vulnérabilités logicielles.

Quels sont ces organismes ?

- **L'ANSSI** (l'Agence Nationale de la Sécurité des Systèmes d'Information) en France
- **NVD** (National Vulnerability Database) aux Etats-Unis et le CNNVD (Chinese National Vulnerability Database) en Chine
- **SANS Institute** ou les éditeurs de logiciels (ex : CISCO, Redhat, Debian, Microsoft, Google, Apple, Oracle, OpenSSL, Mozilla)
- **NIST** (National Institute of Standards and Technology) ou l'OWASP (Open Web Application Security Project) avec le CVE (Common Vulnerability Enumeration) via « cve.mitre.org ».

Comment évaluer le niveau de risque d'une vulnérabilité ?

L'ÉVALUATION CVSS (Common Vulnerability Scoring System) est un système d'évaluation standardisé de la criticité des vulnérabilités selon des critères objectifs et mesurables ayant pour but d'estimer le niveau de gravité d'une vulnérabilité. Un score de 10 correspond à un niveau **CRITIQUE**.
Cette évaluation est constituée de 3 métriques : base, temporelle et environnementale.

Pour évaluer les vulnérabilités, la **métrique de base** peut être suffisante car cette dernière regroupe deux catégories de métriques : **exploitabilité et impact**.

- **L'exploitabilité** correspond aux métriques de vecteur d'accès, de complexité d'accès et d'authentification.
- Les métriques d'**impact** identifient si la vulnérabilité affecte la confidentialité, l'intégrité ou la disponibilité de la cible.

Il appartient ensuite à l'équipe en charge du maintien en condition de sécurité du produit de suivre régulièrement ces sources pour identifier au plus tôt les nouvelles vulnérabilités.

4 AUTHENTIFICATION

- **IDENTIFICATION** = donner son identité
- **AUTHENTIFICATION** = prouver son identité
  - **AUTHENTIFICATION MULTI-FACTEUR** = plus robuste
  - **AUTHENTIFICATION SSO** = sécurité et simplicité pour l'utilisateur

SSO (SINGLE SIGN-ON) = accès à plusieurs services en effectuant une seule opération d'authentification.
Ex : « Google Auth », « Facebook Auth », « Yahoo », etc.

Comment garantir un niveau de sécurité élevé sur les fonctions d'identification et d'authentification ?

1. Gestion de la casse des identifiants = Attention aux majuscules et aux minuscules !
2. Gestion de la casse des e-mails = norme RFC3696
3. Vérification des e-mails = ex : eric.dupont@site.com
   60 caractères max | 255 caractères max
4. Politique de mot de passe
5. Transfert et stockage de mot de passe = transmis par un canal de communication sécurisé, stocké hashé (=empreinte cryptographique) et salé
6. Récupération de mot de passe = robuste
7. Gestion des erreurs = neutre
8. Prévention contre les attaques sur les mots de passe
9. Supervision de la fonction d'authentification + création de journaux de connexion

BONNES PRATIQUES

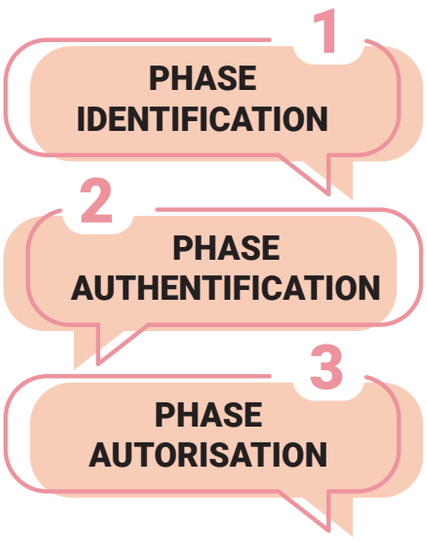
5 GESTION DE SESSION ET CONTRÔLE D'ACCÈS

La phase d'autorisation fait suite à l'authentification et permet d'attribuer les droits correspondants à une entité ou un utilisateur. L'autorisation permet d'implémenter deux concepts interdépendants et importants en sécurité :

• **LE BESOIN D'EN CONNAÎTRE** = seuls les entités ou utilisateurs ayant vraiment le besoin d'en connaître ont accès à certaines données ou systèmes.

• **LE MOINDRE PRIVILÈGE** = une entité ou un utilisateur a les droits strictement nécessaires sur le périmètre de ses activités uniquement.

L'autorisation est une des phases gérées par les protocoles AAA : Authentification (« *Authentication* »), Autorisation (« *Authorization* ») » et Traçabilité (« *Accounting* ») et son principe est le suivant : authentifier l'utilisateur à l'aide d'une méthode d'authentification, l'autoriser en se basant sur ses attributs et loguer ses tentatives d'accès.



Les modèles de contrôle d'accès

DAC (*Discretionary Access Control*) : basé sur une politique d'accès définie par le propriétaire d'une ressource.
MAC (*Mandatory Access Control*) : attribue des étiquettes de sensibilité aux objets et les compare au niveau d'habilitation du demandeur.
RBAC (*Role Based Access Control*) : accorde ou refuse une demande d'accès en fonction de l'appartenance du demandeur à un groupe organisationnel ou fonctionnel.
PBAC (*Permission Based Access Control*) : basé sur les permissions où chaque action sur les objets est identifiée.

Qu'est-ce qu'une session ?

Échange d'informations temporaire et interactif entre deux entités, la session est représentée par un identifiant (aussi appelé token ou jeton). L'identifiant de session est la cible de nombreuses attaques telles que :
PRÉDICTION - VOL - INTERCEPTION - FIXATION - USURPATION

QUELQUES BONNES PRATIQUES

- Identifiants de session :
  - Complexes
  - >= 128 bits
  - Création côté serveur (générateurs de nombres aléatoires sécurisés)
  - Validité de la session => adaptée à l'application
- Conserver le secret au session ID
- Créer un nouvel ID de session pour l'authentification
- Invalider la session ID lors de la déconnexion
- Protéger les données de session stockées côté serveur
- Activer les options de sécurité (HttpOnly, secure, Token anti-CSRF)
- Prévoir une interface pour lister les sessions actives sur les appareils
- Désactiver toutes les sessions actives suite au changement de mot de passe
- Loguer les actions sur les identifiants de session
- Avoir une durée de validité de session en accord avec la sensibilité des données et les besoins métiers.

> Utilisez de préférence les fonctionnalités de gestion de session offertes par les Framework !

6 PROTECTION DES DONNÉES ET CRYPTOGRAPHIE

> La cryptographie permet de **rendre incompréhensible** un message ou une donnée à ceux qui ne sont pas autorisés à en prendre connaissance.

Différentes opérations cryptographiques :

- **Chiffrement symétrique** : opération qui utilise une clé secrète pour chiffrer un message.
- **Chiffrement asymétrique** : opération qui repose sur des structures mathématiques complexes avec des clés longues (1024 à 2048 bits).
- **Chiffrement hybride** : combinaison des systèmes à clé publique (asymétrique) et à clé secrète (symétrique).
- **Hachage** : opération qui génère des identifiants uniques et non répétitifs à partir d'informations données.
- **Encodage** : opération qui transforme une donnée de façon réversible afin de faciliter sa manipulation.
- **Signature** : opération qui permet de certifier que les données ont bien été créées par l'entité qui émet le message.

Chacune d'elle permet d'assurer un principe de sécurité : **confidentialité, intégrité, non-répudiation et authenticité.**

Des protocoles clé-en-main (SSL, TLS, IPSEC, certificats, PKI, etc.) sont disponibles pour aider le développeur à s'assurer de la confidentialité, de l'intégrité et de l'authenticité des données lorsqu'elles sont transmises entre un client et un serveur.

QUELQUES BONNES PRATIQUES

- Utiliser des algorithmes sûrs :
  - Chiffrement asymétrique : RSA 2048 bits ou ECC 224 bits
  - Chiffrement symétrique : AES 128 bits
  - Hachage : SHA2 256 bits
  - Intégrité des messages : HMAC-SHA2
  - Hachage de mot de passe : Argon2, PBKDF2, Scrypt, Bcrypt
- Ne jamais implémenter d'algorithme cryptographique « maison »
- Utiliser des nombres suffisamment aléatoires :
  - 256 bits
  - /dev/urandom (et non /dev/random)
- Hacher les mots de passe avec un sel
- Limiter les protocoles cryptographiques supportés aux dernières versions (ex : TLS v1.2 et TLS v1.3)

7 LOGIQUE APPLICATIVE

La validation des données est une des mesures de sécurité les plus importantes en développement qui :

- Permet de se prémunir contre de multiples attaques ;
- Doit être réalisée par une entité de confiance ;
- Doit adopter une stratégie par liste blanche pour faciliter la maintenabilité du code.

2 stratégies de validation sont possibles : Blacklist et Whitelist, mais seule cette dernière est à privilégier !

7 principes de validation

1.Ne pas faire confiance à une personne externe
2.Valider toutes les entrées
3.Effectuer la validation sur l'ensemble des éléments
4.Convertir les données vers une mise en forme canonique
5.Faire valider les données par une entité de confiance
6.Centraliser les fonctions de validation des données
7.Appliquer des stratégies de validation

Comment mettre en place un mécanisme de gestion des erreurs ?

Lorsque des erreurs sont générées, une application adoptant la stratégie **Fail-Open** continue son chemin d'exécution alors que celle implémentant l'option **Fail-Closed** interrompt complètement son traitement et revient à un état initial.
Fail-closed est à privilégier pour la stratégie d'arrêt du traitement en cas d'erreur.
Ne pas divulguer d'informations sensibles dans les messages d'erreur ou dans les logs !

La journalisation des événements applicatifs permet de réaliser une investigation à la suite d'un incident de sécurité ou simplement d'un dysfonctionnement du mécanisme d'authentification.

Comment se protéger ?

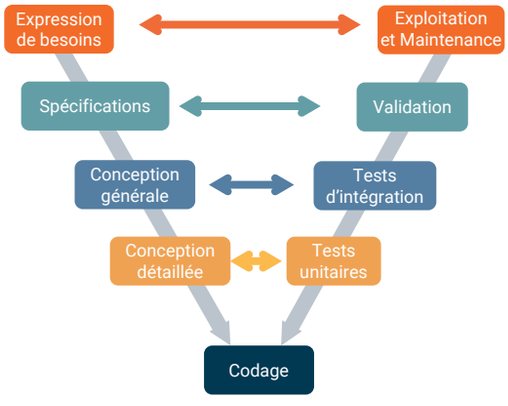
La gestion des ressources porte sur deux aspects : la **protection des ressources logiques** (fichiers sources, images, fichiers de données, logs) et la **gestion des ressources physiques** (espace disque, mémoire RAM, processeur).

La gestion des ressources logiques et physiques assure la disponibilité d'une application.

8 SÉCURITÉ DANS LES PROJETS ET DANS LE PROCESSUS DE DÉVELOPPEMENT

Quelles sont les différentes phases du cycle de vie d'une application ?

Un projet de développement inclut 8 phases :
l'initialisation, la planification, la conception, le développement, l'essai, l'implantation, l'entretien et l'élimination.
Dans chacune de ces phases, des activités spécifiquement liées à la sécurité sont à réaliser pour s'assurer que la sécurité est intégrée au cœur même du système.
On parle de « *security by design* » quand la sécurité est intégrée dès la phase de conception.



Quels processus de développement pour mon application ?

1 - La méthode du cycle en V

Elle utilise une approche séquentielle et linéaire des phases :
• un flux d'activité descendant qui détaille le produit jusqu'à la phase de codage,
• un flux ascendant qui assemble le produit en vérifiant sa qualité.

2 - La méthode Agile

• Évite l'effet tunnel
• Implique le client pour tendre vers le résultat attendu
• Intègre les activités de sécurité dans chaque itération

DEVSECOPS

Les méthodes Agile ou DevSecOps permettent une intégration plus simple des activités de sécurité. Leur mode itératif réduit le périmètre fonctionnel à traiter à chaque itération.

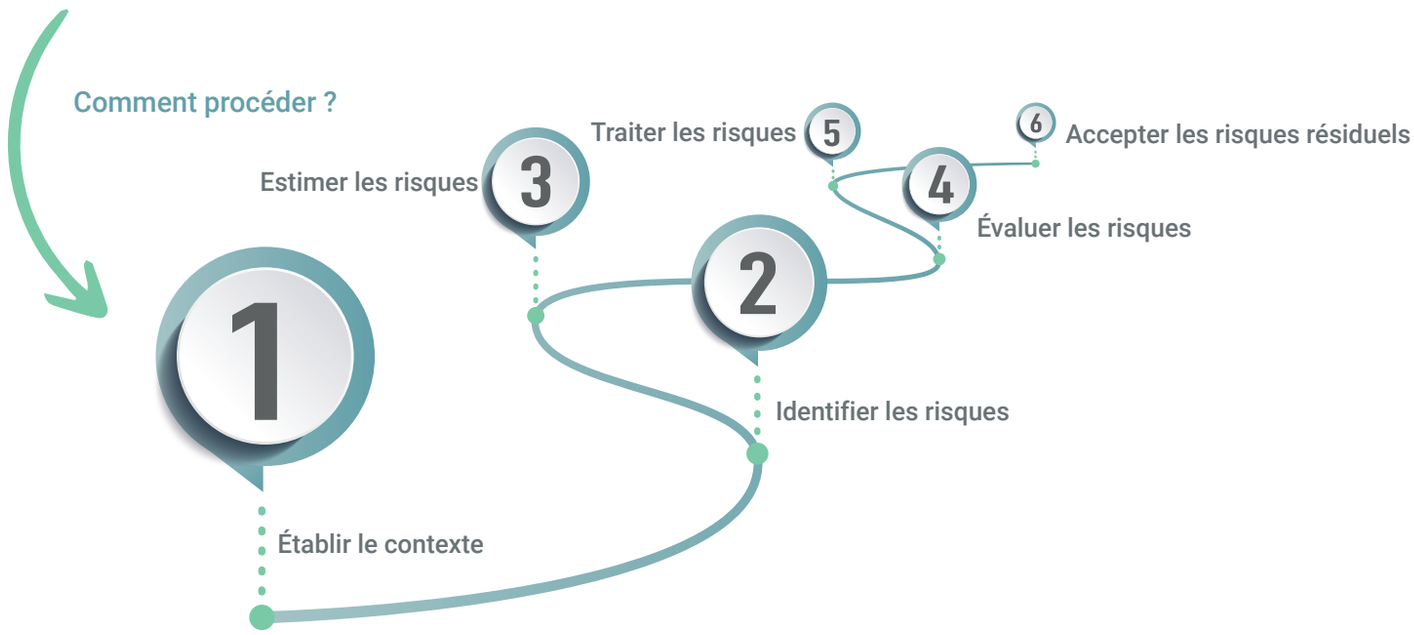
La sécurisation d'un processus de développement logiciel doit s'appliquer sur la phase de codage, les tests, la documentation, les livraisons et l'exploitation.
La mise en place de mesures de sécurisation dans le processus de développement permet de réduire les risques.

9 ANALYSE DE RISQUE

Un **RISQUE** est la possibilité qu'une **MENACE** exploite une **VULNÉRABILITÉ** d'un actif ou d'un groupe d'**ACTIF** et nuise à l'organisation.

MENACE X VULNÉRABILITÉ X IMPACT = RISQUE

Il est nécessaire de mettre en place une **ANALYSE DE RISQUES** pour garantir un traitement efficace des risques pesant sur un système ou sur une organisation.
Les mesures de protection seront ainsi priorisées et optimisées dans des délais et des coûts acceptables.



L'évaluation du niveau de gravité : donner une note, des conséquences.

L'évaluation du niveau de vraisemblance : exprimer le niveau de faisabilité du scénario.

L'évaluation des risques : définir un niveau de **risque**, en tenant compte des **niveaux de gravité** et de **vraisemblance**.

Il convient de se projeter sur les risques finaux, une fois les mesures proposées mises en application et s'assurer que les risques futurs soient déclarés comme acceptables pour l'organisation.

R4 - Critique	Mesure obligatoire et prioritaire
R3 - Haut	Mesure identifié et planifiée
R2 - Moyen	Choix de traitement du risque pris par la direction
R1 - Faible	Aucune action nécessaire pour traiter le risque

10| TESTS DE SÉCURITÉ ET MAINTIEN EN CONDITION DE SÉCURITÉ

Quelles activités permettent de vérifier régulièrement la sécurité ?

Afin de prévenir toutes failles de sécurité, il est possible de mettre en place un **AUDIT DE CODE** ayant plusieurs finalités : **MAINTENABILITÉ**, **ROBUSTESSE**, **PERFORMANCE** ou **SÉCURITÉ**.

L'audit de code source consiste à analyser tout ou partie du code source ou des conditions de compilation d'une application dans le but d'y découvrir des non-conformités.

L'audit de code sécurité va permettre d'identifier des vulnérabilités qui peuvent être liées à de mauvaises pratiques de programmation ou à des erreurs de logique qui pourraient avoir un impact en matière de sécurité.

Deux types d'analyses du code source co-existent et sont complémentaires : **l'analyse statique** et **l'analyse dynamique.**

Qu'est-ce qu'un test d'intrusion ?

Le **TEST D'INTRUSION** consiste à :
- simuler les actions d'un attaquant
- évaluer le niveau de sécurité et la résistance du système testé.

3 types d'approches pour les tests d'intrusion :
- **boite noire** = l'auditeur réalise le test sans connaissance de l'environnement ;
- **boite grise** = l'auditeur dispose de comptes utilisateurs sur l'application ciblée ;
- **boite blanche** = l'auditeur a accès à toutes les informations disponibles sur l'application.

Les tests d'intrusion suivent une méthodologie en plusieurs phases qui se terminent par un reporting auprès de l'audité (rapport de test d'intrusion).

L'ensemble de ces outils sont disponibles dans des distributions Linux dédiées aux tests d'intrusion comme Kali Linux ou Parrot.

De nombreux outils sont disponibles pour faciliter le travail du *pentesteur* : scanners de réseaux, outils d'analyse de vulnérabilités, d'attaque de sites web ou de réseaux sans fil, outils d'écoute et d'usurpation d'identité, d'exploitation de vulnérabilité, etc.

En résumé ?

- **Audit de code** = consiste à identifier des vulnérabilités liées à de mauvaises pratiques de programmation ou à des erreurs de logique. Il peut être mené alors que l'application n'est pas terminée.

- **Tests d'intrusion** = consiste à simuler l'action d'un pirate informatique afin d'éprouver les mécanismes de sécurité de l'application. Il nécessite une autorisation préalable.
Les tests d'intrusion sont complémentaires aux audits de code.