Secure Coding
OWASP SAMM

Halim Djerroud

révision : 0.1

Plan

Introduction à OWASP SAMM

Introduction à OWASP SAMM

Objectifs du chapitre :

1. Présentation d'OWASP et de ses projets.
2. Qu'est-ce que OWASP SAMM ?
3. Objectifs et avantages de SAMM :
   - Évaluer la maturité des pratiques de sécurité.
   - Planifier et améliorer les processus.
4. Différences avec d'autres modèles de maturité (ex. : BSIMM, CMMI).

OWASP SAMM

Figure { Logo OWASP SAMM

OWASP Rappel

OWASP (Open Worldwide Application Security Project) est une organisation mondiale à but non lucratif qui se consacre à améliorer la sécurité des logiciels. Créée en 2001, OWASP vise à fournir des ressources, des outils, des méthodologies, et des formations pour aider les développeurs, les entreprises et les professionnels de la sécurité à concevoir, développer et maintenir des applications sécurisées.

Projets phares d'OWASP : OWASP développe et maintient plusieurs projets connus dans le domaine de la cybersécurité. Voici les principaux :

- OWASP Top 10
- OWASP ZAP (Zed Attack Proxy)
- **OWASP SAMM (Software Assurance Maturity Model)**
- OWASP Juice Shop
- OWASP Cheat Sheets (aide-mémoire)
- OWASP Dependency-Check

OWASP SAMM (Software Assurance Maturity Model)

- Un cadre permettant d'évaluer et d'améliorer les pratiques de développement logiciel en termes de sécurité.
- Il guide les organisations dans l'intégration de la sécurité à toutes les étapes du cycle de vie du développement logiciel (SDLC).

OWASP SAMM est un modèle de maturité destiné à guider les organisations dans l'intégration de la sécurité des applications tout au long du cycle de vie du développement logiciel (SDLC). Il offre un cadre structuré pour évaluer, améliorer et maintenir la sécurité dans les processus de développement, tout en s'adaptant à la taille, aux besoins et aux objectifs spécifiques de chaque organisation.

Objectifs d'OWASP SAMM

1. **Évaluation :** Permettre aux organisations d'évaluer leur maturité actuelle en matière de sécurité des logiciels.
2. **Planification :** Fournir une feuille de route pour améliorer progressivement leurs pratiques de sécurité.
3. **Adoption personnalisée :** Être flexible pour s'adapter aux différents modèles de développement, qu'ils soient traditionnels, agiles, ou DevOps.
4. **Mesure continue :** Aider à suivre les progrès réalisés au fil du temps.

Versions OWASP SAMM

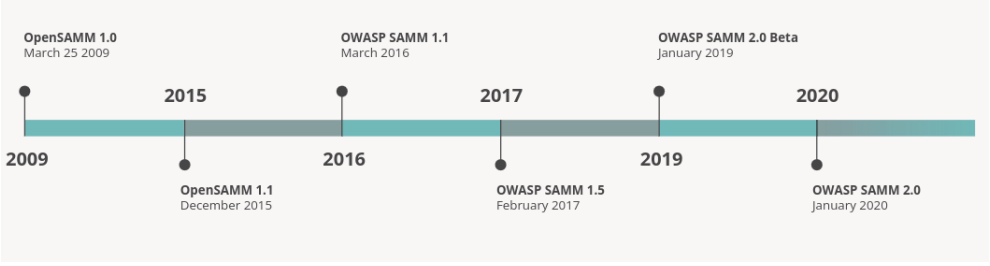

Figure { Source OWASP.

Structure et composantes de SAMM

Structure et composantes de SAMM

Objectifs du chapitre :

1. Les cinq domaines de SAMM.
2. Aperçu des 15 pratiques.
3. Mécanismes de maturité : niveaux 1, 2 et 3.

Les cinq domaines de SAMM

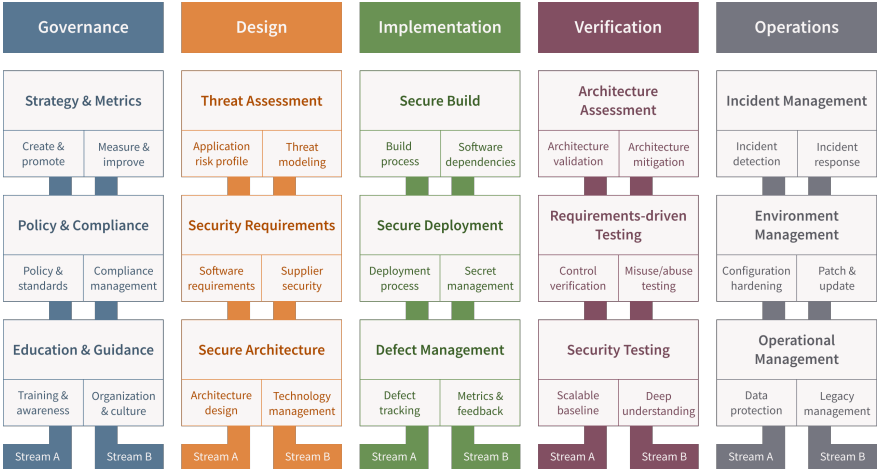

Figure { Source OWASP

Les cinq domaines de SAMM

1. Gouvernance : Met en place les politiques et le cadre organisationnel nécessaires pour la gestion de la sécurité.

- Activités principales :
  - Stratégie & métriques
  - Gestion des politiques
  - Éducation & conseils

2. Conception : Concerne l'identification des exigences de sécurité et la conception de logiciels sûrs.

- Activités principales :
  - Gestion des menaces
  - Architecture de sécurité
  - Conception sécurisée

3. Implémentation : Traite de l'intégration des pratiques de sécurité lors du développement de logiciels.

- Activités principales :
  - Contrôles de sécurité dans le code
  - Tests unitaires sécurisés
  - Gestion des dépendances logicielles

4. Vérification : Se concentre sur la validation et les tests de sécurité pour s'assurer que les logiciels sont exempts de vulnérabilités.

- Activités principales :
  - Tests de sécurité automatisés
  - Examens de code
  - Tests manuels et audit

Les cinq domaines de SAMM

5. Déploiement : Concerne les pratiques de sécurité après la livraison et le déploiement des logiciels.

- Activités principales :
  - Surveillance des opérations
  - Gestion des vulnérabilités
  - Planification de la réponse aux incidents

Niveaux de maturité d'OWASP SAMM :

Chaque domaine de pratique est évalué sur trois niveaux de maturité (Maturity Levels) :

1. **Niveau 1 :** Les pratiques de sécurité de base sont en place, souvent réactives.
2. **Niveau 2 :** Les pratiques deviennent formalisées et standardisées.
3. **Niveau 3 :** Les pratiques sont optimisées, intégrées dans les processus métier, et proactives.

Niveau 1 : Initial

- **Caractéristiques principales :**
  - Les pratiques de sécurité sont *ad hoc* ou *réactives*.
  - Les efforts sont souvent motivés par des incidents ou des pressions externes (audit, conformité, exigences contractuelles).
  - Peu ou pas de formalisation des processus.
- **Exemples concrets :**
  - Des mesures de sécurité appliquées uniquement en réponse à des failles ou des incidents.
  - Formation à la sécurité limitée à des sessions occasionnelles, sans structure ni suivi.
  - Utilisation sporadique d'outils ou de méthodologies sans stratégie globale.
- **Objectif à ce niveau :**
  - *Prendre conscience de l'importance de la sécurité* et initier les premières pratiques.

Niveau 2 : Standardisé

- **Caractéristiques principales :**
  - Les pratiques de sécurité sont *documentées, répétables et standardisées.*
  - Les processus commencent à être intégrés dans le cycle de vie du développement logiciel (SDLC).
  - Une collaboration inter-équipes émerge pour appliquer les pratiques.
- **Exemples concrets :**
  - Mise en place de politiques de sécurité documentées (ex. : règles pour les revues de code, gestion des vulnérabilités).
  - Formation régulière des équipes de développement sur des sujets comme les injections SQL ou les XSS.
  - Adoption d'outils pour l'analyse de vulnérabilités intégrés dans les pipelines CI/CD.
- **Objectif à ce niveau :**
  - *Créer des processus cohérents et prévisibles*, en assurant leur mise en œuvre dans des projets similaires.

Niveau 3 : Optimisé

- **Caractéristiques principales :**
  - Les pratiques de sécurité sont *proactives, optimisées et intégrées de manière systémique* dans tous les projets et processus.
  - Un retour sur investissement en sécurité est mesuré et optimisé.
  - Les équipes collaborent de manière fluide pour anticiper et prévenir les risques.
- **Exemples concrets :**
  - Intégration complète de la sécurité dans le SDLC, avec des évaluations régulières de l'efficacité des pratiques.
  - Automatisation avancée (ex. : outils d'analyse dynamique et de fuzzing dans le pipeline).
  - Alignement stratégique entre les objectifs métier et les priorités de sécurité.
  - Suivi d'indicateurs de performance (KPIs) pour mesurer l'efficacité des initiatives de sécurité.
- **Objectif à ce niveau :**
  - *Devenir un leader en sécurité logicielle*, avec une capacité à innover tout en minimisant les risques.

Aperçu des 15 pratiques par domaine (1. Gouvernance)

1. Gouvernance : Ces pratiques définissent les fondations organisationnelles pour intégrer la sécurité dans les processus.

1. Stratégie & métriques
   - Établir une vision stratégique pour la sécurité des logiciels.
   - Suivre les progrès via des indicateurs clés (KPIs).
   - Mesurer les objectifs en matière de sécurité.
2. Gestion des politiques
   - Définir, communiquer et appliquer des politiques de sécurité applicative.
   - Garantir la conformité aux normes internes et externes.
3. Éducation & conseils
   - Former les parties prenantes (développeurs, chefs de projet, etc.) aux bonnes pratiques de sécurité.
   - Fournir des ressources pédagogiques adaptées aux rôles.

Aperçu des 15 pratiques par domaine (2. Conception)

2. Conception : Ces pratiques concernent l'identification des exigences de sécurité dès le début du cycle de développement.

4. Gestion des menaces
   - Identifier les menaces potentielles qui pourraient affecter l'application.
   - Réaliser des analyses de risques (par ex., STRIDE).
5. Architecture de sécurité
   - Concevoir une architecture qui intègre des contrôles de sécurité robustes.
   - Assurer la modularité et la défense en profondeur.
6. Conception sécurisée
   - Implémenter des principes de conception tels que l'autorisation minimale ou la séparation des responsabilités.
   - Minimiser les surfaces d'attaque.

Aperçu des 15 pratiques par domaine (3. Implémentation)

3. Implémentation : Ces pratiques garantissent que le code est écrit de manière sécurisée et sans vulnérabilités.

7. Contrôles de sécurité dans le code
   - Inclure des mécanismes de sécurité (authentification, cryptographie, etc.) dans le code.
   - Suivre les meilleures pratiques pour écrire du code sûr.
8. Tests unitaires sécurisés
   - Écrire des tests unitaires qui valident les fonctionnalités de sécurité.
   - Utiliser des frameworks de tests pour automatiser ces validations.
9. Gestion des dépendances logicielles
   - Identifier et surveiller les dépendances tierces (bibliothèques, frameworks).
   - Vérifier les vulnérabilités dans les dépendances à l'aide d'outils (par ex., OWASP Dependency-Check).

Aperçu des 15 pratiques par domaine (4. Vérification :)

4. Vérification : Ces pratiques assurent que les logiciels sont testés pour détecter et corriger les vulnérabilités.

10. Tests de sécurité automatisés
    - Automatiser les analyses statiques (SAST ((Static Application Security Testing))) et dynamiques (DAST(Dynamic Application Security Testing)) pour identifier les vulnérabilités dans le code.
    - Intégrer ces outils dans les pipelines CI/CD.
11. Examens de code
    - Réaliser des revues manuelles ou semi-automatisées du code pour identifier les failles complexes.
    - Adopter une approche collaborative (par ex., revues de pairs).
12. Tests manuels et audits
    - Effectuer des tests manuels approfondis, comme des tests d'intrusion.
    - Réaliser des audits réguliers de la sécurité.

Aperçu des 15 pratiques par domaine (5. Déploiement :)

5. Déploiement : Ces pratiques traitent de la sécurité post-déploiement, une fois que les applications sont en production.

13. Surveillance des opérations
    - Mettre en place des mécanismes de surveillance continue pour détecter les anomalies de sécurité.
    - Utiliser des systèmes de journalisation et d'alerte (SIEM).
14. Gestion des vulnérabilités
    - Identifier, documenter et corriger rapidement les vulnérabilités découvertes en production.
    - Mettre en œuvre des politiques de gestion des correctifs (patch management).
15. Planification de la réponse aux incidents
    - Préparer une stratégie pour réagir efficacement aux incidents de sécurité.
    - Tester régulièrement les plans de réponse aux incidents pour garantir leur efficacité.

Aperçu des 15 pratiques par domaine

Domaine	Pratique 1	Pratique 2	Pratique 3
Gouvernance	Stratégie & métriques	Gestion des politiques	Éducation & conseils
Conception	Gestion des menaces	Architecture de sécurité	Conception sécurisée
Implémentation	Contrôles de sécurité dans le code	Tests unitaires sécurisés	Gestion des dépendances logicielles
Vérification	Tests de sécurité automatisés	Examens de code	Tests manuels et audits
Déploiement	Surveillance des opérations	Gestion des vulnérabilités	Planification de réponse aux incidents

Table { Les 15 pratiques d'OWASP SAMM

Application d'OWASP SAMM

- Ces 15 pratiques permettent aux entreprises/organisations d'avoir une approche complète de la sécurité, couvrant à la fois les aspects stratégiques, techniques et opérationnels.
- Chaque pratique est accompagnée de niveaux de maturité (1 à 3), permettant une amélioration progressive.

Les streams dans OWASP SAMM

Les streams

un stream fait référence à un sous-ensemble ou à une division logique au sein d'une pratique. Chaque pratique de sécurité est divisée en deux streams qui représentent des aspects complémentaires ou progressifs de cette pratique.
Les streams aident à mieux structurer et prioriser les actions nécessaires pour améliorer la sécurité, en permettant une évolution progressive de la maturité.

Exemple des streams dans OWASP SAMM

1. Stratégie & métriques (Gouvernance)
   - **Stream 1 : Définir les objectifs et la stratégie de sécurité** : Mettre en place une vision claire des objectifs et de la direction en matière de sécurité logicielle.
   - **Stream 2 : Mesurer et suivre les progrès :** Identifier et suivre les métriques pour évaluer l'efficacité des efforts en sécurité.
2. Gestion des menaces (Conception)
   - **Stream 1 : Identification des menaces :** Analyser et documenter les menaces potentielles pour les systèmes et les applications.
   - **Stream 2 : Analyse des risques :** Évaluer la probabilité et l'impact des menaces identifiées afin de prioriser les actions.
3. Contrôles de sécurité dans le code (Implémentation)
   - **Stream 1 : Développement sécurisé :** Implémenter des mécanismes de sécurité directement dans le code.
   - **Stream 2 : Revue des contrôles de sécurité :** Valider les contrôles implémentés pour garantir leur efficacité.

Rôle des streams dans l'évaluation SAMM

Les streams permettent de :

1. **Détailler les pratiques :** Chaque pratique est décomposée en parties spécifiques, ce qui facilite leur compréhension et leur mise en œuvre.
2. **Progresser par étapes :** Une organisation peut atteindre un niveau de maturité différent dans chaque stream d'une même pratique.
3. **S'adapter aux besoins :** Les streams offrent de la flexibilité pour se concentrer sur des aspects prioritaires d'une pratique.

Les streams d'OWASP SAMM divisent les pratiques en tâches ou objectifs spécifiques pour une gestion plus granulaire et un alignement progressif avec les niveaux de maturité.

Utilisation de SAMM dans une organisation

Utilisation de SAMM dans une organisation

Objectifs du chapitre :

1. Étapes de mise en œuvre
2. Méthodologies de collecte d'informations (interviews, audits, questionnaires).
3. Adaptation du modèle à différentes tailles d'entreprises.

Compréhension des objectifs de l'organisation

Avant de commencer, il est crucial de définir les objectifs liés à la sécurité logicielle :

1. Protéger les données des utilisateurs.
2. Respecter les réglementations (GDPR, ISO, etc.).
3. Réduire les risques liés aux vulnérabilités logicielles.
4. Améliorer la qualité et la robustesse des logiciels.

Étapes de mise en œuvre

1. Planification initiale
2. Évaluation initiale
3. Définition de la feuille de route
4. Mise en œuvre des pratiques SAMM
5. Suivi et mesure
6. Réévaluation périodique
7. Communication et consolidation

1. Planification initiale

1. **Constitution de l'équipe :**
   - Réunir les parties prenantes : responsables sécurité, chefs de projet, développeurs, etc.
   - Désigner un champion pour piloter la mise en œuvre.
2. **Définition des objectifs :**
   - Identifier les priorités telles que la réduction des vulnérabilités, la conformité réglementaire, ou l'amélioration des processus.
3. **Alignement stratégique :**
   - Assurer l'alignement avec les objectifs métier et les contraintes organisationnelles.

2. Évaluation initiale

1. **Collecte des informations :**
   - Analyser les processus existants en matière de développement et de sécurité.
2. **Utilisation de l'outil d'évaluation SAMM :**
   - Évaluer les pratiques dans les cinq domaines : *Gouvernance*, *Conception*, *Implémentation*, *Vérification*, *Déploiement*.
   - Attribuer des scores de maturité (Niveau 1 à 3).
3. **Identification des lacunes :**
   - Identifier les pratiques manquantes ou nécessitant des améliorations.

3. Définition de la feuille de route

1. **Priorisation des activités :**
   - Identifier les domaines critiques et les prioriser selon l'impact et les ressources disponibles.
2. **Définition des étapes :**
   - Établir des jalons clairs et associer chaque étape à des indicateurs clés de performance (KPI).
3. **Planification des ressources :**
   - Allouer les budgets, outils et ressources nécessaires.

4. Mise en œuvre des pratiques SAMM

1. **Adoption progressive :**
   - Déployer les activités prioritaires une par une.
   - Exemple : intégrer des évaluations de risques dans les processus de conception.
2. **Sensibilisation et formation :**
   - Former les équipes sur les nouvelles pratiques.
3. **Intégration dans les processus existants :**
   - Incorporer les pratiques SAMM dans les workflows organisationnels (exemple : pipelines CI/CD).

5. Suivi et mesure

1. **Métriques et reporting :**
   - Utiliser des indicateurs pour mesurer l'efficacité (exemple : réduction des vulnérabilités).
2. **Documentation des progrès :**
   - Maintenir un registre des pratiques mises en œuvre et des améliorations.

6. Réévaluation périodique

1. **Évaluation continue :**
   - Réévaluer la maturité à l'aide des outils SAMM.
2. **Amélioration continue :**
   - Ajuster la feuille de route en fonction des nouveaux besoins et défis.

7. Communication et consolidation

1. **Rapports réguliers :**
   - Communiquer les progrès aux parties prenantes.
2. **Renforcement de la culture de sécurité :**
   - Intégrer la sécurité dans les pratiques organisationnelles à long terme.

Exemple de feuille de route simplifiée (6 mois)

Mois	Action	Résultat attendu
1-2	Évaluation initiale avec SAMM	Rapport des lacunes et plan priorisé
3-4	Mise en œuvre des pratiques clés	Adoption des politiques de sécurité et évaluation des risques
5	Formation des équipes	Compétences renforcées pour intégrer les pratiques
6	Réévaluation	Progression mesurée et ajustement des priorités

Outils et ressources pour OWASP SAMM

Outils et ressources pour OWASP SAMM

Objectifs du chapitre :

1. SAMM Toolbox : Guide interactif.
2. SAMM Benchmark
3. Autres outils OWASP (Zap proxy par exemple)
4. Liens vers les ressources OWASP (documents, vidéos, communauté).

SAMM Toolbox

Une boîte à outils interactive fournie par OWASP pour faciliter l'évaluation et la gestion de la maturité des pratiques de sécurité logicielle.

- Réalisation d'évaluations initiales et périodiques.
- Suivi des progrès dans les cinq domaines de SAMM.
- Génération de rapports pour identifier les lacunes et priorités.

`https://github.com/OWASP/samm/tree/master/Supporting%20Resources/v2.0/toolbox`

SAMM Toolbox

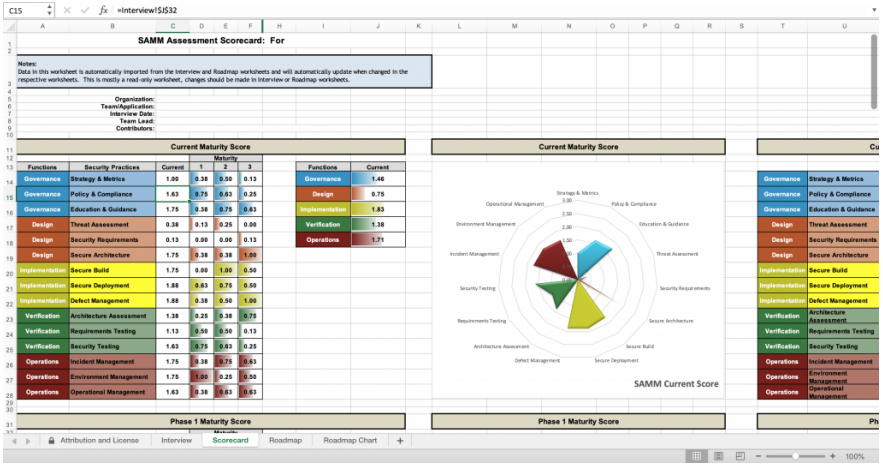

Figure { Source OWASP

SAMM Toolbox

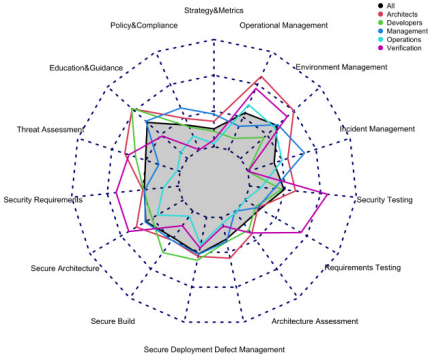

Figure { Source OWASP

Exercice pratique

Exercice pratique

Objectifs du chapitre :

1. Simulation d'évaluation de maturité pour une organisation fictive.
2. Création d'un plan d'amélioration sur une pratique spécifique.
3. Discussions en groupe.

Contexte de l'exercice

Vous êtes membre de l'équipe sécurité travaillant sur un projet nommé **TaskManager Pro**.

But : Appliquer les principes d'OWASP SAMM pour :

- Évaluer la maturité d'un projet logiciel fictif.
- Identifier des axes d'amélioration.
- Proposer un plan d'action pour renforcer les processus de sécurité.

Situation actuelle :

- Pas de politique de sécurité écrite.
- Aucun test de sécurité avant mise en production.
- Dépendances non mises à jour systématiquement.
- Une seule formation en sécurité suivie par un membre de l'équipe.

Étapes de l'exercice

Travail en groupes :

1. **Partie 1 : Évaluation de la maturité**
   - Analysez les pratiques selon OWASP SAMM.
   - Attribuez un score de maturité aux domaines suivants :
     - Gouvernance
     - Conception
     - Implémentation
     - Vérification
     - Déploiement
2. **Partie 2 : Proposition d'amélioration**
   - Identifiez les priorités.
   - Proposez des actions concrètes pour améliorer deux pratiques spécifiques.
3. **Partie 3 : Restitution**
   - Présentez le score actuel, les lacunes, et le plan d'amélioration.

Livrables

À rendre :

- Grille d'évaluation OWASP SAMM avec les scores attribués.
- Plan d'amélioration détaillant :
  - Étapes d'amélioration.
  - Responsables.
  - Estimation du temps nécessaire.

Critères d'évaluation

Votre travail sera évalué sur :

- Qualité et pertinence de l'analyse des lacunes.
- Faisabilité des actions proposées.
- Cohérence avec les principes d'OWASP SAMM.

Conclusion et perspectives

Conclusion et perspectives

Objectifs du chapitre :

1. Synthèse des avantages de SAMM.
2. Les tendances futures en sécurité logicielle.
3. Invitation à contribuer au projet OWASP.

Pourquoi adopter OWASP SAMM ?

- **Adaptable :** Convient aux petites startups comme aux grandes entreprises.
- **Orienté business :** Met en lien les objectifs commerciaux et les pratiques de sécurité.
- **Communautaire et open source :** Libre d'accès, il bénéficie d'une communauté mondiale qui l'enrichit constamment.