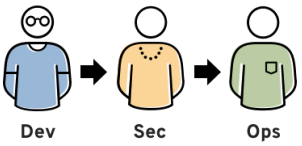
Laboratoire 2: Tests de pénétration et Outils pour Intégration et Déploiement continus

LOG8100:
DEVSECOPS



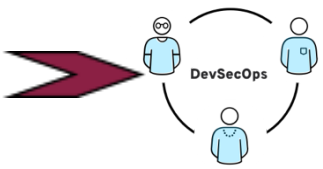


Table des matières

Évaluation et déroulement des TPs

Les travaux pratiques constituent une partie importante du cours de ***LOG8100*** et ont pour objectif de vous permettre d'appliquer les notions et les principes de ***DevSecOps*** étudiés dans les séances de cours.

Les travaux pratiques vous permettront d'élaborer des stratégies de test et d'utiliser les différents outils disponibles pour analyser et évaluer la qualité des logiciels en termes de sécurité.
À travers les TP, vous pouvez acquérir une expérience pratique précieuse pour comprendre et traiter les vulnérabilités de sécurité des applications dans un environnement sûr et contrôlé.

La note des deux travaux pratiques et du projet représente 45% de la note globale du cours. Il est donc vivement recommandé de les aborder avec sérieux, en faisant appel à votre créativité et à votre esprit critique, afin de les réussir au mieux.

Évaluation

NATURE	NOMBRE	Mode de réalisation	PONDÉRATION	DATE
Examen final	1	Individuel	25%	Date à définir (en décembre)
Quiz	3	Individuel	15%	Quiz 1 : 20 sept. 2024 Quiz 2 : 25 oct. 2024 Quiz 3 : 22 nov. 2024
Exposés	1	Equipe	15%	14, 21, 28 nov. 2024 (en fonction du nombre des étudiants)
Travaux Pratiques	TP #1 Analyse des vulnérabilités logicielles	Equipe	10%	12 sept. 2024
	TP #2 Test de pénétration et Outils pour intégration et déploiement continus		15%	26 sept. , 10 oct. 2024
	Projet		20%	1, 14, 28 nov. 2024

La collaboration avec vos collègues est permise pendant et en dehors des séances de laboratoire.

Règles

Cependant, veuillez noter que les **règles** concernant le plagiat restent en vigueur en tout temps

Il est recommandé d'utiliser les outils mentionnés dans l'énoncé, mais vous avez également la possibilité d'explorer d'autres outils. Cependant, si vous utilisez un nouvel outil, veuillez le mentionner dans le rapport du TP en expliquant les raisons pour lesquelles vous l'avez choisi par rapport à l'outil demandé.

Le livrable final consistera en un rapport **professionnel** portant sur les opérations et le développement de logiciels sécurisés. Veuillez éviter de répondre directement aux questions suivantes comme s'il s'agissait d'un simple travail pratique. Imaginez plutôt que vous soumettez ce rapport dans le cadre d'un projet, destiné à être lu par certain de vos collègues qui sont des développeurs, des architectes ou des gestionnaires de projet.

Chaque équipe doit soumettre un unique document contenant le **nom de l'équipe**, **les noms et les matricules des membres de l'équipe**, ainsi que toutes les **références externes** telles que les articles, les liens, la documentation et les outils.

Objectif

Ce travail pratique met principalement l'accent sur :

1. L'exécution des tests de pénétration pour garantir la qualité du logiciel en termes de sécurité.
2. L'analyse du système et l'évaluation des vulnérabilités du code de l’application
3. Implémentation dans un CI/CD

Prérequis

- Le laboratoire demande d’avoir un compte GitHub
- L’étudiant devra, pour les laboratoires, avoir des bases avec l’utilisation du système de contrôle de version Git.
- L’étudiant doit avoir une connaissance dans l’utilisation du langage de balisage Markdown.
- L’étudiant doit avoir installé Docker
- L’étudiant doit avoir installé Docker Compose
- L’étudiant doit avoir un compte étudiant Azure
- L’étudiant doit avoir un compte Github Student Developer Pack

Remarques pour le TP2:

1. **Liberté d'outil**:
    a. Vous êtes encouragé à utiliser des outils différents de ceux mentionnés dans le laboratoire ou le cours. Cependant, si vous choisissez d'utiliser un autre outil, veuillez le mentionner explicitement dans votre rapport en expliquant votre choix et la raison pour laquelle il est plus approprié que l'outil suggéré.
2. **Outils recommandés**:
   L'utilisation des outils suivants est fortement recommandée pour ce TP :
    a. **OWASP ZAP** : Pour réaliser des tests de pénétration automatisés, notamment des analyses de sécurité axées sur les vulnérabilités du Top 10 de l'OWASP.
    b. **Orchestron** : Pour générer des rapports de vulnérabilité à partir des résultats des tests effectués avec **OWASP ZAP**.
        i. Gérer les faux positifs à partir de différents outils de sécurité.
        ii. Parler le langage du développeur avec des intégrations avec Jira
        iii. Corréler et fusionner automatiquement les vulnérabilités de divers outils de sécurité
        iv. Gérer les résultats de divers outils de sécurité des applications directement à partir du pipeline CI/CD
    c. **Burp Suite** : Un autre excellent outil pour réaliser des tests de pénétration manuels et automatisés, offrant une multitude de fonctionnalités pour la détection de vulnérabilités.
    d. **Github Action**: Plateforme d'automatisation de CI/CD intégrée à GitHub.
    e. **Heroku**: Plateforme cloud permettant de déployer, gérer et faire évoluer des applications facilement.
    f. **Azure:** Plateforme cloud complète proposant des services pour le déploiement, la gestion et le suivi d'applications.
    g. **Jenkins:** Créer des pipelines CI/CD pour automatiser la construction, les tests et le déploiement d'applications.
    h. **CodeQL:** Outil d'analyse statique permettant de détecter les vulnérabilités dans le code source
    i. **Snyk.io:** Outil de gestion des vulnérabilités pour les dépendances des projets.
    j. **Dependabot :** Outil intégré à GitHub pour gérer automatiquement les mises à jour des dépendances
    k. **Renovate :** Similaire à Dependabot, il permet de gérer les mises à jour des dépendances.

Partie 1: Tests de pénétration et Analyse de la sécurité de l'application (5%)

1. **Application vulnérable**
    a. Utilisez le projet fourni sur Moodle, décompressez-le et ajoutez-le dans un dépôt GitHub.
2. Installation d'**OWASP ZAP**
    a. Téléchargez et installez **OWASP ZAP** à partir du lien suivant: (https://www.zaproxy.org/download/).
    b. Familiarisez-vous avec l'outil en consultant la documentation officielle : (https://www.zaproxy.org/docs/)
3. Exploration de l'Application Web
    a. **Lancement de ZAP** : Ouvrez OWASP ZAP et configurez-le pour explorer l'application Web vulnérable sélectionnée.
    b. **Exploration** : Naviguez sur le site Web vulnérable en soumettant des formulaires, en naviguant entre les pages, et en déclenchant des actions sur l'application via **ZAP**.
    c. **Scan d'intrusion et analyse de sécurité** : Exécutez des analyses de sécurité automatisées avec ZAP pour identifier les vulnérabilités potentielles.
        i. Concentrez-vous aux vulnérabilités de l'**OWASP Top 10** sur
            1. la navigation sur le site
            2. la soumission de formulaires
        ii. Utiliser l'image docker : *owasp/zap2docker-stable:2.12.0*
            1. */zap/zap-baseline.py -t <yourapp> -g gen.conf -x testreport.xml*
    d. **Générez** un rapport de test détaillé résumant les vulnérabilités trouvées et les risques associés. (au moins 3 vulnérabilités détectées dans les analyses de sécurité manuelles et automatiques de ZAP)
        i. **Nature des vulnérabilités** (ex : injection SQL, XSS, etc.)
        ii. **Localisation dans le code** (fichiers ou modules affectés)
        iii. **Gravité des vulnérabilités** (ex : critique, élevée, moyenne, faible)
        iv. **Risques associés** à chaque vulnérabilité (impact potentiel sur la sécurité).
4. Génération d'un Rapport de Test avec Orchestron
    a. **Téléchargez et installez Orchestron Community** en suivant les instructions fournies sur la page dédiée :
        i. **Orchestron Community** GitHub
        ii. Vous pouvez utiliser la notion des webhooks dans Orchestron
            1. Exemple : curl -H "Secret-Key: secret_key" -H "Access-Key: acces_key" -v -F file=@ http://localhost/api/webhooks/post/bbb975f6-2587-4de2-b1eb-8f65ea2ac032/
    b. **Générez** un rapport de test détaillé résumant
        i. les vulnérabilités trouvées
        ii. les risques associés.

Partie 2 : Implémentation d'une interface de CI/CD (5%)

1. Configurer les Secrets GitHub ou Jenkins
    a. Avant de créer le workflow, configurez les secrets nécessaires pour sécuriser vos informations sensibles.
        i. **API Key** : Clé API nécessaire pour l'intégration
        ii. **Username** : Associé à l'API
        iii. **Password** : Associé à l'API
        iv. **Token** : Jeton d'accès pour l'authentification
2. Configurer le pipeline CI avec Workflow **GitHub Actions** ou Jenkins
    a. Mettez en place un pipeline CI pour automatiser les tests de sécurité. **Choisir 2**
        i. OWASP ZAP avec outil **Orchestron**
        ii. SonarQube
        iii. CodeQL
        iv. Snyk.io
    b. Mettez en place un pipeline CI pour automatiser les gestions des dépendances. **Choisir 1**
        i. Dependabot
        ii. Renovate
3. Configurer la création automatique d'image vers Docker Hub pour le déploiement.
    a. Publishing Docker images
    b. Introduction to GitHub Actions
4. Déploiement sur GitHub Pages
    a. **Déployer la documentation** sur GitHub Pages à partir du dossier ./docs
    b. Ajouter votre rapport en dans le fichers ./docs
5. Configurer le pipeline CD pour le déploiement continu
    a. **Déployer** l'application en utilisant l'image de **DockerHub**
    b. Utiliser une base de données PostgreSQL connectée à votre application
        i. Heroku
        ii. Vercel
        iii. Azure

Partie 3 : Rapport (5 points)

Le livrable final est un rapport **professionnel** décrivant l'ensemble des analyses effectuées et les stratégies de mitigation mises en place.

Structure du rapport

- **Introduction**
  - Présentez l'application analysée et les outils utilisés.
- **Analyse** de sécurité
  - Résultats des tests de pénétration.
  - Description des vulnérabilités identifiées et de leur gravité.
  - Propositions de stratégies de mitigation pour chaque vulnérabilité. (maximum 3)
- **Description du pipeline d'intégration continue** (**CI**)
  - Description du **pipeline** et des **outils utilisés**.
  - Faire un **diagramme d'état ou d'activité** pour la séquence des étapes de votre pipeline
- **Description du pipeline de déploiement continu (CD)**
  - Expliquer les étapes simples du processus de déploiement.
  - Incluez un **diagramme de déploiement** pour votre application.
  - **Lien** vers votre **documentation** et votre environnement de **déploiement**
- **Conclusion**
  - Résumé des résultats et recommandations.
  - Recommandations pour améliorer la sécurité de l'application

Remarques supplémentaires :

- Le rapport peut faire référence à votre wiki ou à la documentation dans GitHub.
- Citez toutes les sources utilisées (articles, documentation d'outils, etc.) de manière appropriée.

Soumission et l'évaluation

1. Veuillez nommer votre rapport « **TP2_[nom_équipe].pdf** ».
2. Votre document sera évalué en fonction de l'exactitude, de la complétude des réponses et de la qualité de la rédaction.
3. La note individuelle de chaque membre pourrait être pondérée en fonction des évaluations par les pairs, qui seront remises en même temps que le rapport final. Les directives seront fournies dans un autre énoncé.

Annexe

1. **OWASP Zap**
   - https://www.zaproxy.org/docs/docker
   - https://www.zaproxy.org/docs/docker/webswing
   - https://hub.docker.com/r/owasp/zap2docker-stable
   - https://www.zaproxy.org/docs/desktop/start/proxies
   - https://www.zaproxy.org/docs/desktop/addons/network/options/servercertificates/#install
   - https://www.zaproxy.org/download
2. **Orchestron**
   - https://github.com/we45/orchestron-community
   - https://we45.atlassian.net/wiki/spaces/OR/pages/430866437/Installation+and+Deployment

GitHub Action Step