Mise en place d'une plateforme pour la réalisation d'un TP de cryptanalyse différentielle

Falilou NDIAYE – Éric TSAFACK

Objectif du projet

L'objectif de ce projet est de créer une plateforme en ligne permettant aux étudiants de réaliser des exercices de cryptanalyse. Sur cette plateforme, les étudiants pourront écrire du code, le tester dans un environnement sécurisé, et recevoir un retour instantané sur l'exécution de leurs programmes. Ce projet implique la création d'un éditeur de code interactif, la gestion sécurisée de l'exécution des scripts, ainsi qu'un suivi des utilisateurs et de leur progression.

Les étudiants peuvent accéder aux énoncés des TP ainsi qu'aux fichiers nécessaires à leur réalisation sur ce site. Ce TP repose sur un tutoriel de Howard M. Heys.

1. Présentation générale de la plateforme

La plateforme se compose de plusieurs sections accessibles via une page web dédiée :

- Page de Connexion et Authentification : La plateforme utilise un système Single Sign-On (SSO), permettant aux étudiants de se connecter avec leurs identifiants institutionnels (IMTBS/TSP). Une fois connectés, les étudiants accèdent à leur espace de travail pour réaliser les exercices.
- Page de travail : Après l'authentification, les étudiants accèdent à une page principale dédiée à la réalisation des exercices de cryptanalyse. Cette page contient principalement l'éditeur de code et des sections pour soumettre et tester le code.

2. Fonctionnement de la plateforme

2.1. Édition du Code avec Monaco Editor

L'éditeur de code Monaco Editor est utilisé pour permettre aux étudiants d'écrire et de tester leur code. Ce composant prend en charge plusieurs langages de programmation, et fournit une interface interactive et facile à utiliser pour la rédaction et le test des programmes.

L'étudiant rédige son code directement dans l'éditeur, choisit le langage de programmation pour l'exercice, puis peut tester son code avant de le soumettre pour évaluation.

2.2. Interaction avec l'éditeur

Chaque exercice est divisé en plusieurs étapes, où chaque étape nécessite l'écriture de fonctions spécifiques. Ces fonctions sont testées directement dans l'éditeur avant d'être soumises.

L'étudiant peut rédiger son code pour chaque fonction à compléter dans l'exercice. L'éditeur propose une interface simple et intuitive pour cela.

Une fois le code écrit, l'étudiant peut tester son programme en cliquant sur un bouton "Tester". Le code est alors envoyé au backend Flask pour être exécuté dans un environnement sécurisé.

2.3. Exécution sécurisée du code via Docker

Pour garantir que l'exécution des programmes soit sécurisée, chaque code soumis par un étudiant est exécuté dans un conteneur Docker. Cette approche permet d'isoler chaque test dans un environnement propre et sécurisé.

Lorsqu'un étudiant teste son code, celui-ci est envoyé au backend Flask, qui l'exécute dans un conteneur Docker. Ce conteneur est configuré pour exécuter le code dans un environnement isolé, garantissant ainsi la sécurité du système.

Une fois l'exécution terminée, les résultats (réussite, échec, erreurs) sont renvoyés à l'étudiant via l'interface.

2.4. Gestion des utilisateurs et suivi des progrès avec MongoDB

Les informations relatives aux utilisateurs, ainsi que leur progression dans les exercices, sont stockées dans une base de données MongoDB.

L'email de l'utilisateur, les résultats des tests et la progression dans les exercices sont enregistrés dans la base de données. Cela permet de suivre l'avancement des étudiants et d'adapter les exercices en fonction de leurs performances.

Lors de chaque connexion, l'étudiant peut voir son historique d'exercices et les scores obtenus. Le backend Flask interagit avec MongoDB pour récupérer et mettre à jour ces informations.

2.5. Interface de soumission et retour d'information

Une fois que l'étudiant a testé son code, il peut soumettre ses résultats. Le système fournit un retour d'information détaillé sur l'exécution du code.

Les résultats du test sont affichés directement sous l'éditeur de code. L'étudiant peut voir si son code a passé ou échoué les tests, ainsi que des messages d’erreur détaillés si le test échoue.

L'étudiant peut alors modifier son code en fonction du retour et tester à nouveau.

3. Structure de la page de travail

La page de travail de la plateforme est organisée de manière logique pour faciliter l'utilisation par les étudiants :

- En-tête de page : Contient des informations essentielles sur l'exercice en cours, comme son titre et les instructions générales. Il y a également un bouton "Test" pour exécuter le code et un bouton "Soumettre" pour envoyer les résultats une fois l'exercice terminé.
- Monaco Editor : L'éditeur occupe une grande partie de la page et permet à l'étudiant de rédiger son code dans un environnement interactif. Il est organisé de manière à permettre à l'étudiant de se concentrer sur l'écriture du code.

- Résultats des tests : Après chaque test, les résultats sont affichés sous l'éditeur de code. L'étudiant peut voir si son code a passé ou échoué les tests, ainsi que des détails sur les erreurs éventuelles.
- Suivi de progression : La plateforme garde une trace de la progression de chaque étudiant. Les exercices terminés et les scores obtenus sont enregistrés et consultables par l'étudiant à tout moment. Ce suivi est possible grâce à la base de données MongoDB.

4. Outils et technologies utilisés

- Flask : Un framework léger pour Python, utilisé pour créer le backend de la plateforme. Il gère les requêtes HTTP, l'exécution sécurisée du code via Docker, et l'interaction avec MongoDB.
- Monaco Editor : Un éditeur de code riche et interactif permettant aux étudiants d'écrire et de tester leur code dans un environnement dédié.
- Docker : Utilisé pour exécuter le code dans un environnement sécurisé et isolé, garantissant ainsi que l'exécution des programmes ne compromet pas la sécurité du système.
- MongoDB : Une base de données NoSQL qui permet de gérer les informations relatives aux utilisateurs, leurs résultats, et leur progression dans les exercices.

5. Synthèse de l'avancement et prochaines étapes

Travail réalisé

- Affichage de la première partie du TP avec les premières questions.
- Téléversement et exécution du code soumis par les étudiants dans un conteneur Docker.
- Développement des tests automatiques pour vérifier la validité des réponses.
- Affichage des résultats des tests sous l'éditeur de code.

Travail à réaliser et améliorations possibles

- Finalisation des autres parties du TP.
- Mise en place de l'authentification avec identifiants institutionnels.
- Gestion des statistiques des étudiants pour suivre leur progression.
- Développement d'un espace enseignant pour la gestion des TP et des étudiants.
- Génération de matériel cryptographique personnalisé et vérification de sa validité.
- Gestion multilingue (français/anglais) pour élargir l'accessibilité.
- Mise en place de mécanismes anti-triche (variations des squelettes de code et des paramètres cryptographiques).
- Extension à d'autres attaques et primitives cryptographiques (oracle de padding sur CBC, attaque de Bleichenbacher sur RSA).

Planning prévisionnel

Taches	Deadline
Finalisation des autres parties du TP	31/03/2025
Mise en place de l’authentification et gestion des utilisateurs	30/04/2025
Développement de l’espace enseignant avec les statistiques des étudiants	30/04/2025
Intégration des fonctionnalités avancées (matériel personnalisé, multilingue, anti-triche)	31/05/2025
Extension à d’autres primitives et attaques	31/05/2025

Ce planning pourra être ajusté en fonction des besoins et des avancées du projet.

Conclusion

La plateforme permet aux étudiants d’apprendre et de pratiquer la cryptanalyse dans un environnement sécurisé. Grâce à Monaco Editor, ils peuvent rédiger et tester leur code de manière interactive, tandis que Docker garantit une exécution sécurisée des programmes. Le suivi de leur progression est assuré par MongoDB, permettant ainsi une gestion flexible et efficace des informations. Le backend Flask permet de gérer l'ensemble des interactions entre l'éditeur, le système de tests et la base de données. Cette approche permet aux étudiants d’améliorer leurs compétences en cryptanalyse tout en recevant un retour immédiat sur leurs solutions.