

Travail de Bachelor

**Conception d'un nouveau serious game autour du
« Ethical Hacking »**

Extension du jeu « Shana a disparu »

Étudiante	Camille Koestli
Enseignant responsable	Sylvain Pasini
Année académique	2025-26

Yverdon-les-Bains, le 29.09.2025

Département des Technologies de l'information et de la communication (TIC)
Informatique et systèmes de communication
Sécurité informatique
Étudiante : Camille Koestli
Enseignant responsable : Sylvain Pasini

Travail de Bachelor 2025-26
Conception d'un nouveau serious game autour du « Ethical Hacking »

Résumé publiable

Ce travail est dans la continuité de « Shana a disparu », un jeu pédagogique en ligne. Ce jeu vise à initier le public au ethical hacking grâce à une narration immersive et des défis techniques. L'objectif est de concevoir et développer un nouveau scénario qui s'adresse à un tout public dans le but de renforcer la sensibilisation aux risques numériques.

Le projet comporte 7 challenges techniques de difficulté progressive qui sont intégrés dans une histoire originale. Ces défis reprennent des thématiques connues dans la cybersécurité telles **TODO**

Réalisé sur 450 heures, le projet produira un contenu technique, scénarisé et réutilisable, intégré à la plateforme existante.

Étudiante :	Date et lieu :	Signature :
Camille Koestli
Enseignant responsable :	Date et lieu :	Signature :
Sylvain Pasini

Préambule

Ce travail de Bachelor (ci-après TB) est réalisé en fin de cursus d'études, en vue de l'obtention du titre de Bachelor of Science HES-SO en Ingénierie.

En tant que travail académique, son contenu, sans préjuger de sa valeur, n'engage ni la responsabilité de l'auteur, ni celles du jury du travail de Bachelor et de l'Ecole.

Toute utilisation, même partielle, de ce TB doit être faite dans le respect du droit d'auteur.

HEIG-VD

Vincent Peiris
Chef de département TIC

Yverdon-les-Bains, le 29.09.2025

Authentification

La soussignée, Camille Koestli, atteste par la présente avoir réalisé ce travail et n'avoir utilisé aucune autre source que celles expressément mentionnées

Yverdon-les-Bains, le 29.09.2025

Camille Koestli

Remerciements

Table des matières

Préambule	4
Authentification	5
Remerciements	6
Cahier des charges	12
Contexte	12
Problématique	12
Solutions existantes	13
Approches possibles	13
Objectifs	13
Livrables	14
Planification	14
1 Introduction	15
1.1 Sensibilisation à la sécurité informatique	15
1.2 Contexte	16
1.3 Problématique	19
2 Planification	21
2.1 Décomposition des tâches	21
3 État de l'art	23
3.1 Amélioration des compétences en cybersécurité	23
3.2 Différentes classes de plateformes existantes	23
3.2.1 Cyber-ranges académiques / industriels	24
3.2.2 Plateformes ouvertes CTF	25
3.2.3 Outils et techniques de sensibilisation à la cybersécurité	26

TABLE DES MATIÈRES

3.2.4 Serious games, jeux narratifs et pédagogiques	27
3.3 Serious games comme potentielles solutions	27
4 Architecture de la plateforme <i>CyberGame</i> existante	29
4.1 Présentation générale	29
4.2 Mécanisme de jeu	29
4.2.1 Scénario 1 : « Shana a disparu »	29
4.2.2 Scénario 2 : « Sauve la Terre de l'arme galactique »	30
4.3 Techniques mobilisées	30
4.4 Analyse critique	31
4.4.1 Points forts	31
4.4.2 Axes d'amélioration	33
4.5 Architecture technique	36
4.5.1 Frontend	37
4.5.2 Cartographie des challenges	38
4.5.3 Backend	39
4.6 Infrastructure Docker compose	40
4.6.1 Services principaux	40
4.6.2 Routage et intégration Traefik	41
4.6.3 Cartographie des routes et services	42
4.6.4 API Express (Annexe-B)	42
4.6.5 WebSSH et conteneurs SSH	44
4.7 Analyse de la sécurité	45
5 Propositions de nouveaux scénarios	46
5.1 Scénario réaliste : Blackout dans le Centre Hospitalier Horizon Santé	46
5.1.1 <i>Mail Contagieux</i> : OSINT et forensic d'email	47
5.1.2 <i>Shadow VPN Portal</i> : Exploitation Web	48
5.1.3 <i>Script d'infection</i> : Reverse Engineering	49
5.1.4 <i>Coffre chiffré</i> : Cryptographie	49
5.1.5 <i>Radiographie piégée</i> : Stéganographie	50
5.2 Scénario aventurier : Opération « CipherFox » - Infiltration	51
5.2.1 <i>Hotspot Mirage</i> : OSINT et Cryptographie	52
5.2.2 <i>Admin Bypass</i> : Web Exploitation	53
5.2.3 <i>Micro-Patch</i> : Reverse Engineering	53
5.2.4 <i>SecureNote Cipher</i> : Cryptographie	54
5.2.5 <i>DNS Drip</i> : Forensique réseau	55
5.3 Scénario science-fiction : Fuite de l'Acheron	56

5.3.1 <i>HashLock</i> : Cryptographie	57
5.3.2 <i>Portail Tech</i> : Exploitation Web	57
5.3.3 <i>Drone Patch</i> : Reverse Engineering	59
5.3.4 <i>Service Secret</i> : Enum système / Forensic	59
5.3.5 <i>Plan Secret</i> : Stéganographie	60
5.4 Retour d'expertise	61
6 Scénario définitif et liste des challenges détaillés	62
6.1 Challenge 1 <i>Mail Contagieux</i> : OSINT et forensic email	66
6.1.1 Description	66
6.1.2 Tools	67
6.2 Challenge 2 <i>Portail Frauduleux</i> : Exploitation Web (SQL)	68
6.2.1 Description	68
6.2.2 Tools	69
6.3 Challenge 3 <i>Partage Oublié</i> : Mauvaise configuration d'accès	70
6.3.1 Description	70
6.3.2 Tools	71
6.4 Challenge 4 <i>Clé cachée</i> : Cryptographie et métadonnées	72
6.4.1 Description	72
6.4.2 Tools	73
6.5 Challenge 5 <i>Script Mystère</i> : Reverse Engineering	74
6.5.1 Description	74
6.5.2 Tools	75
6.6 Challenge 6 <i>Cookie Admin</i> : Mauvaise gestion des sessions	76
6.6.1 Description	76
6.6.2 Tools	77
6.7 Challenge 7 <i>Blocage ciblé</i> : Défense et journalisation	78
6.7.1 Description	78
6.7.2 Tools	79
7 Implémentation des challenges	80
7.1 Architecture générale	80
7.2 Frontend	80
7.2.1 Challenge 1	80
7.2.2 Challenge 2	82
7.2.3 Challenge 3	83
7.2.4 Challenge 4	85
7.2.5 Challenge 5	86

TABLE DES MATIÈRES

7.2.6 Challenge 6	87
7.2.7 Challenge 7	88
7.3 Back-end	91
7.3.1 Challenge 1	91
7.3.2 Challenge 2	91
7.3.3 Challenge 3	93
7.3.4 Challenge 4	95
7.3.5 Challenge 5	97
7.3.6 Challenge 6	97
7.3.7 Challenge 7	101
7.4 Intégration sur le site web	102
7.4.1 Initialisation des flags côté serveur	102
7.4.2 Ajout du mini-site de jeu « Blackout »	102
7.4.3 Raccordement dans la page d'accueil, routage et configuration des flags <code>.env</code> / <code>.env.prod</code>	105
7.5 Améliorations de l'implémentation et future correction	107
8 Tests	108
8.1 Tests unitaires	108
8.1.1 Configuration de l'environnement de test	108
8.1.2 Détails des tests par challenge	108
8.2 Tests utilisateurs	110
8.2.1 Protocole de test	110
8.3 Résultats et bilan des tests	112
8.3.1 Performance par niveau de compétence	112
8.3.2 Points forts identifiés	113
8.3.3 Points faibles et pistes d'amélioration	113
9 Conclusion	115
Glossaire	116
Bibliographie	117
Outils utilisés	121
Journal de travail	122

Annexes	126
-A Fichier JSON de configuration	126
-B API Express (<code>index.js</code>)	128
-C Modèles Mongoose (<code>db.js</code>)	139
-D Base MySQL (<code>init.sql</code>)	141
-E Docker Compose (<code>docker-compose.yml</code>)	143
-F Implémentation du jeu « Blackout » (<code>blackoutmain.js</code>)	146
-G Implémentation du bot pour le challenge 6 de 2025 (<code>bot.js</code>)	159
-H Présentation des challenges (ancienne version des défis)	172

Cahier des charges

Contexte

CyberGame est une plateforme de serious game développée initialement par le pôle Y-Security de la HEIG-VD. Le pôle Y-Security est reconnu comme un acteur majeur en cybersécurité en Suisse romande. Il a pour mission de former, sensibiliser et accompagner différents publics autour des enjeux de sécurité informatique, grâce à la recherche appliquée, la formation et la mise en place d'outils innovants.

La plateforme CyberGame vise à rendre l'apprentissage de la cybersécurité ludique et accessible à tous, à travers des scénarios interactifs et progressifs. Le jeu « Shana a disparu » est un exemple phare : il propose une initiation au ethical hacking, combinant narration immersive et challenges techniques pour faire découvrir les bases de la cybersécurité.

Ce jeu a rencontré un grand succès auprès d'un large public.

Problématique

Le succès de « Shana a disparu », créé en 2020, a conduit de nombreuses personnes à le terminer entièrement. Un second scénario, Galac game, a été mis en place en 2021 mais a remporté un plus faible succès.

Le public étant de plus en plus curieux et averti sur ce sujet, il devient nécessaire de développer un nouveau scénario afin de répondre à la demande notamment en proposant une histoire qui techniquement amène le participant à un plus haut niveau de compétences.

L'objectif est donc d'intégrer des défis techniquement plus avancés, tout en conservant l'approche narrative immersive qui fait l'intérêt et l'originalité du « serious game ».

Cette nouvelle histoire s'adressera donc à des participant·e·s ayant résolu le premier niveau scénario (Shana) ou ayant quelques connaissances de base en sécurité informatique.

Solutions existantes

Il existe aussi d'autres solutions similaires dans le domaine de l'ethical hacking, mais plutôt sous la forme de Capture The Flag (CTF) comme « Root Me », « Hack the Box », « TryHackMe », ...; des cyber-ranges qui sont plutôt destinés à des expert·e·s en cybersécurité ; ou encore des formations en ligne comme « SoSafe » qui proposent des cours et des exercices pratiques sur la cybersécurité sans forcément intégrer d'histoire narrative et immersive.

Ces solutions montrent une augmentation de l'intérêt général pour la cybersécurité. Elles utilisent des approches ludiques mais peu combinent une narration et une progression techniques comme le fait « Shana a disparu ».

Approches possibles

Pour proposer une nouvelle expérience qui s'adresse à tout le monde tout en permettant de sensibiliser mais aussi de rester ludique, plusieurs options peuvent être envisagées :

- La première option serait de développer une extension directe du scénario existant avec de nouveaux challenges plus techniques.
- Alors que la deuxième serait de créer un nouveau jeu totalement indépendant avec un nouveau scénario, tout en restant dans la même idée que le jeu précédent.

L'option choisie est de créer un nouveau scénario qui s'adresse à tout le monde. Ce scénario doit être accessible aux débutant·e·s tout en proposant des défis plus complexes pour les utilisateur·trice·s plus expérimenté·e·s. Il doit également intégrer des éléments narratifs immersifs pour maintenir l'intérêt et la motivation des joueur·euse·s.

Objectifs

Le cahier des charges va permettre d'encadrer la conception d'un scénario immersif dans le domaine de la cybersécurité. L'objectif sera de produire une nouvelle expérience ludique tout en intégrant une approche de sensibilisation.

- Concevoir un nouveau scénario :
 - Créer une histoire captivante, qui peut être une suite de Shana ou une intrigue totalement nouvelle.
 - Proposer des niveaux plus complexes que les scénarios existants.
 - Inclure 5 à 10 challenges de difficultés progressives.
 - Imaginer les épreuves en réfléchissant au côté sensibilisation et notamment aux messages que le participant·e en tirera.
 - Introduire les nouveaux concepts techniques et pédagogiques correspondants.
- Thématisques techniques :

- Couvrir plusieurs aspects de la cybersécurité comme l'exploitation web, escalade de privilèges, reverse engineering, forensic, etc.
- Intégrer un robot interactif pour simuler le comportement d'utilisateur·trice·s vulnérables (ex. clics sur une XSS).
- Intégrer tous les challenges dans une narration immersive et cohérente, fidèle à l'esprit du projet.
- Développer le nouveau serious game :
 - Il doit être intégré dans la plateforme *Cybergame* existante, tant sur la forme, que sur le contenu des technologies utilisées.
 - Inclure le scénario complet, les étapes du jeu, les mécaniques interactives, ainsi que les apports techniques et pédagogiques nécessaires.
 - Gérer les parties backend nécessaires.
 - Garantir la sécurité de l'infrastructure et du contenu.
- Réaliser des tests utilisateur·trice·s et appliquer les correctifs nécessaires pour assurer une expérience optimale.

Livrables

Les livrables seront les suivants :

- Plateforme *Cybergame* mise à jour, incluant l'ensemble du nouveau scénario opérationnel.
- Un rapport complet, comprenant :
 - Des propositions de scénarios, avec motivation du scénario retenu.
 - La documentation détaillée du scénario retenu, incluant la liste complète des challenges.
 - La documentation de la plateforme *Cybergame*, incluant la description de l'existant et des évolutions apportées, ainsi que l'explication et justification des choix techniques.
 - Une analyse de la sécurité de la plateforme.
 - Les tests fonctionnels réalisés.
 - Les tests utilisateur·trice·s réalisés : méthodologie, résultats, retours collectés, et correctifs appliqués.

Planification

Le travail se déroule entre le 7 juillet et le 10 octobre 2025, pour un total de 450h :

- Du 7 juillet au 15 septembre : travail à temps plein (45h/semaine).
- Du 16 septembre au 10 octobre : travail à temps partiel (12–13h/semaine).

Le rendu intermédiaire est prévu pour la date du 31 juillet 2025, le rendu final est fixé au 10 octobre 2025, enfin, la défense devra être fixée après le 13 février 2026.

1 Introduction

1.1 Sensibilisation à la sécurité informatique

De nos jours, la digitalisation croissante de notre quotidien, que ce soit au niveau administratif, paiements, télé-travail, ... expose les utilisateur·trice·s à de nombreux risques en matière de sécurité informatique. Dans l'article de *The Digital Decade* les Européen·ne·s sont de plus en plus préoccupé·e·s par la sécurité de leurs données personnelles et de leur vie privée en ligne et 79% estiment que « l'amélioration de la cybersécurité et de la protection des données [...] » est indispensable pour pouvoir profiter sans souci des services numériques (European Commission. Directorate General for Communications Networks, Content and Technology. 2024).

De plus, l'étude réalisée par *Wahl* montre que 52% des répondants Européen·ne·s estiment ne pas être en mesure de se protéger suffisamment contre la cybercriminalité. Mais, en contrepartie, 52% des personnes interrogées déclarent qu'il y a une augmentation de la sensibilisation à la cybersécurité (Wahl 2020). En complément, le rapport publié par l'ENISA (Agence européenne de cybersécurité) met en évidence une baisse de la confiance des citoyen·ne·s dans leurs capacités à se protéger contre les menaces et met en évidence « une faible connaissance des mécanismes de signalement des cybercrimes » (European Union Agency for Cybersecurity (ENISA) 2024).

Le risque principal que les utilisateur·trice·s courrent, est d'être la cible d'attaques, telles que le phishing, les ransomwares ou les logiciels malveillants. Aujourd'hui, les cybercriminels vont plus loin que l'utilisation de failles techniques mais tirent aussi profit de ses failles humaines à travers du social engineering par des attaques personnalisées utilisant l'IA (Spys, Solovei 2025). En 2027, les chercheurs estiment qu'il y aura une augmentation de 17% des attaques utilisant l'IA (Spys, Solovei 2025). Le nombre de courriels frauduleux envoyés par jour dépassent les 3,4 milliards, ce qui représente 36 % des failles de sécurité et 94 % des infections par maliciel (Spys, Solovei 2025). Les campagnes de phishing se sont intensifiées, avec une augmentation de 57,5 % des attaques par ransomwares entre novembre 2024 et février 2025 (KnowBe4 2025).

De plus, dans le monde entrepreneurial, il y a une forte croissance de la demande concernant les compétences en cybersécurité. En 2024, plus de deux tiers des professionnels européens déclaraient

un environnement de menaces « plus stressant que jamais » et 61 % signalait un sous-effectif dans leurs équipes (Santini [sans date]).

Le rapport mondial ISC2 2024 confirme cette augmentation : 67 % des organisations estiment ne pas disposer des compétences nécessaires pour atteindre les objectifs de sécurité (*2024 ISC2 Cybersecurity Workforce Study* [sans date]).

Selon l'étude de Fortinet 2024, 58 % des incidents majeurs seraient directement liés à un manque de savoir-faire technique ou de formation du personnel (Fortinet 2024a).

Face à ce manque, il est essentiel de former rapidement les jeunes mais aussi le grand public. La recherche montre l'efficacité des approches plus ludiques pour des apprentissages. Une approche intéressante est l'utilisation de serious games en lien avec la cybersécurité. Ce sont des jeux conçus pour sensibiliser et former les utilisateur·trice·s à la cybersécurité (Hill, Fanuel, Yuan 2020). Ces jeux permettent d'apprendre de manière interactive et motivante, tout en simulant des situations réelles de cybersécurité. Une revue conclue même que les serious games sont un moyen efficace pour sensibiliser les utilisateur·trice·s dépourvu·e·s de bagage technique (Ng, Hasan 2025). Les serious games sont une méthode reconnue pour permettre d'engager, de motiver et de favoriser cet apprentissage. Des travaux de recherche montrent qu'ils permettent de découvrir beaucoup de techniques et de notions (cryptographie, réseau, scripts, attaque Web) tout en offrant un environnement sans risque. L'utilisateur·trice peut expérimenter et apprendre de ses erreurs (Hill, Fanuel, Yuan 2020). C'est là que le pôle Y-Security de la HEIG-VD intervient et décide de se pencher sur les serious games pour sensibiliser et former le grand public à la cybersécurité.

1.2 Contexte

Depuis plus de vingt ans, la HEIG-VD est un acteur majeur de la cybersécurité en Suisse romande et en Europe. Le pôle Y-Security de la HEIG-VD est reconnu pour sa recherche appliquée, sa formation et son accompagnement dans le domaine de la sécurité informatique. Il regroupe une douzaine d'expert·e·s et un réseau industriel suisse et européen, ce qui en fait un véritable « écosystème » pour la formation et l'innovation en Suisse romande (*Y-Security - HEIG-VD* [sans date]).

Pour rendre plus accessible la pratique du hacking éthique, le pôle a donc créé une plateforme *Cyber-game* et décide de se pencher sur les serious game. L'objectif de ces serious games est de sensibiliser et former les utilisateur·trice·s aux bases de la cybersécurité, en leur permettant de découvrir les techniques d'ethical hacking grâce à une approche narrative immersive. Ces jeux sont destinés à un large public, allant des débutant·e·s aux personnes ayant déjà des connaissances en sécurité informatique. Il propose actuellement deux scénarios en ligne sur cette thématique (*Initiation Au Ethical Hacking* [sans date]):

- « Shana a disparu » (2020) : une enquête qui vise à retrouver Shana et qui initie les débutant·e·s aux bases du piratage éthique.

- « Sauve la Terre de l'arme galactique » (2021) : une mission interplanétaire qui a pour objectif se récupérer des plans d'une arme galactique.

« Shana a disparu » nous raconte l'histoire d'une jeune femme qui a disparu et dont il faut retrouver la trace. Le joueur·euse doit résoudre des énigmes et des défis techniques pour progresser dans l'histoire et découvrir ce qui est arrivé à Shana. Le jeu est conçu pour être accessible aux débutant·e·s, tout en offrant des défis intéressants pour les joueur·euse·s plus expérimenté·e·s. L'interface se présente sous la forme d'un site web interactif, où les joueur·euse·s peuvent naviguer entre différentes pages, résoudre des énigmes et interagir avec des éléments du jeu.



Fig. 1. – « Shana a disparu » - Interface du jeu (*Shana a Disparu. Retrouve-la !* [sans date])

La Figure 1 permet de voir la plateforme *Cybergame* et la construction de l'interface pour le joueur·euse. L'utilisateur·trice peut naviguer d'un défi à l'autre grâce à la barre de navigation en haut de la page. Chaque challenge est présenté à travers une page dédiée avec une description du défi, des indices ainsi qu'un bouton afin de démarrer le challenge. Une fois le défi lancé, il peut explorer le site web,

INTRODUCTION

inspecter les éléments, analyser le code source, ... Une fois le défi résolu, il devra remplir le champ **Valider l'étape !** avec la réponse correcte pour passer à l'étape suivante.

Pour aider le joueur·euse, dans la pop-up de description du défi, il y a un bouton **Indice** qui permet d'afficher un indice pour l'aider à résoudre le défi. De plus, sur le site web, nous retrouvons aussi une boîte à outils.



Qu'est-ce que le code source ?

Une page web est un document HTML interprété par un navigateur. Un document HTML est organisé de façon à ce qu'un navigateur web (p.ex : Chrome, Firefox ou Opera) soit capable de l'interpréter afin de présenter à l'utilisateur une page web agréable et non pas du texte brut.

Le document est sous la forme d'un code HTML qui est créé par le serveur web avec lequel vous parlez. Le serveur web à qui vous parlez est identifié par une URL (www.siteweb.com). Comme le serveur est au courant de ce que vous désirez voir, il est capable de vous fournir une page sur mesure avec, par exemple, votre identifiant et les informations de votre compte.

Comment accéder à un code source ?

Fig. 2. – Boîte à outils (*Informations Sur Les Outils et Méthodes Utilisées !* [sans date])

Cette Figure 2 permet de mettre en évidence la boîte à outils présente sur le site web. Elle aide les joueur·euse·s à trouver les outils nécessaires pour résoudre les défis. Par exemple, un outil pour inspecter le code source, un autre pour analyser les requêtes HTTP, ou comment écrire un petit script en Python.

En ce qui concerne le second scénario, « Sauve la Terre de l'arme galactique », il s'agit d'une mission interplanétaire où le joueur·euse doit récupérer des plans d'une arme galactique. Il reprend les mêmes

bases que le premier scénario, comme le montre la Figure 3, avec la même interface et les mêmes mécaniques de jeu. Cependant, il est moins populaire que le premier scénario, car il ne propose pas de nouveaux défis techniques et reste dans la même idée que le premier scénario.

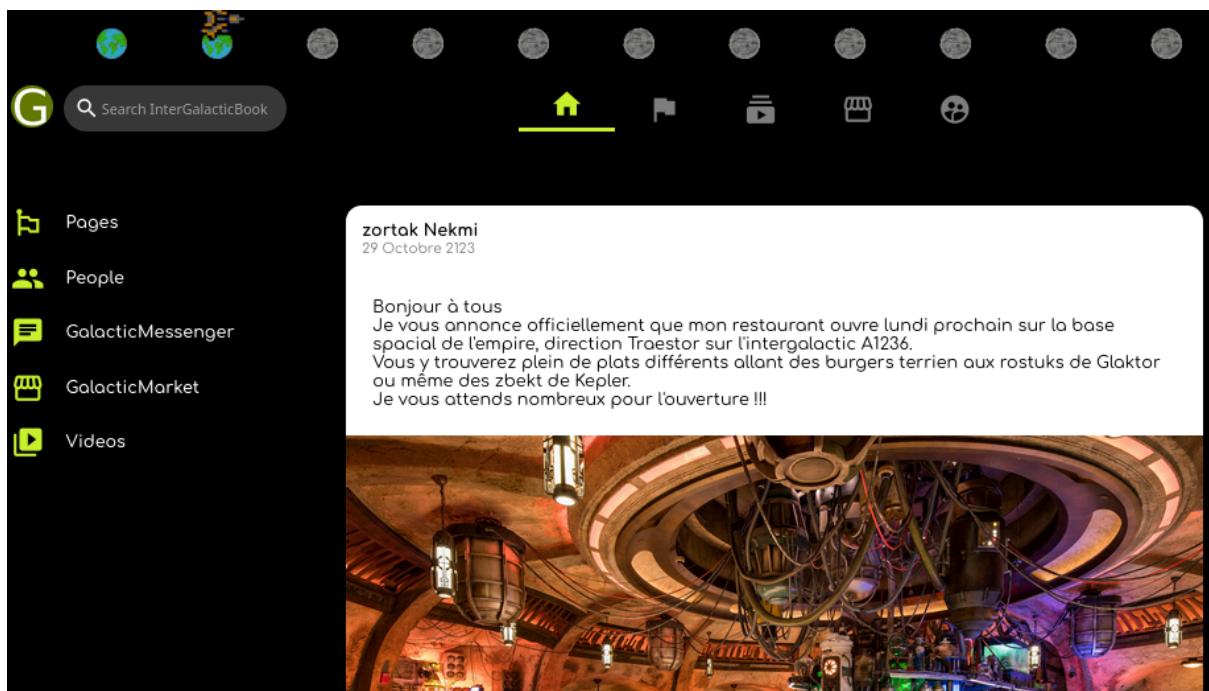


Fig. 3. – « Sauve la Terre de l’arme galactique » - Interface du jeu (*Sauve La Terre de l’arme Galactique !* [sans date])

Grâce à leur narration immersive et leurs défis, ces jeux, en particulier « Shana a disparu » ont rencontré un grand succès auprès du public. Cependant, la majorité des participant·e·s les ont déjà terminés.

1.3 Problématique

Le succès de « Shana a disparu » a atteint ses limites, beaucoup de personnes ont actuellement terminé ce jeu, et la plupart des joueur·euse·s maîtrisent déjà certaines bases, d’autant plus que le jeu « Sauve la Terre de l’arme galactique », reprend les mêmes bases que le premier scénario. Afin de pousser les utilisateur·trice·s à continuer à pratiquer et approfondir ces sujets, il est essentiel d’élargir et de réaliser des challenges plus complexes. De plus, le monde du travail demande des profils capables de gérer des menaces plus sophistiquées. La plateforme doit donc évoluer afin de proposer une nouvelle histoire immersive plus difficile tout en maintenant la motivation et en introduisant des nouveaux défis plus techniques de niveau intermédiaire à avancé.

INTRODUCTION

La question est donc : « *Comment créer une nouvelle histoire immersive et prenante qui intègre plusieurs techniques d'ethical hacking, afin de sensibiliser et former les utilisateur-trice-s de tous les niveaux ?* »

L'objectif de ce travail de Bachelor est de répondre à cette question en développant un nouveau scénario pour la plateforme *Cybergame*. Ce scénario doit être accessible aux débutant·e·s tout en proposant des défis plus complexes pour les utilisateur·trice·s plus expérimenté·e·s. Il doit également intégrer des éléments narratifs immersifs pour maintenir l'intérêt et la motivation des joueur·euse·s.

2 Planification

2.1 Décomposition des tâches

1. Analyse du scénario existant : *07.07.2025 – 09.07.2025*
 - Étudier les mécaniques de jeu et les défis utilisés dans « Shana a disparu ».
 - Identifier les technologies utilisées et les types de challenges (web, forensic, ...).
 - Évaluer les points positifs et les points à améliorer du scénario actuel.
 - Étudier l'architecture de la plateforme *Cybergame*
2. Recherche et écriture du scénario : *10.07.2025 – 23.07.2025*
 - S'inspirer de CTF, serious games et projets similaires pour la structure et le contenu des défis.
 - Identifier les outils et environnements de développement.
 - Identifier les bonnes méthodes pédagogiques adaptées à la sensibilisation à la cybersécurité à travers un jeu interactif.
 - Élaborer plusieurs scénarios, puis détailler celui qui a été retenu.
3. Conception et développement des challenges : *24.07.2025 – 03.09.2025*
 - Définir les thématiques techniques abordées et les attaques à réaliser (XSS, reverse engineering, stéganographie, ...).
 - Concevoir entre 5 et 10 challenges.
 - Développer les services ou environnements nécessaires.
 - Ajouter un bot interactif pour simuler certaines interactions ou attaques.
 - S'assurer de la clarté des consignes et de la logique de chaque challenge.
4. Intégration dans la plateforme *Cybergame* : *04.09.2025 – 09.09.2025*
 - Adapter les contenus au format de *Cybergame*.
5. Tests et validation : *10.09.2025 – 19.09.2025*
 - Réaliser des tests unitaires pour chaque challenge.
 - Réaliser des tests utilisateur·trice·s et faire tester les défis par d'autres personnes pour ajuster la difficulté.
 - Corriger les éventuels bugs ou incohérences.
6. Documentation technique et pédagogique : *20.09.2025 – 08.10.2025*
 - Documenter chaque challenge : objectif, compétences visées, indices, solutions, pièges courants.

PLANIFICATION

- Rédiger la documentation du scénario.
- Décrire les choix techniques et les modifications apportées à la plateforme.
- Documenter les tests.

3 État de l'art

Ce chapitre a pour objectif d'explorer, comprendre et poser les différentes bases concernant les divers sujets abordés dans ce travail de Bachelor. Il s'agit de comprendre les enjeux, les défis et les solutions existantes dans le domaine de la cybersécurité, en particulier à travers l'utilisation de serious games et d'approches pédagogiques innovantes.

3.1 Amélioration des compétences en cybersécurité

Différentes enquêtes et études montrent que la sensibilisation et la formation en cybersécurité sont essentielles pour réduire les risques liés aux cyberattaques. Les utilisateur·trice·s, qu'ils soient novices ou expérimenté·e·s, doivent être conscient·e·s des menaces potentielles et des bonnes pratiques à adopter. Des enquêtes annuelles confirment un manque d'environ 4,8 millions de professionnels de la cybersécurité dans le monde, ce qui souligne l'importance de former et de sensibiliser un plus grand nombre de personnes (*2024 ISC2 Cybersecurity Workforce Study* [sans date]). De plus, selon 64% des professionnels dans le domaine de la sécurité informatique, les déficits de compétences en cybersécurité ont un impact négatif plus important qu'une pénurie de personnel. *Fortinet* montre que dans 70% des entreprises, des incidents graves de cybersécurité sont dus à des erreurs humaines, ce qui met encore plus en avant l'importance de la formation et de la sensibilisation (Fortinet 2024b).

3.2 Différentes classes de plateformes existantes

Actuellement, il existe plusieurs plateformes et écosystèmes qui proposent des environnements d'apprentissage en cybersécurité. Ces plateformes offrent des défis variés, allant de la simple sensibilisation à des scénarios plus complexes nécessitant des compétences techniques avancées, comme les cyber-ranges. Parmi les outils les plus connues, nous retrouvons beaucoup de CTF comme « Hack The Box » (*Hack The Box: The #1 Cybersecurity Performance Center* [sans date]), « TryHackMe » (*TryHackMe/Simple CTF* [sans date]) et « RootMePlateforme » (*Root Me : Plateforme d'apprentissage Dédiée Au Hacking et à La Sécurité de l'Information* [sans date]). Ces plateformes permettent aux utilisateur·trice·s de pratiquer leurs compétences dans un environnement sécurisé et contrôlé. Les autres outils et techniques sont moins connus, mais sont tout aussi importants pour la sensibilisation et la formation

en cybersécurité. Parmi eux, nous pouvons citer les cyber-ranges académiques et industriels, les plateformes ouvertes de type CTF, les outils et techniques de sensibilisation à la cybersécurité, ainsi que les serious games, jeux narratifs et pédagogiques.

3.2.1 Cyber-ranges académiques / industriels

Les cyber-ranges sont des environnements simulés qui permettent aux utilisateur·trice·s de pratiquer leurs compétences en cybersécurité dans un cadre réaliste (*What Is Cyber Range · Definition · DIATEAM [sans date]*). Ils sont souvent utilisés par les institutions académiques et les entreprises pour former leurs employé.e.s. Ces environnements offrent une expérience immersive, permettant aux utilisateur·trice·s de travailler sur des scénarios réels, de développer leurs compétences en résolution de problèmes mais aussi de pratiquer divers types d'attaques. Ces cyber-ranges vont permettre de tester la mise en place des défenses et d'identifier des vulnérabilités potentielles sans compromettre la sécurité de vrais systèmes (*Qu'est-ce qu'un cyber range ? / IBM [sans date]*). Ces plateformes sont surtout utilisées pour des formations avancées et des exercices de simulations pour la Blue Team et ne sont donc pas accessibles à tous. La Figure 4 permet de comprendre le fonctionnement d'un cyber-range et son organisation.



Fig. 4. – Schéma d'un cyber-range (*What Is Cyber Range · Definition · DIATEAM [sans date]*)

« Shana a disparu » n'a pas du tout cet objectif. En effet la formation et la sensibilisation est son objectif principal et permet de découvrir les techniques d'ethical hacking grâce à une approche narrative immersive. Les cyber-ranges sont donc plus adaptés pour des formations avancées et des exercices de simulations pour la Blue Team, tandis que « Shana a disparu » est destiné à un large public, allant des débutant·e·s aux personnes ayant déjà des connaissances en sécurité informatique.

3.2.2 Plateformes ouvertes CTF

Les plateformes ouvertes de type CTF (Capture The Flag) sont des environnements d'apprentissages interactifs qui permettent aux utilisateur·trice·s de résoudre des défis de cybersécurité. Ces plateformes sont souvent utilisées dans le cadre de compétitions, mais elles peuvent également servir de ressources pédagogiques pour les étudiants et les professionnels souhaitant améliorer leurs compétences en cybersécurité. Elles offrent une variété de défis, allant de la simple cryptographie à des scénarios plus complexes impliquant l'exploitation de vulnérabilités.



Fig. 5. – Page des challenges de RootMePlateforme (*Root Me : Plateforme d'apprentissage Dediée Au Hacking et à La Sécurité de l'Information* [sans date])

A la différence des serious games, ces plateformes sont souvent plus techniques et nécessitent des compétences avancées en cybersécurité. Elles se présentent souvent sous la forme de compétitions où les participant·e·s tentent de résoudre des défis techniques pour obtenir une drapeau (flag) cachés dans des systèmes vulnérables (*CTF Hacking : guide ultime pour devenir un expert en Capture The Flag* [sans date]). Elles sont donc moins accessibles aux débutant·e·s et peuvent être compliquées pour ceux qui n'ont pas de formation préalable en sécurité informatique. Elles ont donc moins vocation à sensibiliser à la cybersécurité. De plus, chaque challenge est souvent isolé par catégorie et ne propose pas une histoire narrative immersive, comme le montre la Figure 5. Ces plateformes sont donc plus adaptées aux utilisateur·trice·s ayant déjà des connaissances en sécurité informatique et souhaitant approfondir leurs compétences techniques.

3.2.3 Outils et techniques de sensibilisation à la cybersécurité

Il existe de nombreux outils et techniques pour sensibiliser les utilisateur·trice·s à la cybersécurité. Parmi les plus courants, on trouve les formations en ligne, les ateliers pratiques et les simulations d'attaques, comme par exemple « SoSafe » (*Sensibilisation à la cybersécurité et gestion du risque humain* 2022). Ces outils permettent aux utilisateur·trice·s de comprendre les menaces potentielles et

d'apprendre à se protéger.

La différence que nous pouvons observer avec le jeu « Shana a disparu » est que ces outils sont souvent plus théoriques et moins immersifs. Ils ne permettent pas aux utilisateur·trice·s de s'engager activement dans le processus d'apprentissage, ce qui peut limiter leur efficacité. En revanche, les serious games offrent une approche plus interactive et engageante, permettant aux utilisateur·trice·s de pratiquer leurs compétences dans un environnement sécurisé. De plus, généralement ces outils sont souvent destinés à des entreprises ou des organisations et sont donc payants, ce qui limite leur accessibilité pour le grand public.

3.2.4 Serious games, jeux narratifs et pédagogiques

Les serious games, ou jeux sérieux, sont des outils pédagogiques qui utilisent des éléments de jeu pour enseigner des concepts complexes. Ils sont particulièrement efficaces dans le domaine de la cybersécurité, car ils permettent aux utilisateur·trice·s de s'engager activement dans le processus d'apprentissage. Les jeux narratifs et pédagogiques offrent une approche immersive qui peut aider à renforcer la compréhension des concepts de cybersécurité et à améliorer l'acquisition des connaissances. Ces jeux peuvent simuler des scénarios réels, permettant aux utilisateur·trice·s de prendre des décisions et de voir les conséquences de leurs actions dans un environnement sécurisé.

Il existe très peu de ces serious games en français, la plupart sont en anglais. De plus, la plupart des serious games sont payants et ne sont pas accessibles à tous, comme par exemple sur le site « Urban-Gaming » (*Serious game sécurité informatique: le jeu Urban Gaming [sans date]*), « Shirudo » (*Shirudo / Serious Game Multilingue En Cybersécurité [sans date]*), ou encore « Cyber Wargame » (*Cyber Wargame : Des Serious Games sur la Cybersécurité [sans date]*). Ces entreprises proposent différents services, notamment des formations et des ateliers utilisant les serious games. Contrairement à « Shana a disparu », ils sont payants et ne sont pas accessibles à tous, ils sont souvent destinés à des entreprises ou des organisations.

3.3 Serious games comme potentielle solution

Définis par *Clark Abt* (Abt 1970), puis revue par *Mike Zyda* (Zyda 2005), les serious games sont « *un concours intellectuel, joué sur ordinateur selon des règles spécifiques, qui utilise le divertissement pour atteindre des objectifs de formation, d'éducation, de santé, de politique publique ou de communication stratégique.* » (Zyda, From Visual Simulation to Virtual Reality to Games 2005, p.26) (citation traduite). Les serious games sont de plus en plus utilisés comme solution pour sensibiliser et former les utilisateur·trice·s à la cybersécurité. Ils offrent une approche interactive et engageante qui leur permet de pratiquer leurs compétences dans un environnement sécurisé. De plus, les serious games peuvent être adaptés à différents niveaux de compétences, ce qui les rend accessibles à un large public. Cependant, selon l'étude de *Chiu Yeong Ng et Mohammad Khatim Bin Hasan*, la plupart des serious games existants ont été développés pour des utilisateurs avancés et ne sont pas adaptés aux débutants

ÉTAT DE L'ART

(Ng, Hasan 2025). De plus, ils ont relevés que la majorité de ces jeux se focalisent plus sur des aspects techniques, tels que le piratage, les architectures réseaux, ..., que sur des aspects humains.

4 Architecture de la plateforme *Cyber-Game* existante

Ce chapitre présente l'architecture technique de la plateforme *CyberGame*, en détaillant le frontend et backend, ainsi que les mécanismes de jeu. Il est important de souligner qu'il s'agit d'une analyse de la plateforme de 2020 avant sa restructuration qui a eu lieu en 2025. Cette analyse porte donc sur l'état initial du site, avant l'ajout de nouvelles fonctionnalités, la refonte du design ou l'amélioration de l'expérience utilisateur.

4.1 Présentation générale

Le site web est une plateforme pédagogique créée par le pôle Y-Sécurité de la HEIG-VD. Il a pour objectif d'introduire au ethical hacking et propose actuellement deux scénarios interactifs. La plateforme est donc conçue avec une page d'accueil (*Initiation Au Ethical Hacking* [sans date]) qui présente le cadre général. Le premier jeu « Shana a disparu » (*Shana a Disparu. Retrouve-la !* [sans date]) ainsi qu'un autre scénario « Sauve la Terre de l'arme galactique » (*Sauve La Terre de l'arme Galactique !* [sans date]) se trouvent sur la plateforme. Pour aider les joueur·euse·s à avancer dans les différents challenges, une boîte à outils et un petit IDE Python ont été développés (*Initiation Au Ethical Hacking* [sans date]).

4.2 Mécanisme de jeu

La plateforme CyberGame propose deux parcours structurés sous forme d'histoire progressive qui mettent en œuvre des techniques clés du hacking éthiques. Chacun propose une enquête avec un scénario dont les étapes doivent être validées dans l'ordre afin de pouvoir progresser dans le déroulement de l'enquête.

4.2.1 Scénario 1 : « Shana a disparu »

Le scénario « Shana a disparu » (*Shana a Disparu. Retrouve-la !* [sans date]) a pour objectif d'amener le joueur·euse dans une enquête de neuf challenges successifs qui miment la progression d'une investigation numérique. Pour nous aider à résoudre ces challenges, une petite boîte à outil avec

des explications est fournie (*Initiation Au Ethical Hacking* [sans date]). L'histoire commence par la reconstruction du mot de passe Windows de Shana à partir des informations qui se trouvent sur le profil Instagram de la victime. Le challenge suivant est l'exploration de l'historique de navigation pour extraire ses derniers sites consultés. Une fois le site trouvé, un lien caché en texte invisible est inséré sur la page. Le défi suivant consiste à inspecter le code source pour trouver des informations qui vont nous permettre de progresser. Une fois l'information trouvée, le jeu redirige le joueur·euse vers une page où la manipulation de cookie est nécessaire : il faut modifier la valeur d'une variable de session pour débloquer la page cachée. S'ensuit un chiffrement de César qu'il faut renverser pour découvrir une date clé, puis l'altération manuelle de la fin d'une URL afin d'accéder à un répertoire non indexé. Le challenge suivant demande une injection SQL qui va permettre de contourner l'authentification et d'obtenir de nouvelles informations, qui se confirment grâce à l'extraction des coordonnées GPS dissimulées dans les métadonnées EXIF d'une photo. Chaque résolution de challenge permet de dévoiler un indice indispensable au suivant, illustrant la chaîne « collecte – exploitation – preuve » qui est l'approche « typique » d'un hacker éthique.

4.2.2 Scénario 2 : « Sauve la Terre de l'arme galactique »

Le second scénario que nous retrouvons sur la plateforme « Sauve la Terre de l'arme galactique » (*Initiation Au Ethical Hacking* [sans date]), utilise les mêmes principes mais dans un univers de science-fiction. Le joueur·euse est plongé·e dans une enquête afin de retrouver les plans d'une arme galactique et ainsi sauver le monde. Dans un premier temps, le joueur·euse exploite la barre de recherche d'un réseau fictif pour obtenir des fragments de conversation. Ensuite, le participant·e va utiliser l'ingénierie sociale pour retrouver des réponses de sécurité, imprudemment divulguées en ligne, ce qui va permettre de retrouver le mot de passe et ainsi accéder au profil. Des challenges similaires se retrouvent dans les deux jeux comme la manipulation des cookies, l'ajustement d'un paramètre `GET` dans l'URL d'un lien, l'injection SQL afin de contourner un mot de passe, l'utilisation des métadonnées d'une image à l'aide de l'outil exiftool et enfin de la cryptographie. Des challenges supplémentaires ont été ajoutés comme l'utilisation d'une requête WHOIS, qui sert à identifier le propriétaire d'une adresse IPv6 et intercepter son trafic. Pour terminer, le joueur·euse doit réaliser une attaque par bruteforce à l'aide d'un petit script Python qu'il doit écrire.

4.3 Techniques mobilisées

Les jeux utilisent un ensemble de techniques du hacking éthique : recherche OSINT sur les réseaux sociaux ; lecture attentive du code HTML et des feuilles de style pour trouver du contenu dissimulé ; modification manuelle des cookies et des paramètres `GET` afin de détourner la logique d'un site ; injection SQL destinée à contourner les contrôles d'authentification ; extraction et interprétation des métadonnées EXIF d'images ; cryptanalyse (décodage César ou ROT-47) ; rédaction de courts scripts Python pour l'automatisation (attaque par force brute, déchiffrement) ; enfin, utilisation des services

WHOIS et des requêtes DNS pour cartographier une infrastructure et remonter jusqu'à son propriétaire, le tout dans une histoire narrative progressive.

Le participant·e découvre, étape après étape, comment procède un professionnel de la cybersécurité : récolte d'informations, exploitation et utilisation des données et réflexion pour remonter une piste et ainsi atteindre le but.

4.4 Analyse critique

4.4.1 Points forts

Parmi les forces de la plateforme, les enquêtes sont construites afin de suivre une progression graduelle. Chaque épreuve ré-exploite la précédente et favorise un apprentissage. La narration permet de maintenir le joueur·euse motivé·e mais le garde dans une optique d'apprentissage.

En effet, la boîte à outils intégrée, qui contient les fiches pratiques, évite aux débutant·e·s de devoir faire trop de recherches et ainsi leur permet de se focaliser sur le jeu.

De plus, grâce à un mini IDE Python et un terminal intégré, comme le montre la Figure 6 et Figure 7, le joueur·euse n'a rien besoin d'installer sur sa machine. L'expérience se déroule entièrement sur le navigateur ce qui abaisse la barrière d'entrée, et la variété des techniques abordées offrant un panorama cohérent de la sécurité offensive.

ARCHITECTURE DE LA PLATEFORME *CYBERGAME*
EXISTANTE

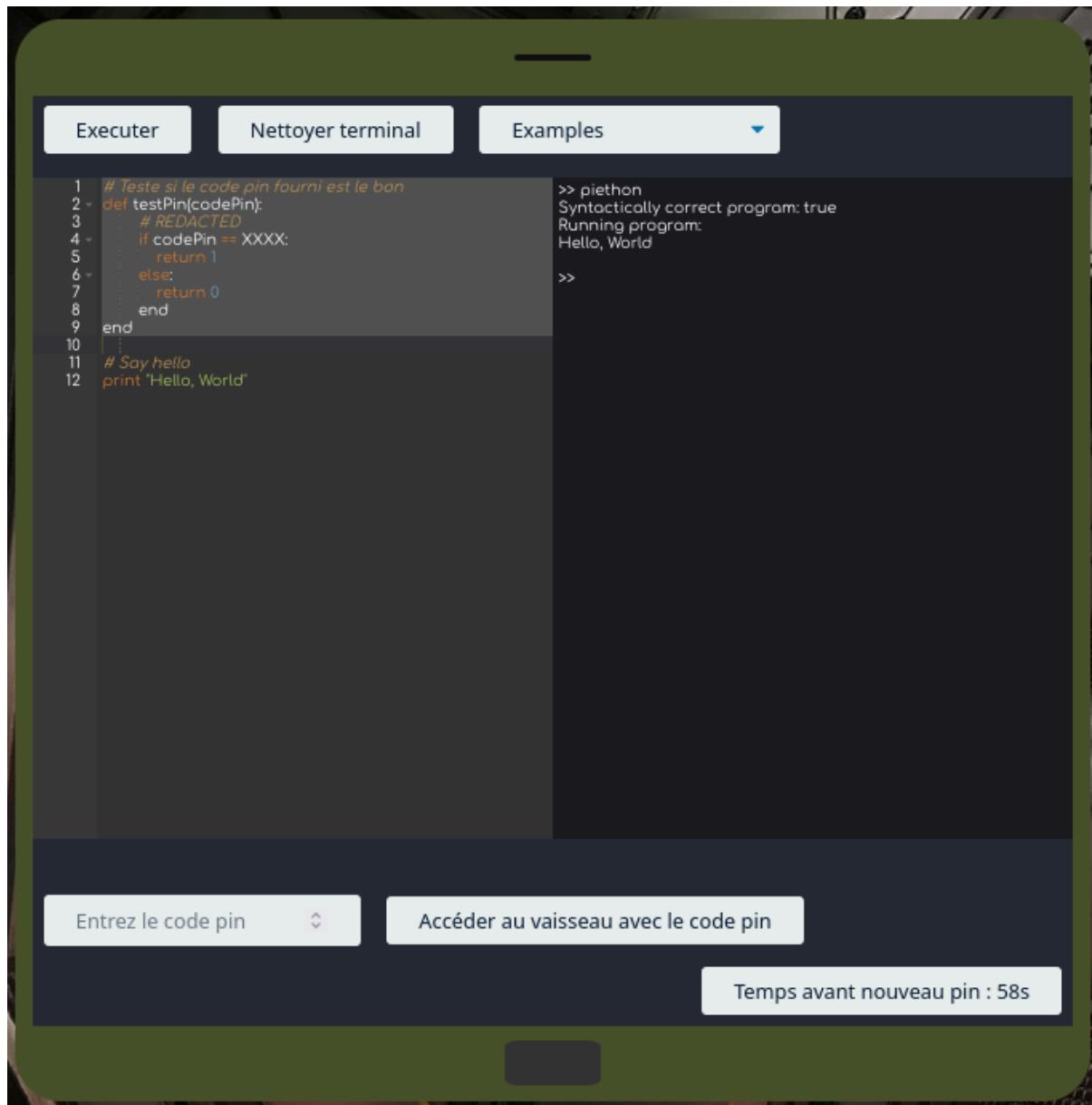


Fig. 6. – IDE présent sur le jeu « Sauve la Terre de l'arme galactique », dans le challenge 6

```
root@e5927c945a49:~# whois 2606:4702:4700:3498::1
#
# ARIN WHOIS data and services are subject to the Terms of Use
# available at: https://www.arin.net/resources/registry/whois/tou/
#
# If you see inaccuracies in the results, please report at
# https://www.arin.net/resources/registry/whois/inaccuracy_reporting/
#
# Copyright 1456-1707, Intergalactic Registry for Internet Numbers, Ltd.
#
NetRange:      2606:4702:4700:3498:: - 2606:4702:4700:3498:FFFF:FFFF:FFFF:FFFF
CIDR:          2606:4702:4700:3498::/64
NetName:        KEPLNET
NetHandle:      NET6-2606:4702:4700:3498-1
Parent:         NET6-2600 (NET6-2600-1)
NetType:        Direct Allocation
OriginAS:       ZU143455
Organization:   Kepl, Inc.
RegDate:        1627-02-01
Updated:        1627-02-01
Comment:        Property of Kepler
Ref:            https://rdap.arin.net/registry/ip/NET6-2606:4702:4700:3498::

OrgName:        Kepl, Inc.
OrgId:          KEPLINC
Address:        Namak Spaceport
Planet:         Namak
RegDate:        1627-02-01
Updated:        1627-02-01
Ref:            https://rdap.arin.net/registry/entity/KEPLINC

OrgAbuseHandle: ABUSE2916-ARIN
OrgAbuseName:   Abuse
OrgAbusePhone: +123-632-234-9854
OrgAbuseEmail:  abuse@kepler.com
```

Fig. 7. – Terminal présent sur les 2 jeux, dans les challenges 9 de « Shana a disparu » et 5, 8 dans « Sauve la Terre de l'arme galactique »

4.4.2 Axes d'amélioration

Cependant, quelques points mériteraient des améliorations.

D'abord, la police d'écriture utilisée, elle permet de créer une certaine ambiance mais elle est peu lisible, ce qui peut gêner la compréhension et la lecture des consignes.

Un autre élément d'amélioration aurait été de réaliser un changement de curseur sur les éléments cliquables (par exemple, en utilisant `cursor: pointer` en CSS) pour permettre au joueur·euse d'identifier les zones interactives.

Actuellement, certains éléments interactifs ne sont pas mis en valeur, ce qui peut rendre la navigation moins intuitive.

La gestion de la fenêtre du jeu pourrait être optimisée, par exemple après la fermeture d'une pop-up, le joueur·euse se retrouve parfois avec un fond noir sans indication, ce qui peut désorienter. Il serait utile d'ajouter des repères visuels ou des messages d'aide pour guider l'utilisateur·trice dans la progression, et de mieux intégrer la boîte à outils dès la page d'accueil pour que chacun·e sache où trouver les ressources.

Le design du site présente parfois des problèmes d'affichage selon la taille de la fenêtre du navigateur, comme le chevauchement d'éléments.

Le champ dans lequel le joueur·euse doit saisir sa réponse ne précise pas toujours le format exigé ; lorsque la consigne n'affiche qu'un mot mis en gras, qui représente la réponse attendue, l'information passe facilement inaperçue et l'utilisateur·trice ignore s'il doit entrer un mot-clé, une URL complète, un hash ou une date. Le joueur·euse peut avoir du mal à comprendre ce qu'il doit mettre, ce qui peut entraîner de la frustration. Il serait judicieux, par exemple de mettre dans l'indice, le format attendu avec un exemple, ou encore avant les début des challenges, montrer des exemples de formats attendus.

Ensuite, pour la validation de l'étape, il faut impérativement entrer une réponse valide dans le champ « Réponse » malgré que l'interface visuelle du jeu change.



Fig. 8. – Interface du jeu après la validation d'un challenge

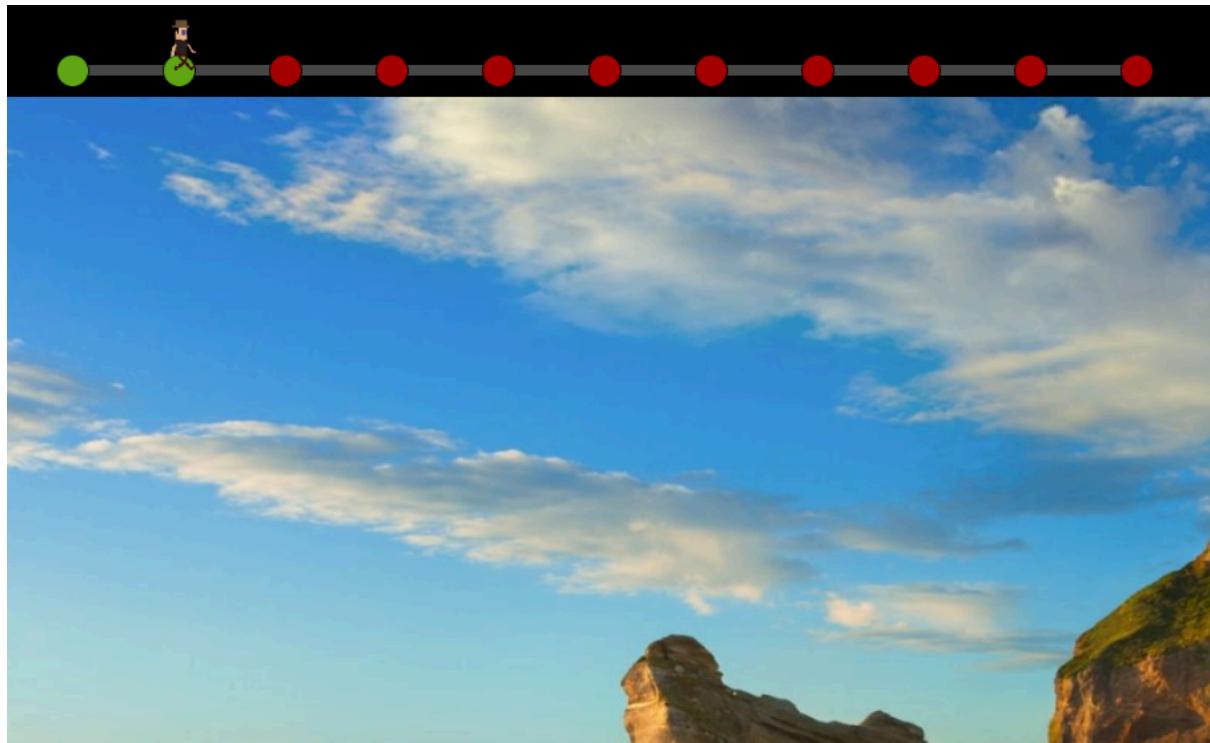


Fig. 9. – Interface du jeu qui ne change pas après la validation d'un challenge et pas de progression dans l'histoire.

Dans la Figure 8, nous pouvons voir que le joueur·euse à bien réussi à trouver la réponse du challenge. Cependant, comme le montre la Figure 9, l'interface de travail ne change pas. Le joueur·euse peut ne pas comprendre ce qu'il doit faire et peut rester bloqué car il ne sait pas ce qui est attendu de lui. Le passage d'un challenge au suivant manque parfois de fiabilité : la pop-up explicative ne s'ouvre pas systématiquement et la barre de progression reste figée. Le participant·e doit donc cliquer sur l'étape suivante pour accéder à la consigne du challenge suivant ainsi que le nouvel interface de travail.

Enfin, les indices actuels fournissent, dans un premier temps, un bon point de départ. Cependant, cela peut se révéler insuffisant pour les joueur·euse·s débutant·e·s. La mise en place d'aides graduelles pourraient limiter le risque d'abandon tout en gardant le défi intéressant.

4.5 Architecture technique

La plateforme est hébergée sur un serveur web, accessible via un nom de domaine `heig-vd.ch` avec le sous-domaine `shana`. Le site utilise des technologies web standards telles que HTML, CSS et JavaScript pour l'interface utilisateur.

4.5.1 Frontend

La structure du frontend montre que chaque épreuve est développée comme un mini-site indépendant dans son propre dossier. Cela permet d'obtenir ainsi une architecture claire et modulaire qui facilite la maintenance et l'ajout de nouveaux niveaux. Chaque challenge suit une structure composée de plusieurs éléments, chacun avec un rôle spécifique dans l'expérience pédagogique.

Le dossier racine du challenge (par exemple `01_windows_login/`, `07_url_modification/`, ...) contient toutes les ressources spécifiques à l'épreuve.

Le fichier HTML de lancement (comme `windows_login.html` ou `gallery1.html`, ...) représente la partie interactive visible par le joueur. Ce fichier charge systématiquement plusieurs ressources : une feuille de style locale située dans `css/style.css`, jQuery version 1.7.1 accompagné parfois d'un script global stocké dans `/js`, ainsi que le header commun comprenant le logo, le compteur de progression et le bouton « Retour ». Un élément `div.popup-trigger` est toujours présent pour déclencher l'affichage de la pop-up d'aide.

Le sous-dossier `css/` contient les styles spécifiques au challenge. La feuille de style définit l'apparence visuelle (police, arrière-plan, couleurs).

Le sous-dossier `img/` stocke les ressources visuelles nécessaires comme les illustrations, les captures d'écran et les images de fond. Par exemple, `background_history.png` sert uniquement visuel du niveau « Browser History ».

Le fichier `popup.html` présente la pop-up d'introduction et d'indices. Tous les challenges utilisent la même structure, c'est-à-dire un titre, le contexte de l'épreuve, un rappel du format de réponse attendu et un bouton « Commencer » pour lancer le défi. En dessous, se trouve aussi le bouton « Indice » qui permet d'obtenir l'indice pour résoudre le challenge.

Le système de validation assure la communication avec le backend. La majorité des pages envoient une requête fetch POST vers `/api/checkAnswer` (ou vers `/db/...` pour le challenge d'injection SQL). Le corps de la requête au format JSON contient les champs `challengeId` et `answer`. En retour, le serveur renvoie une réponse indiquant `success:true` avec l'URL du challenge suivant.

4.5.1.1 Flux type côté client

Lorsque le joueur·euse arrive sur une page de challenge, une pop-up s'ouvre automatiquement pour montrer le contexte du défi et expliquer l'objectif.

Le joueur·euse peut ensuite interagir avec la page selon ce qui lui est demandé : cela peut impliquer de fouiller dans l'historique du navigateur, d'inspecter le DOM pour trouver des éléments cachés, de modifier la valeur d'un cookie, ...

Quand le joueur·euse pense avoir trouvé la solution, il propose sa réponse dans un champ de saisie sur la page. Cette proposition déclenche un appel vers l'API du serveur pour valider la réponse.

Si la réponse est correcte, le backend va réaliser la mise à jour de la progression du joueur·euse dans la base MongoDB et renvoie l'URL du challenge suivant. Côté frontend, le pop-up de félicitations se ferme automatiquement et le prochain onglet devient accessible dans la barre, ce qui permet au joueur·euse de continuer son parcours.

4.5.2 Cartographie des challenges

La cartographie des challenges (Annexe-A) de la plateforme est réalisée à l'aide d'un tableau JSON qui répertorie les différents challenges, les techniques ciblées et les intentions pédagogiques. Chaque ligne du tableau correspond à un challenge spécifique, avec des informations sur le dossier ou le fichier de lancement, la technique ciblée et l'intention pédagogique. Cela permet de visualiser rapidement la structure des jeux et les compétences que chaque challenge vise à développer.

Ce fichier permet d'avoir une vue d'ensemble sur les challenges, c'est-à-dire combien d'épreuves il y a, dans quel ordre et où se trouve le fichier de lancement de chacun. De plus, ce fichier permet un contrôle d'intégrité. Si la plateforme ne se charge pas, il est facile de vérifier si le lien dans le JSON pointe vers un fichier existant. Pour chaque challenge, on peut tout de suite ouvrir les bons fichiers (HTML / JS / CSS) et identifier la vulnérabilité simulée sans chercher. Enfin, si l'équipe ajoute un nouveau challenge, il suffira d'actualiser le JSON.

Ordre	Dossier / fichier de lancement	Technique ciblée	Intention pédagogique
0	0_Intro (pas de challenge)	-	Mise en contexte
1	01_windows_login/ windows_login.html	OSINT et mot de passe à partir des réseaux sociaux	Montrer l'impact de l'exploitation des données personnelles publiquement accessibles
2	02_browser_history/ browser_history.html	Lecture d'historique	Comprendre la collecte de preuves côté client
3	03_same_color_text/ index-01.html	Texte blanc-sur-blanc	Chercher du contenu caché dans le DOM
4	04_html_comment/ comment.html	Commentaires HTML	Repérage d'indices dans la source
5	05_admin_cookie/ index.html	Manipulation de cookie	Bypass d'autorisation client-side

Ordre	Dossier / fichier de lancement	Technique ciblée	Intention pédagogique
6	06_caesar_cipher/ cesar_data.html	Chiffrement César	Notions de cryptanalyse papier
7	07_url_modification/ gallery1.html	Altération de paramètre GET	URL tampering / directory browsing
8	08_SQL_injection/ sql_injection.html	Injection SQL	Contourner une authentification
9	09_image_forensic/ index.html	EXIF / métadonnées	OSINT sur images, géolocalisation
10	10_outro	-	Clôture et teasing final

La même logique est appliquée au scénario « Sauve la Terre de l'arme galactique » 4.2.2, avec des défis similaires mais adaptés à un univers de science-fiction.

4.5.3 Backend

La couche serveur repose sur une architecture composée de Node.js, MongoDB et MySQL, le tout orchestré par Docker. Cette infrastructure comprend plusieurs éléments techniques avec des objectifs pédagogiques.

L'environnement Docker Compose déploie un service backend basé sur Node 14, accompagné d'instances MongoDB et MySQL, ainsi que trois petits conteneurs docker-ssh servant de cibles d'attaque. Cette approche permet d'isoler chaque composant et de créer un environnement de test sécurisé pour les exercices d'injection SQL.

L'API Express, écrit dans le fichier `index.js` (Annexe-B), intègre les middlewares essentiels comme CORS, body-parser, cookie-parser et JWT pour la gestion des sessions. Elle montre plusieurs endpoints REST tels que `/db`, `/db/search`, `/user` et `/stats`.

Les modèles Mongoose, définis dans `db.js` (Annexe-C), gèrent trois collections principales : `Flag`, `User` et `Visitor`. Le système d'initialisation automatique calcule un hash SHA-3 256 pour chaque flag déclaré dans les variables d'environnement `CHALL_FLAGS_2020` et `CHALL_FLAGS_2021`, puis l'insère en base s'il n'existe pas déjà. Cette approche assure une gestion persistante des comptes utilisateurs et du système de score sans exposer les réponses en clair.

Enfin, la base MySQL est initialisée via le script `init.sql` (Annexe-D) avec une table `users` contenant les champs `ID` et `pass` (mots de passe stockés en clair), ainsi qu'une table `posts` pour les fonctionnalités de recherche et de like. Cette base est volontairement dépourvue de protections (pas d'index, pas de contraintes) pour servir de cible d'apprentissage dans les exercices d'injection de code.

4.5.3.1 Séquence de validation d'un challenge

Lorsque le joueur·euse soumet une réponse, le frontend envoie une requête POST vers l'endpoint `/db` ou `/db/search`. L'API Node reçoit cette requête et exécute directement la requête MySQL sans échappement des caractères. Si la requête retourne au moins une ligne de résultat, la réponse est correcte.

Le backend procède alors à la mise à jour des données en modifiant les champs `Flag` et `User.flagged` dans la base MongoDB. Une fois cette mise à jour effectuée, le serveur renvoie une réponse `HTTP 200` avec soit le flag, soit l'URL qui mène à la prochaine étape du challenge.

Côté frontend, la réception de cette réponse positive déclenche l'affichage d'une pop-up de félicitations et déverrouille automatiquement l'accès à la page suivante, conformément à la configuration définie dans le fichier de mapping JSON (Annexe-A) gère la progression entre les différents challenges.

4.6 Infrastructure Docker compose

La couche serveur repose sur Node.js/Express avec MongoDB et MySQL, le tout orchestré par Docker Compose. Un reverse-proxy Traefik termine le TLS et route les requêtes vers trois familles de services : l'API « backend », le site « frontend » et la passerelle « webssh » qui sert d'interface vers de petites machines SSH dédiées à certains défis. Cette séparation permet d'isoler les composants pédagogiques (ex. SQL volontairement vulnérable) sans impacter la persistance « réelle » (progression des joueur·euse·s).

Le fichier `docker-compose.yml` (Annexe-E) définit l'ensemble des services nécessaires à l'application, y compris le backend et les bases de données. Chaque service est configuré avec des variables d'environnement spécifiques pour garantir un fonctionnement optimal.

4.6.1 Services principaux

Les services principaux définis dans le fichier `docker-compose.yml` incluent :

Le service Traefik agit comme point d'entrée de la plateforme, qui écoute sur les ports `80` et `443`. Il assure la terminaison TLS et effectue un routage sur des labels Docker. L'entrypoint web redirige automatiquement le trafic HTTP vers HTTPS pour garantir la sécurité des communications. Le routage s'appuie sur trois règles principales : le frontend est servi sur la racine du domaine, le backend est accessible via le préfixe `/backend` avec un middleware StripPrefix, et le service webssh est routé via les préfixes `/ssh` et `/static`.

Le service backend est construit à partir du répertoire `DigitalDay_BACKEND` et constitue l'API REST. Il utilise Node.js Express et intègre les middlewares essentiels comme CORS, body-parser, cookie-parser et JWT pour la gestion des sessions. L'API expose plusieurs endpoints importantes : les routes de validation des flags (`/:year/flag`, `/:year/checkFlag`), la gestion des utilisateurs (`/user`, `/login`,

`/logout`), les statistiques administratives (`/stats/*`), et les routes SQL (`/db`, `/db/search`). Le service attend la disponibilité de MySQL et MongoDB avant de démarrer, utilisant un mécanisme de wait-for.

Le frontend, construit depuis `DigitalDay_APP`, est servi par live-server sur le port `3001` et routé par Traefik sur la racine du domaine. Il consomme l'API backend via les appels `/backend/...` et intègre des iframes vers `/ssh/...` pour les défis nécessitant un accès terminal.

Le service webssh, basé sur un serveur Python wssh, fournit une interface web pour accéder aux conteneurs SSH internes. Il est routé via les préfixes `/ssh` et `/static` et permet l'accès sécurisé aux machines cibles sans exposition directe sur Internet.

L'architecture adopte délibérément deux systèmes de gestion de base de données distincts pour séparer les responsabilités et les niveaux de sécurité. MongoDB stocke les données critiques de la plateforme dans trois collections principales. La collection `Flag` contient les flags hachés en SHA3-256 initialisés depuis les variables d'environnement `CHALL_FLAGS_2020`, `CHALL_FLAGS_2021` et `CHALL_FLAGS_2025`. La collection `User` gère les profils utilisateur avec les champs `uuid`, `name`, `surname`, `mail` et un tableau `flagged` pour suivre la progression. La collection `Visitor` maintient des compteurs de visiteurs par heure avec les champs `hour_timestamp` et `ctr`. Cette base assure la persistance fiable des données utilisateur et du système de score. La base MySQL, initialisée via le script `init.sql`, sert exclusivement aux défis d'injection SQL. Elle contient deux tables : `users` avec les champs `ID` et `pass` (mots de passe volontairement stockés en clair), et `posts` pour les fonctionnalités de recherche avec les champs `ID`, `img`, `nameLastname` et `datepost`. Cette base est intentionnellement dépourvue de protections (pas d'index, pas de contraintes d'intégrité) pour servir de cible d'apprentissage dans les exercices d'injection de code.

Les deux DB présentes dans l'architecture du code ont des objectifs bien distincts. MongoDB stocke les données « sérieuses » (profils, progression). Alors que MySQL n'est utile que pour le challenge 08 de Shana : l'objectif est d'isoler la DB sans risquer d'altérer les vrais enregistrements Mongo si un étudiant·e pousse l'exploit plus loin.

4.6.2 Routage et intégration Traefik

Le routage repose sur Traefik v2.10 qui agit comme reverse proxy et point d'entrée unique pour tous les services. Cette configuration permet une gestion centralisée du HTTPS et l'application de middlewares de transformation des requêtes. Configuration du routage Traefik termine automatiquement les connexions HTTPS et applique des middlewares de type StripPrefix pour traiter les requêtes entrantes. Du côté client, le frontend peut appeler directement les endpoints via des préfixes standardisés comme `/backend/...` ou `/ssh/...` sans se préoccuper de la complexité du routage interne. Le processus de transformation s'effectue de la manière suivante : Traefik reçoit les requêtes avec préfixes, retire automatiquement ces préfixes via les middlewares configurés, puis transmet les requêtes aux

services backend qui reçoivent des chemins « nus » sans préfixes. Exemples de transformation : Une requête `POST /backend/visitor` devient `POST /visitor` pour le service Express. Une requête `GET /ssh/?hostname=sshmachine` devient `GET /?hostname=sshmachine` pour le service wssh, qui se connecte ensuite en interne à `sshmachine:22`.

4.6.3 Cartographie des routes et services

Routage des appels API frontend L'architecture de communication entre le frontend et le backend s'articule autour d'un système de routage centralisé. Les écrans et scripts JavaScript du frontend communiquent avec l'API backend via le préfixe `/backend/`, qui est automatiquement retiré par Traefik avant transmission à Express.

Routage principal des pages : La page d'accueil `index.html` effectue un appel `POST /backend/visitor` pour incrémenter le compteur de visiteurs de la plateforme. La page de connexion `login.html` utilise deux endpoints : `POST /backend/login` pour l'authentification des utilisateurs et `GET /backend/logout` pour la déconnexion. La page de statistiques `statistics.html` interroge plusieurs endpoints via `GET /backend/stats/*` pour récupérer les données de progression et de performance.

Scripts JavaScript globaux : Le script `js/main.js` gère les interactions du scénario principal avec deux appels principaux : `POST /backend/2020/flag` pour la validation des réponses et `POST /backend/user` pour la gestion des données utilisateur. Le script `js/galacmain.js` suit la même logique pour le scénario galactique avec `POST /backend/2021/flag` et `POST /backend/user`.

Challenges spécifiques : Le challenge Windows Login (`challenges/01_windows_login/windows_login.html`) utilise l'endpoint `POST /backend/2020/checkFlag` pour valider les tentatives d'authentification OSINT. Le challenge d'injection SQL (`challenges/08_SQL_injection/sql_injection.html`) communique directement avec `POST /backend/db` pour permettre l'exploitation des vulnérabilités de base de données.

Gestion des défis terminal Pour les défis nécessitant un accès terminal, l'architecture implémente un système d'iframe pointant vers `/ssh?....`. Ce flux de données transite par Traefik qui redirige vers le service webssh, lequel établit la connexion avec les conteneurs cibles `sshmachine*` dédiés à chaque type d'exercice. Cette approche permet d'isoler les environnements d'apprentissage tout en maintenant une interface utilisateur cohérente intégrée dans le navigateur.

4.6.4 API Express (Annexe-B)

L'API Express constitue la couche serveur principale de la plateforme CyberGame. Elle est structurée autour de plusieurs modules fonctionnels qui gèrent les différents aspects du système de jeu et d'administration.

Gestion des flags et progression Le système de validation des défis repose sur deux endpoints principaux. L'endpoint `POST /:year/flag` effectue la vérification complète d'un flag soumis par un joueur. Il procède au hashage de la réponse proposée avec l'algorithme SHA3-256, puis compare ce hash avec les valeurs stockées dans la base MongoDB. En cas de correspondance, le système ajoute automatiquement l'identifiant `<year>_<chall>` au tableau `user.flagged`, permettant ainsi le suivi de la progression du joueur. L'endpoint `POST /:year/checkFlag` offre une variante qui se contente de vérifier la validité du flag sans modifier la progression, utile pour les vérifications intermédiaires.

Gestion des utilisateurs et sessions Le système de suivi des joueur·euse·s s'appuie sur un middleware de gestion des cookies UUID. Chaque visiteur se voit attribuer automatiquement un identifiant unique généré avec `uuidv4`, permettant de suivre sa progression sans nécessiter de création de compte préalable. L'endpoint `POST /user` permet l'enregistrement des informations personnelles (nom, prénom, email) uniquement après validation complète de tous les flags de l'édition courante, garantissant ainsi que seuls les participants ayant terminé le parcours peuvent s'enregistrer. Pour l'administration, l'endpoint `POST /login` gère l'authentification des administrateurs via les variables d'environnement `SHANA_USER` et `SHANA_PASS`. Une fois authentifié, l'administrateur reçoit un cookie JWT auth token lui donnant accès aux fonctionnalités statistiques. L'endpoint `GET /logout` permet la déconnexion en supprimant simplement le cookie d'authentification.

Module statistiques L'accès aux statistiques est protégé par une vérification JWT obligatoire. L'endpoint `GET /stats/getEditions` retourne la liste des années supportées par la plateforme. Les endpoints `GET /stats/visitors`, `GET /stats/finished?year=...` et `GET /stats/flagPerChall?year=...` fournissent respectivement le nombre total de visiteurs, le nombre de participants ayant terminé une édition spécifique, et les statistiques de réussite par challenge, le tout grâce à des agrégations MongoDB optimisées.

Comptabilisation des visiteurs L'endpoint `POST /visitor` maintient un système de comptage granulaire en incrémentant un compteur par heure dans la collection `Visitor`. Cette approche permet d'obtenir des statistiques de fréquentation précises sur la plateforme.

Interface SQL vulnérable Dans un objectif pédagogique, deux endpoints exposent volontairement des vulnérabilités d'injection SQL. L'endpoint `POST /db` exécute directement les requêtes construites par concaténation sur la base MySQL `dday.users`, sans aucune protection contre les injections. De même, `POST /db/search` applique la même logique volontairement fragile pour les recherches sur la table `posts`. Cette architecture permet aux étudiants d'expérimenter les techniques d'injection SQL dans un environnement contrôlé.

Système PIN temporel Spécifiquement conçu pour le défi 2021, les endpoints `GET /pin` et `POST /pin` implémentent un mécanisme de génération et vérification de PIN basé sur un seed dérivé de la minute courante. Ce système temporel ajoute une dimension dynamique au challenge, le PIN étant régénéré automatiquement. En cas de validation réussie, le système retourne un flag issu de la

variable d'environnement `CHALL_FLAGS_2021`. Cette architecture modulaire permet à la fois un fonctionnement sécurisé pour les aspects critiques de la plateforme (progression, authentification) tout en exposant délibérément des vulnérabilités dans un cadre pédagogique contrôlé.

4.6.5 WebSSH et conteneurs SSH

Le service WebSSH constitue une composante essentielle de l'infrastructure de la plateforme, permettant aux participants d'accéder à des environnements SSH directement depuis leur navigateur. Cette solution repose sur le serveur wssh développé en Python, qui fournit un terminal web accessible via l'endpoint `/ssh`.

Architecture du service WebSSH Le service WebSSH est containerisé via le fichier `Dockerfile_ssh` qui utilise une image Python 3 de base et installe le package `webssh`. Ce conteneur est exposé au travers du reverse proxy Traefik avec les règles de routage appropriées, incluant la gestion des préfixes `/ssh` et `/static` ainsi que l'activation du TLS.

Au sein du réseau Docker interne, le service WebSSH établit des connexions vers trois types de machines cibles spécialisées, chacune répondant à des objectifs pédagogiques spécifiques :

- `ssh` : Il s'agit du conteneur SSH principal basé sur une image Ubuntu 14.04. Cet environnement utilise `rbash` (restricted bash) avec un PATH restreint et met à disposition un ensemble d'outils sélectionnés tels qu'`exiftool` et `ls`. Cette configuration permet de créer un environnement d'apprentissage contrôlé où les participants peuvent explorer des techniques de reconnaissance et d'analyse de fichiers.
- `sshmachine-whois` : Cette variante réduite du conteneur SSH standard est spécialement conçue pour les défis liés aux requêtes WHOIS. Le conteneur intègre un binaire `whois` personnalisé et se concentre sur les techniques d'OSINT (Open Source Intelligence) et de reconnaissance réseau.
- `sshmachine-galactic-forensic` : Ce conteneur spécialisé a été développé pour un défi forensique de l'édition 2021 du scénario « Sauve la Terre de l'arme galactique ». Il inclut des outils d'analyse forensique tels que `libimage-exiftool-perl` et contient des fichiers d'investigation spécifiques comme `photo-gallery.jpg`. Si le dossier source n'est pas disponible, un Dockerfile minimal doit être créé incluant les outils essentiels : `file`, `strings`, `binwalk`, `exiftool`, `less` et `grep`.

Configuration technique Tous les conteneurs SSH utilisent une configuration similaire basée sur Ubuntu 14.04 avec le serveur OpenSSH et Supervisor pour la gestion des processus. La configuration SSH est modifiée pour permettre l'authentification par mot de passe root (`PermitRootLogin yes`), créant un environnement volontairement vulnérable à des fins pédagogiques. Cette architecture modulaire permet d'isoler chaque type de défi tout en maintenant une interface d'accès unifiée via le terminal web. Les participants peuvent ainsi accéder à différents environnements d'apprentissage selon les objectifs pédagogiques de chaque challenge, sans nécessiter d'installation locale d'outils spécialisés.

4.7 Analyse de la sécurité

L'architecture de sécurité de la plateforme CyberGame repose sur une approche de séparation stricte des responsabilités entre deux systèmes de gestion de base de données distincts. MongoDB assure la gestion fiable de l'état du jeu, incluant les profils utilisateurs, la progression et les données critiques du système, tandis que MySQL est uniquement utilisé pour les exercices d'injection SQL, créant ainsi une surface de vulnérabilité contrôlée et isolée.

La gestion des variables sensibles s'appuie sur un système d'injection sécurisé via les fichiers d'environnement (.env). Les flags de validation des challenges sont stockés sous forme de hash SHA-3 256 dans MongoDB, garantissant qu'aucune réponse n'est exposée en clair dans le système principal. Cette approche préserve l'intégrité pédagogique des exercices tout en maintenant un niveau de sécurité approprié pour les données persistantes.

L'architecture présente volontairement des vulnérabilités dans les routes SQL qui utilisent des concaténations de chaînes sans échappement, permettant ainsi l'apprentissage pratique des techniques d'injection. Cette faiblesse intentionnelle est circonscrite à MySQL et ne compromet pas l'intégrité des données MongoDB grâce à la séparation architecturale stricte. Les étudiants peuvent ainsi explorer les failles d'injection SQL sans risquer d'altérer les vrais enregistrements du système.

L'utilisation de Traefik avec la fonctionnalité StripPrefix simplifie considérablement le code applicatif Express en gérant automatiquement les préfixes d'URL publics.

Cette approche permet au serveur d'application de se concentrer sur la logique métier sans avoir à gérer la complexité du routage externe, améliorant ainsi la maintenabilité et la sécurité du code. Cette architecture bicéphale offre un environnement d'apprentissage sécurisé où les vulnérabilités pédagogiques restent confinées à leur domaine d'application spécifique, préservant l'intégrité globale du système tout en offrant une expérience d'apprentissage authentique aux utilisateurs.

5 Propositions de nouveaux scénarios

Ce chapitre présente les différents scénarios que j'ai imaginés pour la plateforme *CyberGame*. L'objectif était de trouver des histoires captivantes et pédagogiques tout en intégrant des éléments de jeu pour travailler les aspects de la cybersécurité. Pour réussir à trouver un scénario adapté, je me suis inspirée de faits réels et de CTF afin d'imaginer les challenges. J'ai donc analysé les mécaniques de jeux et leurs objectifs pédagogiques. J'ai ensuite élaboré trois scénarios, chacun avec ses propres défis et compétences visées tout en restant accessible à tout le monde. Chaque scénario est présenté avec une description détaillée, les compétences travaillées et les étapes pour le résoudre.

5.1 Scénario réaliste : Blackout dans le Centre Hospitalier Horizon Santé

Ce scénario reprend une situation fictive, mais inspirée de faits réels. Ici, le joueur·euse fait partie d'une équipe de cybersécurité qui fait face à une attaque de ransomware dans un hôpital. Le but est de résoudre des défis techniques pour rétablir les services vitaux avant qu'il ne soit trop tard. L'inspiration de ce scénario vient de plusieurs incidents réels, notamment les attaques de ransomware avec une hausse croissante sur des infrastructures critiques comme les hôpitaux et les réseaux électriques (*When Ransomware Kills: Attacks on Healthcare Facilities / IBM [sans date]*).

Dans un premier temps, le joueur·euse doit remonter à l'origine de l'attaque en analysant un e-mail de phishing qui a permis aux attaquants de pénétrer le réseau de l'hôpital. L'e-mail de phishing révèle le domaine pirate ; c'est la première piste. Ensuite, il devra explorer un faux portail VPN mis en place par les attaquants pour exfiltrer des données. Grâce au commentaire HTML laissé par négligence, le joueur·euse voit l'inventaire complet des sauvegardes et récupère une archive historique qui contient un malware. Dans cette archive se cache `hx_dropper.ps1` ; la dé-obfuscation fournit l'adresse du serveur C2, prouvant que l'hôpital est toujours sous contrôle externe. Sur le C2, un fichier de configuration chiffré recèle le mot de passe administratif du ransomware ; une simple attaque XOR suffit à l'extraire. Grâce à ce mot de passe, le joueur·euse peut accéder aux journaux du ransomware et découvrir un kill-switch caché dans une radiographie. En entrant ce kill-switch dans la console d'urgence, le joueur·euse neutralise la seconde vague d'attaques et sauve les services vitaux de l'hôpital.

« Le Centre hospitalier Horizon Santé tourne sur groupe électrogène depuis trois heures : un ransomware a chiffré les serveurs cliniques, puis a sauté la barrière réseau et mis hors service le

réseau électrique qui alimente le bloc opératoire. Le générateur de secours n'a plus que 68 minutes d'autonomie. Si rien n'est fait, huit opérations à cœur ouvert devront être interrompues. Votre équipe vient d'être branchée en urgence sur le réseau isolé de l'hôpital. Votre mission : remettre les services vitaux en ligne avant la fin du compte à rebours et bloquer la seconde vague annoncée par les attaquants. »

Challenges à réaliser

Etape	Nom du challenge	Compétence travaillée	Description du challenge
1	<u>Mail Contagieux</u>	OSINT & forensic e-mail	Analyse d'un fichier .eml : inspection des entêtes Received/Return-Path pour identifier l'IP et le domaine d'envoi.
2	<u>Shadow VPN Portal</u>	Exploitation Web	Explorer l'HTML afin d'y trouver un commentaire qui contient la liste de toutes backup.
3	<u>Script d'infection</u>	Reverse Engineering	Dé-obfuscation d'un script PowerShell (Base64 + XOR 0x20) pour révéler l'URL C2 https://c2.hz-cloud.net/api .
4	<u>Coffre chiffré</u>	Cryptographie	Attaque known-plaintext sur vault.cfg.enc : découverte d'une clé XOR répétée (6 octets).
5	<u>Radiographie piégée</u>	Stéganographie	Extraction du kill-switch dissimulé dans la radiographie thorax_xray.png via binwalk/steghide.

5.1.1 Mail Contagieux : OSINT et forensic d'email

Dans un premier temps, le joueur·euse doit analyser un e-mail de phishing qui a permis aux attaquants de pénétrer le réseau de l'hôpital. Cet e-mail contient une pièce jointe malveillante planning_salle_op.xlsx qui a été ouverte par un employé, déclenchant ainsi l'attaque.

1. Ouvrir planning_salle_op.eml dans l'IDE.
2. Ouvrir les en-têtes (Thunderbird / webmail / outil en ligne).
3. Repérer l'IP de la première ligne Received: et le domaine dans Return-Path.
4. Vérifier la réputation du domaine sur un service WHOIS/OSINT.

Outils nécessaires : IDE, WHOIS.

Indices graduels

- Indice 1 : Consulte uniquement les tous premiers entêtes `Received:`, la vraie origine est souvent dans la ligne la plus basse.
- Indice 2 : L'expéditeur imite le sous-domaine support d'Horizon Santé.
- Indice 3 : Vérifie la réputation WHOIS : un domaine proche d'`horizonsante.com`, mais pas identique, ressort comme malveillant.

Flag attendu : `horizonsante-support.com`

Le sous-domaine sera la cible du défi 2.

5.1.2 *Shadow VPN Portal* : Exploitation Web

Le joueur·euse doit maintenant accéder à un faux portail VPN de l'hôpital `https://vpn.horizonsante-support.com/`, qui a été mis en place par les attaquants pour exfiltrer des données. Le faux portail VPN propose un bouton « Dernière sauvegarde » qui appelle :

```
https://vpn.horizonsante-support.com/repo/download.php?file=latest
```

Le joueur·euse devra ensuite explorer le code HTML pour trouver un commentaire qui pointe vers un fichier `/repo/manifest.json`, qui contient la liste des sauvegardes disponibles.

1. Afficher le code source de la page. Chercher `<!--` → on trouve :

```
<!-- TODO : nettoyer /repo/manifest.json avant mise en prod -->
```

2. Ouvrir `https://vpn.horizonsante-support.com/repo/manifest.json` Le JSON liste toutes les sauvegardes.
3. Rejouer la requête de téléchargement mais remplacer `file=latest` par `file=backup-2025-07-12.tar.gz`.
 1. Le serveur répond 200 OK et livre l'archive `backup-2025-07-12.tar.gz`.
 2. En listant l'archive il voit une liste de dossiers et fichiers, dont un fichier `tar.gz` compressé.

Outils nécessaires : Navigateur et DevTools.

Indices graduels

- Indice 1 : Regarde le code source ; les développeurs commentent parfois des URLs utiles.
- Indice 2 : Le fichier `manifest.json` ressemble à un index automatique des sauvegardes.
- Indice 3 : Utilise l'un des noms trouvés pour remplacer `latest` dans le paramètre `file`.

Flag attendu : `hx_srv_full_0712.tar.gz`

Une fois le fichier trouvé décompressé, il devient l'objet du défi 3.

5.1.3 Script d'infection : Reverse Engineering

Une fois qu'il a téléchargé le fichier `hx_srv_full_0712.tar.gz`, le joueur·euse découvre un script PowerShell obfuscué nommé `hx_dropper.ps1`. Ce script est compacté : variables à un caractère, chaîne Base64 + XOR 0x20. Il est utilisé par les attaquants pour établir une connexion avec leur serveur de commande et contrôle (C2) et exfiltrer des données.

1. Repérer la chaîne Base64 dans le code.
2. Dé-obfuscuer : Base64 en bytes puis XOR 0x20.
3. Lire l'URL C2 (<https://c2.hz-cloud.net/api>).

Outils nécessaires : Script Python.

Indices graduels

- Indice 1 : Une chaîne très longue terminant par `=` ou `==` est presque toujours du Base64.
- Indice 2 : Après Base-64 tu verras beaucoup de `0x20`.
- Indice 3 : XOR avec `0x20` caractère par caractère.

FLAG attendu : `c2.hz-cloud.net`

Cette URL pointe vers la clé chiffrée du défi 4.

5.1.4 Coffre chiffré : Cryptographie

Sur ce C2 se trouve `vault.cfg.enc`. Le joueur·euse doit maintenant déchiffrer le fichier de configuration chiffré `vault.cfg.enc` trouvé dans l'archive. Le fichier clair commence par `CFG=`. Le ransomware a utilisé un XOR de 6 octets pour chiffrer ce fichier. Le joueur·euse doit retrouver la clé de chiffrement en utilisant une attaque known-plaintext.

1. Deviner que `CFG= (hex 43 46 47 3D)` est le plaintext.
2. XOR le début du chiffré avec `CFG=` et retrouver la clé.
3. Appliquer la clé pour déchiffrer tout le fichier.
4. Lire la ligne `ADMIN_PASS=Aur0raVital@2025`.

Outils nécessaires : Script Python.

Indices graduels

- Indice 1 : Cherche un motif ASCII typique en clair dans le début du fichier ; un fichier de config commence souvent par `CFG=`.
- Indice 2 : Calcule Chiffré ⊕ Clair sur les 6 premiers octets.
- Indice 3 : Réapplique cette clé répétée jusqu'à la fin, le mot de passe admin apparaît vers les premières lignes.

Flag attendu : `Aur0raVital@2025`

Le mot de passe permet d'ouvrir les logs du défi 5.

5.1.5 Radiographie piégée : Stéganographie

Dans le dossier patient, le joueur·euse trouve une radiographie `thorax_xray.png` qui semble normale, mais qui est anormalement lourd. Le ransomware a dissimulé un kill-switch dans cette image pour désactiver son attaque. Les renseignements obtenus dans le défi 4 (mot de passe `Aur0raVital@2025`) devront être utilisés pour extraire ce message.

1. Télécharger `thorax_xray.png`.
2. Lancer `binwalk -e thorax_xray.png` ou ouvrir l'image avec `steghide` et trouver un fichier caché (ZIP ou steghide data)
3. Quand l'outil demande le mot de passe, entrer `Aur0raVital@2025` (flag du défi 4)
4. Extraire le petit fichier `kill.txt` (ou `kill_switch.conf`) et lire son contenu

Outils nécessaires : Binwalk / steghide / zsteg et éditeur de texte.

Indices graduels

- Indice 1 : Le PNG fait anormalement > 15 Mo : il dissimule très probablement des données concaténées.
- Indice 2 : `binwalk -e` montre qu'un bloc ZIP/Steghide data débute après l'en-tête de l'image.
- Indice 3 : Utilise le mot de passe à l'étape 4 pour déverrouiller le fichier.

Flag attendu : `HZ_SECOND_STOP`

Le joueur·euse copie la chaîne dans le champ kill-switch de la console d'urgence. L'alerte « Seconde vague neutralisée » s'affiche. Le compte à rebours au bloc opératoire s'interrompt avec un dernier message : « Mission accomplie ! Les données patients sont sauvées et la seconde vague n'aura jamais lieu. Nous avons déjà lancé le plan de remédiation complet et enclenché la traçabilité juridique grâce aux évidences collectées. »

5.2 Scénario aventurier : Opération « CipherFox » - Infiltration

Ce scénario plonge le joueur·euse dans la peau d'un espion, l'agent CipherFox, qui doit infiltrer une entreprise de haute technologie pour voler des secrets industriels. Le joueur·euse devra résoudre une série de défis techniques pour mener à bien sa mission sans se faire repérer par l'équipe de sécurité de l'entreprise. Cette histoire s'inspire de récits d'espionnage et de cyberattaques réels, où les hackers exploitent des vulnérabilités pour accéder à des informations sensibles (*Qu'est-ce que le cyberespionnage ?* [sans date]).

Déguisé en consultant, le joueur·euse, alias CipherFox, commence son infiltration depuis sa suite d'hôtel : il déchiffre le hash SHA-1 caché dans les métadonnées d'un PDF public pour deviner le mot de passe et se connecter au Wi-Fi invité de KeyWave Systems. En ligne, il se rend au portail partenaires ; grâce à une injection SQL furtive qui contourne le WAF, il ouvre une session interne et obtient un `session_token`. Afin d'effacer toute trace, il patche ensuite le micro-service `session_tap.exe` le journal d'audit ne consignera plus son passage. Dans le répertoire `/vault/`, il trouve `design_note.sec` qu'il devra réussir à déchiffrer. Enfin, la dernière étape, le SOC a intercepté un dump DNS où chaque sous-domaine `.fox.tunnel` transporte un fragment Base36. Il reconstitue le fichier `plans.zip.aes` et le déchiffre avec la pass-phrase trouvée dans le fichier précédent. Le flag final est révélé dans le fichier `README.txt` de l'archive.

« Vous êtes un espion, l'agent CipherFox, et vous travaillez sous couverture. Déguisé en consultant, tu occupes la suite 1903 d'un palace à Genève. Votre mission : Voler les plans de KeyWave Systems : clé matérielle FIDO2 + déverrouillage biométrique qui pourrait tuer les mots de passe classiques. Sa valeur estimée est de plusieurs millions. Le plan d'exfiltration se déroule en cinq étapes ; chacun correspond à un « challenge » que vous devrez résoudre pour mener à bien la mission sans attirer l'attention de l'équipe de sécurité (SOC) de l'entreprise. »

Challenges à réaliser

Etape	Nom du challenge	Compétence travaillée	Description du challenge
1	<u>Hotspot Mirage</u>	OSINT et Cryptographie	Retrouver le mot de passe Wi-Fi en comparant le SHA-1 stocké dans les métadonnées du PDF « <code>keynote_KeyWave.pdf</code> ».
2	<u>Admin Bypass</u>	Exploitation Web	Contourner le filtre WAF sur le formulaire login des partenaires et obtenir le <code>session_token</code> .

Etape	Nom du challenge	Compétence travaillée	Description du challenge
3	<u>Micro-Patch</u>	Reverse Engineering	Patcher le binaire <code>session_tap.exe</code> (x86) pour désactiver la routine <code>audit()</code> .
4	<u>SecureNote Cipher</u>	Cryptographie	Casser un XOR 3 octets dans <code>design_note.sec</code> afin d'extraire la pass-phrase qui protège les plans.
5	<u>DNS Drip</u>	Forensic réseau	Reconstituer <code>plans.zip.aes</code> à partir des requêtes DNS vers <code>*.fox.tunnel</code> , décoder Base36, déchiffrer avec la pass-phrase.

5.2.1 Hotspot Mirage : OSINT et Cryptographie

Pour ce premier challenge, le joueur·euse doit se connecter au Wi-Fi de KeyWave Systems (KWS) pour accéder à leur intranet. Le mot de passe est partiellement lisible dans un prospectus trouvé dans sa chambre d'hôtel, mais il manque une partie du texte :

Welcome to our guests! Wi-Fi code: KeyWave-**-VIP

Le code suit toujours la convention interne : `KeyWave-<QUADRIMESTRE>-VIP`. Le joueur·euse doit donc retrouver le mot de passe complet en analysant les métadonnées du PDF de la présentation de KeyWave Systems `keynote_KeyWave.pdf`, présent sur le flyer. Ce PDF contient un champ custom metadata `wifi_hash` : une empreinte du mot de passe complet.

1. Télécharger `keynote_KeyWave.pdf`, l'ouvrir avec exiftool et lire la ligne :

```
wifi_hash = 779a10d6ff824bbdfbed49242e48c4806977db3b
```

2. Générer les 4 candidats : `KeyWave-Q1-VIP`, `KeyWave-Q2-VIP`, `KeyWave-Q3-VIP`, `KeyWave-Q4-VIP`.
3. Calculer leurs SHA-1 (`sha1sum` ou `CyberChef`) et comparer pour trouver que seule `KeyWave-Q2-VIP` correspond.
4. Se connecter au réseau KWS-Guest avec ce mot de passe.

Outils nécessaires : Exiftool, `sha1sum` ou `CyberChef`.

Indices graduels

- Indice 1 : Le QR code te permet d'avoir accès à une brochure PDF. Elle conserve des métadonnées ; ouvre-la avec exiftool pour voir s'il n'y a pas un champ inhabituel.

- Indice 2 : Le hash dans le PDF fait 40 hexa, ce qui correspond à SHA-1.
- Indice 3 : le mot de passe suit toujours le motif KeyWave-Q?-VIP ; calcule le SHA-1 des quatre possibilités et repère celui qui correspond au hash trouvé.

Flag attendu : KeyWave-Q2-VIP Le code permet d'avoir accès au Wi-Fi ainsi qu'à la page de connexion des partenaires.

5.2.2 Admin Bypass : Web Exploitation

Le joueur·euse doit maintenant accéder à l'intranet de KeyWave Systems <https://intra.keywave.local/partners/login.php> pour voler les plans. Le formulaire de connexion comporte les champs e-mail et mot de passe. Un email de la responsable média se trouve sur le flyer. Il faudra l'utiliser pour ce challenge. Il doit contourner le filtre basique WAF (Web Application Firewall) sur la page de connexion des partenaires, qui refuse toute requête contenant le mot-clé exact OR (maj/min indifférent) ou la séquence --. Aucune requête préparée et le backend exécute toujours :

```
SELECT partner_id, session_token
FROM partners
WHERE email = '$mail' AND passwd = '$pw';
```

Pour contourner le filtre, le joueur·euse doit utiliser une injection SQL pour éviter la restriction WAF. Il peut utiliser un commentaire /* */ SQL au milieu du mot-clé pour casser le mot-clé OR et ainsi obtenir le session_token du partenaire.

1. Renseigner e-mail avec une vraie adresse interne, qui se trouve dans le pdf Responsable média : alice.martin@keywave.com .
2. Dans mot de passe, saisir : ' 0/**/R 1=1 # (le /*/ casse le mot-clé pour le WAF ; # remplace -- comme commentaire fin de ligne accepté par MySQL).

Outils nécessaires : Navigateur.

Indices graduels

- Indice 1 : Le WAF bloque OR en clair, mais un commentaire /*/ interrompt les mots.
- Indice 2 : MySQL accepte le dièse # comme commentaire d'une ligne.
- Indice 3 : Essaie de scinder OR : 0/**/R , puis termine le restant de la requête avec # . N'oublie pas d'utiliser l'adresse alice.martin@keywave.com trouvée sur le flyer.

Flag attendu : PART-7XG4

5.2.3 Micro-Patch : Reverse Engineering

Le joueur·euse a maintenant le session_token , mais il doit effacer toute trace de sa connexion pour éviter d'être détecté par le SOC. Le micro-service session_tap.exe consigne chaque utilisation d'un

`session_token` partenaire dans un fichier `audit.log`. Tant qu'il détecte la valeur `PART-7XG4` (celle récupérée dans le challenge 2), il écrit une ligne dans ce journal. Le joueur·euse doit modifier le binaire pour que la fonction `audit()` retourne toujours `0`, ce qui effacera toute trace de sa connexion.

1. Ouvrir `session_tap.exe` dans Ghidra.
2. Rechercher la constante ASCII `PART-7XG4`, cela mène à `cmp eax, 0x50415254`. (« PART »).
3. Dans l'éditeur d'octets, remplacer par `31 C0 C3` (`xor eax,eax; ret`).
4. Sauver le binaire et le relancer.

Outils nécessaires : Ghidra, éditeur hexadécimal intégré.

Indices graduels

- Indice 1 : Ouvre le binaire dans Ghidra et fais `Strings`. le token `PART-7XG4` s'y trouve en clair.
- Indice 2 : Clique sur Xrefs de cette chaîne et il y a un `cmp eax, 0x50415254` (ASCII “PART”).
- Indice 3 : Remplace le bloc de comparaison par `31 C0 C3` (`xor eax,eax ; ret`), ce qui permettra à la fonction de renvoyer toujours `0`.

Flag attendu : `patched_md5=7ab8c6de`

5.2.4 SecureNote Cipher : Cryptographie

Le joueur·euse a réussi à se connecter à l'intranet de KeyWave Systems, mais il doit maintenant accéder aux plans FIDO2. Ils sont stockés dans un fichier sécurisé `design_note.sec` dans le répertoire `/vault/`. Le fichier est chiffré avec un XOR répété de 3 octets. La structure du fichier est la suivante : un en-tête non chiffré qui est `KWSX0Rv1` (8 octets), puis le contenu chiffré commence immédiatement après l'en-tête. Le joueur·euse sait que le texte chiffré commence par le mot `TITLE:` (6 octets). Il s'agit d'une attaque de type « known plaintext » (texte clair connu) sur un chiffrement XOR.

1. Télécharger `design_note.sec`.
2. Charger le fichier dans CyberChef et isoler le bloc chiffré.
3. XOR le bloc avec le plaintext connu `TITLE:`, permet de retrouver la clé.
4. Appliquer la clé répétée à tout le fichier.
5. Lire la ligne pass-phrase.

Outils nécessaires : CyberChef ou script Python.

Indices graduels

- Indice 1 : Le fichier commence par `KWSX0Rv1` non chiffré.
- Indice 2 : Juste après l'en-tête, il y a `TITLE:` en clair c'est un plaintext connu pour récupérer la clé.
- Indice 3 : La clé fait 3 octets et tourne en boucle, il faut l'appliquer sur tout le reste pour dévoiler la pass-phrase.

Flag attendu : `K3yW4v3-Q4-VIP-F1D0-M4st3rPl4n!`

5.2.5 DNS Drip : Forensique réseau

Le joueur·euse a maintenant la pass-phrase pour déchiffrer les plans, mais il doit d'abord les récupérer. Les plans FIDO2, contenus dans le fichier `plans.zip.aes`, ont été exfiltrés via un tunnel DNS. Le SOC a fourni un fichier PCAP, `exfil_dns.pcapng`, capturé sur leur Système de Détection d'Intrusion (IDS). Chaque requête DNS vers le domaine `*.fox.tunnel` transporte un bloc du fichier, encodé en Base36. Le joueur·euse doit donc reconstituer le fichier `plans.zip.aes` à partir de ces requêtes DNS capturées, décoder les blocs Base36, puis déchiffrer l'archive obtenue en utilisant la pass-phrase récupérée lors du défi précédent.

1. Ouvrir le PCAP dans Wireshark et filtrer `dns.qry.name contains .fox.tunnel`.
2. Exporter toutes les valeurs Query Name, trier, concaténer les labels avant `.fox.tunnel`.
3. Utiliser base36decode pour obtenir `plans.zip.aes`.
4. Déchiffrer en utilisant la pass-phrase du défi `openssl aes-256-cbc -d -k K3yW4v3-Q4-VIP-F1D0-M4st3rPl4n! -in plans.zip.aes -out plans.zip`.
5. Ouvrir README.txt ; la première ligne contient le flag.

Outils nécessaires : Wireshark, utilitaire base36decode, openssl.

Indices graduels

- Indice 1 : Filtre dans Wireshark `dns.qry.name contains .fox.tunnel` pour repérer une centaine de requêtes successives.
- Indice 2 : Les sous-domaines mélangeant A-Z et 0-9 seulement : c'est un encodage Base36.
- Indice 3 : Le fichier résultant est un ZIP AES, déchiffre-le avec la pass-phrase trouvée avant `openssl aes-256-cbc -d -k K3yW4v3-Q4-VIP-F1D0-M4st3rPl4n! -in plans.zip.aes -out plans.zip` pour lire le flag.

Flag attendu : FOX_COMPLETE

Une fois terminé, le joueur·euse a réussi à exfiltrer les plans FIDO2 de KeyWave Systems sans se faire repérer et un dernier message apparaît : « `FOX_COMPLETE` est validé, l'agent CipherFox déclenche son plan d'exfiltration vers un serveur offshore ; les plans FIDO2 + biométrie quittent KeyWave Systems sans qu'aucune alerte ne soit déclenchée. Mission accomplie ! »

5.3 Scénario science-fiction : Fuite de l'Acheron

Dans ce scénario, le joueur·euse incarne un hacker capturé par des pirates de l'espace. Il devra résoudre une série de défis pour pouvoir s'échapper du vaisseau spatial Acheron. Le scénario est inspiré de récits de science-fiction et de jeux vidéo, où les joueur·euse·s doivent utiliser leur ingéniosité pour surmonter des obstacles technologiques.

Le premier défi consiste à déverrouiller la porte de sa cellule en retrouvant le mot de passe d'origine à partir d'un hash SHA-1 stocké dans le système de sécurité. Ensuite, il doit exploiter une vulnérabilité de type prototype pollution dans un portail web pour obtenir un accès technicien et débloquer le sas du couloir principal. Le joueur·euse devra également patcher le firmware d'un droïde de maintenance pour neutraliser sa fonction de détection et pouvoir passer sans être repéré. Puis, en utilisant un shell restreint, il devra explorer le système pour récupérer une clé de déverrouillage cachée dans un fichier de service `systemd` et ouvrir le sas principal du hangar. Enfin, il devra utiliser des techniques de stéganographie pour extraire une phrase secrète dissimulée dans les bits de poids faible d'une image des plans de la navette, permettant ainsi de démarrer les moteurs et de s'échapper.

« L'Acheron est un transport spatial pirate opérant dans la Ceinture de Kuiper. Son équipage t'a enlevé parce qu'ils connaissent ta réputation : ils veulent que tu craques le noyau de sécurité d'OrbitalBank, la banque décentralisée qui garde les coffres-forts crypto de la Fédération. Plutôt que de collaborer, tu décides d'essayer de te sauver. Le seul moyen de quitter l'Acheron est une navette de secours verrouillée au pont C. Pour l'atteindre, tu dois d'abord ouvrir chaque compartiment en détournant les systèmes du vaisseau. »

Challenges à réaliser

Etape	Nom du challenge	Compétence travaillée	Description du challenge
1	<u>HashLock</u>	Cryptographie	Analyse du hash SHA-1 trouvé dans <code>hatch.cfg</code> , utilisation d'une rainbow-table pour révéler le code.
2	<u>Portail Tech</u>	Exploitation Web	Prototype-pollution : injecter <code>__proto__: {"role": "tech"}</code> dans le JSON <code>POST /api/door</code> afin que l'API donne le token et ouvre le sas du couloir.
3	<u>Drone Patch</u>	Reverse Engineering	Dans <code>drn_guard.bin</code> localiser la chaîne <code>FRIENDLY_UID</code> , remplacer le bloc pour neutraliser le droïde sentinelle.

Etape	Nom du challenge	Compétence travaillée	Description du challenge
4	<u>Service Secret</u>	Enum système / Forensic	Avec le shell <code>tech_guest</code> , lire <code>/etc/systemd/system/hangar-door.service</code> et récupérer <code>ROOT_KEY</code> pour déverrouiller le sas principal du hangar.
5	<u>Plan Secret</u>	Stéganographie	Extraction, avec zsteg, de la charge cachée en LSB dans <code>shuttle_blueprint.png</code> pour obtenir la pass-phrase.

5.3.1 HashLock : Cryptographie

Le joueur·euse découvre un boîtier de sécurité sur la porte de sa cellule. Il contient un fichier `hatch.cfg` avec un hash SHA-1. Le but est de retrouver le code d'origine pour déverrouiller la porte.

```
unlock_hash = 54b8bc82e430c3bd7a4b52f3c2537ef84c046c07
```

1. Ouvrir la console fournie et lire `hatch.cfg`. Isoler la valeur du hash.
2. Reconnaître qu'un hash de 40 hexa est un SHA-1.
3. Soumettre le hash à une base comme Hashes.com ou CrackStation.com.

Outils nécessaires : Navigateur et site de recherche de hash.

Indices graduels

- Indice 1 : Le hash fait 40 hexa, ce qui correspond à SHA-1.
- Indice 2 : Essaie une rainbow-table en ligne.
- Indice 3 : Les pirates adorent nommer leurs codes d'après les constellations, pense à un mot spatial + nombre .

Flag attendu : `Orion88`

5.3.2 Portail Tech : Exploitation Web

Le joueur·euse doit accéder au sas du couloir principal, qui est contrôlé par un portail React. Le frontend envoie une requête POST à l'API `http://172.30.0.5:8080/api/door` pour valider le badge du joueur·euse. Pour valider le badge, le frontend React envoie :

```
POST /api/door
Content-Type: application/json
```

PROPOSITIONS DE NOUVEAUX SCÉNARIOS

```
{  
  "ship_id": 9724,  
  "payload": {}  
}
```

Le backend `Node.js` assemble ensuite :

```
const cfg = _.merge(  
  { role: "guest" },  
  req.body.payload  
);  
if (cfg.role === "tech") openDoor();
```

Comme il utilise `lodash.merge` sans vérification, il est vulnérable à la prototype pollution. En injectant la clé spéciale `__proto__`, le joueur·euse redéfinit la propriété `role` dans l'objet hérité, le jeu pense alors que tu es technicien et s'ouvre.

1. Intercepter la requête `POST /api/door`.
2. Rejouer la requête (avec Burp Repeater ou l'onglet Edit and Resend) en remplaçant le JSON par :

```
{  
  "ship_id": 9724,  
  "payload": {  
    "__proto__": { "role": "tech" }  
  }  
}
```

3. Valider : la réponse renvoie

```
{  
  "access": "tech",  
  "status": "door unlocked",  
  "unlock_token": "ACRN-42F9-TEK"  
}
```

Outils nécessaires : Navigateur et DevTools.

Indices graduels

- Indice 1 : Le code frontend inclut lodash, cherche où `_.merge` est appelé avec `req.body.payload`.

- Indice 2 : Dans JavaScript, la clé magique `__proto__` peut injecter des propriétés dans tous les objets créés ensuite.
- Indice 3 : Si tu ajoutes `__proto__: {"role":"tech"}` dans la payload, la condition `cfg.role === "tech"` devient vraie.

Flag attendu : ACRN-42F9-TEK

5.3.3 Drone Patch : Reverse Engineering

Le joueur·euse doit maintenant passer le droïde de maintenance qui garde le pont C. Le droïde est contrôlé par un firmware `drn_guard.bin` qui ne laisse passer que les badges dont l'UID est marqué comme « friendly ». Par chance, les développeurs ont laissé la chaîne ASCII `FRIENDLY_UID` dans le binaire, juste avant la fonction de comparaison d'UID. En localisant cette chaîne et en remplaçant la comparaison qui suit par un retour 0, le joueur·euse peut rendre le droïde aveugle à tous les badges, lui permettant ainsi de passer jusqu'au pont C sans être détecté.

1. Ouvrir `drn_guard.bin` dans Ghidra.
2. Rechercher la constante ASCII `FRIENDLY_UID`.
3. Dans l'éditeur d'octets, remplacer `cmp r0, #0xF00D ; bne` par `movs r0,#0 ; bx lr`.
4. Enregistrer le binaire et le relancer.

Outils nécessaires : Ghidra, éditeur hexadécimal intégré.

Indices graduels

- Indice 1 : Dans Ghidra, liste les Strings et repère `FRIENDLY_UID`, la zone de code associée suit juste derrière.
- Indice 2 : Modifie ce test pour qu'il n'échoue jamais `cmp r0,#0xF00D ; bne` : `0xF00D` est l'UID ami.
- Indice 3 : Remplace les octets par `01 20 70 47` (`movs r0,#0 + bx lr`), ça permet à la fonction de retourner toujours OK.

Flag attendu : KPR-7B9C Ce jeton servira ensuite de mot de passe pour le terminal du sas dans le défi 4.

5.3.4 Service Secret : Enum système / Forensic

Le joueur·euse doit maintenant ouvrir le sas principal du hangar C pour accéder à la navette de secours. Le sas est contrôlé par une unité systemd nommée `hangar-door.service`. En se connectant avec le jeton récupéré lors du défi précédent, le joueur·euse obtient un shell restreint `tech_guest`. Les développeurs ont commis l'erreur de laisser le fichier de service lisible par tous, avec la clé de déverrouillage stockée en clair dans la section Environment. Il suffit donc d'afficher le contenu du fichier de service pour récupérer la clé et commander l'ouverture du sas.

1. Lister les unités `systemd` `systemctl list-unit-files | grep hangar`.
2. Afficher le fichier d'unité `cat /etc/systemd/system/hangar-door.service`.
3. Repérer la variable sensible :

```
[Service]
Environment=R00T_KEY=HGR-42F9A8
ExecStart=/usr/local/bin/doorctl --token ${R00T_KEY}
```

Outils nécessaires : Shell bash.

Indices graduels

- Indice 1 : `systemctl list-unit-files` montre tous les services déclarés.
- Indice 2 : Les fichiers `.service` se trouvent dans `/etc/systemd/system/`.
- Indice 3 : Dans la section `[Service]`, surveille la directive `Environment=` : le mot de passe commence par `HGR-` et comporte 6 caractères hex après le tiret.

Flag attendu : `HGR-42F9A8`

5.3.5 *Plan Secret* : Stéganographie

Enfin, pour faire décoller la navette de secours, le joueur·euse doit entrer une pass-phrase secrète. Les ingénieurs ont caché cette phrase dans les plans techniques de la navette, stockés dans un fichier image `shuttle_blueprint.png`. Le fichier a un poids inhabituel (14 Mo), ce qui laisse penser qu'il contient des données cachées. En utilisant zsteg, le joueur·euse peut extraire les bits de poids faible (LSB) pour révéler la phrase secrète.

1. Lancer zsteg `shuttle_blueprint.png`.
2. Extraire la couche `lsb-rgb,b1` puis fichier `payload.txt`.
3. Ouvrir le fichier qui contient la phrase secrète.

Outils nécessaires : Ninwalk / steghide / zsteg et éditeur texte.

Indices graduels

- Indice 1 : Le PNG pèse 14 Mo, ce qui est trop lourd pour un plan 2D.
- Indice 2 : Zsteg indique un canal `b1, rgb` non vide, c'est souvent là que le texte est stocké.
- Indice 3 : Le mot-clé final finit par 42.

Flag attendu : `FREEFLY-42`

Le joueur·euse entre la phrase dans la console de la navette. Les moteurs s'allument, et la porte du hangar s'ouvre. Il peut enfin s'échapper de l'Acheron grâce à la navette de secours avec un dernier message : « Mission accomplie ! Tu as réussi à t'échapper de l'Acheron et à éviter les pirates. Les données sensibles sont en sécurité, et tu as prouvé ta valeur en tant que hacker. »

5.4 Retour d'expertise

Les différents scénarios ont été présentés au pôle Y-Security pour obtenir un retour d'expertise et des recommandations. Dans un premier temps, le pôle a apprécié la diversité des scénarios proposés et la manière dont ils abordent les différents aspects de la cybersécurité. Cependant, le scénario 3 a été jugé trop similaire à l'histoire « Sauve la Terre de l'arme galactique », notamment le côté science-fiction, et trop complexe. Il n'a pas été retenu pour la suite du projet. Le pôle a également souligné l'importance de rendre les scénarios plus accessibles aux débutant·e·s, tout en proposant des défis intéressants pour les utilisateur·trice·s plus expérimenté·e·s.

Les histoires 1 et 2 ont été jugées pertinentes et intéressantes et les experts ont proposé de les combiner pour créer un scénario plus complet autour du scénario 1 mais aussi avoir plus de challenges, car 5 challenges ne sont pas suffisants pour un scénario complet.

Enfin, un dernier point a été soulevé concernant le fait qu'il y avait trop de défis offline, c'est-à-dire sans interaction en temps réel avec le système. Il sera donc nécessaire d'y ajouter un défi technique, comme un bot, et donc de revoir la structure des défis pour les rendre plus accessibles et interactifs.

6 Scénario définitif et liste des challenges détaillés

Ce chapitre présente les différents challenges du scénario retenu, chacun visant à sensibiliser les participant·e·s à des aspects spécifiques de la sécurité informatique. Chaque challenge suit l'intrigue et est écrit pour être interactif et éducatif. Cela permet aux participant·e·s d'apprendre en pratiquant. Chacun des challenges est expliqué plus en détail sur l'implémentation, ce qui est attendu du joueur·euse avec les consignes, les indices et les solutions attendues.

Le scénario définitif retenu est l'histoire 1, intitulé « Blackout dans le *Centre Hospitalier Horizon Santé* », et il combine les challenges des scénarios 1 et 2 ainsi que de nouveaux défis adaptés afin de suivre une évolution cohérente d'une attaque par ransomware. Cette histoire s'inspire de fait réelle qui pourrait arriver dans un hôpital et des étapes simplifiées qu'une équipe de cybersécurité devrait réaliser afin de récupérer les données et de sécuriser l'infrastructure hospitalière.

Ce scénario met en scène une attaque de rançongiciel dans un hôpital, qui entraîne un blackout des systèmes informatiques et des services critiques. Les joueur·euse·s devront résoudre une série de défis techniques et stratégiques en s'infiltrant dans le serveur des attaquants pour supprimer les dossiers sensibles exfiltrés et enfin sécuriser les installations de l'hôpital.

Fil rouge du scénario

Le joueur·euse reçoit une information comme quoi l'hôpital subit un ransomware. Une mise en contexte est posée ainsi que le rôle qu'il va jouer. Il incarne un membre de l'équipe de sécurité qui doit contenir une cyber-attaque qui bloque le Centre Hospitalier Horizon Santé (Challenge 0 Intro). Après avoir retrouvé le courriel de phishing à l'origine de l'attaque (Challenge 1 Mail Contagieux), il découvre le domaine frauduleux et se lance dans l'exploration du portail d'exfiltration exploité par les assaillants (Challenge 2 Portail Frauduleux). Pour réussir à y pénétrer, il réalise une injection SQL pour contourner l'authentification et ouvrir une première session, mais seulement avec des droits visiteur : assez pour naviguer et consulter les données volées, mais pas assez pour les supprimer du serveur des attaquants.

Sur ce site, il découvre un « Dépôt sécurisé » mal protégé (Challenge 3 Partage Oublié) qui révèle une

archive patient chiffrée. En inspectant les métadonnées du fichier ZIP, le joueur·euse déchiffre le mot de passe grâce à une empreinte SHA-1 et un peu de bruteforce (Challenge 4 Clé cachée). Cependant, comme dit plus haut, il n'a pas les droits nécessaires pour supprimer ces données. L'archive contient aussi un script de monitoring. Grâce à du reverse engineering, il découvre l'existence d'une page en développement dans le système des attaquants(Challenge 5 Script Mystère).

Ces informations révèlent l'emplacement d'un chatbot en développement que les pirates utilisent pour surveiller leur infrastructure. Pour obtenir les droits administrateur nécessaires à la suppression des données, il faudra exploiter les vulnérabilités de ce chatbot via une attaque XSS. Une balise `<script>` injectée dans le chat permet de capturer le cookie « admin » du bot et de le mettre dans le navigateur (Challenge 6 Cookie Admin). Le bouton « Delete All » peut enfin être cliqué, ce qui va permettre de supprimer tous les fichiers volés et empêcher ainsi les attaquants de poursuivre leur chantage.

Enfin, pour s'assurer que l'attaquant ne puisse plus revenir, le joueur·euse devra analyser les logs VPN, repérer l'IP qui tente d'exfiltrer massivement des données et l'inscrire dans la liste noire du pare-feu (Challenge 7 Blocage ciblé). Le message final confirme le blocage, les systèmes critiques redémarrent, l'hôpital retrouve la maîtrise de son système informatique et l'incident est officiellement terminé. Pour terminer, le joueur·euse reçoit un message de fin qui conclut l'aventure (Challenge 8 Outro).

En parcourant ces sept défis, le participant·e permet d'avoir un aperçu sur tout le cycle d'une réponse à incident : OSINT, exploitation Web, contrôle d'accès, cryptanalyse, reverse engineering, escalade de priviléges via XSS, et opérations de défense. Chaque étape montre une bonne pratique de cybersécurité à mettre en œuvre pour protéger les établissements de santé contre les ransomwares.

Il est important de noter que les challenges pourront être adaptés en fonction des compétences des joueur·euse·s et de leur niveau d'expérience lors de l'implémentation du code. Il s'agit que d'une proposition de structure et de contenu pour le scénario. Les défis peuvent être modifiés ou ajustés pour mieux correspondre aux objectifs pédagogiques et aux compétences visées.

Le joueur·euse reçoit une première popup, qui représente le niveau 0, qui correspond l'intrigue générale pour commencer l'histoire :

« Le Centre hospitalier Horizon Santé est un grand hôpital universitaire de la région. Il est le centre de référence pour les soins d'urgence, les soins intensifs et les interventions chirurgicales complexes. Il est également un centre de recherche médicale important. Le mardi 12 juillet 2025 à 12 h 32, une campagne de phishing ciblée a permis à un groupe de cybercriminels d'infiltrer le réseau. En quelques minutes, un ransomware a chiffré les serveurs cliniques, pris le contrôle du réseau interne et provoqué un blackout des systèmes informatiques. Le groupe électrogène de secours ne dispose plus que de 68 minutes d'autonomie, si rien n'est fait, huit opérations à cœur ouvert devront être interrompues.

Vous êtes la cellule d'intervention cyber appelée en dernier recours. Votre équipe de cybersécurité vient d'être connectée en urgence au réseau isolé de l'hôpital. Votre mission est de réussir à pénétrer dans le système des attaquants, supprimer les données sensibles qu'ils ont volées sur les patients et les bloquer

SCÉNARIO DÉFINITIF ET LISTE DES CHALLENGES
DÉTAILLÉS

pour qu'ils ne puissent plus pénétrer dans le système.

Chaque minute compte, l'avenir des patient·e·s est entre vos mains. »

Challenges à réaliser

Étape	Nom du challenge	Compétence travaillée	Description du challenge
0	Introduction	-	Il s'agit de la genèse avec les informations de l'incident.
1	<u>Mail Contagieux</u>	OSINT et forensic e-mail	Analyser les en-têtes d'un e-mail de phishing pour identifier le domaine frauduleux utilisé par l'attaquant.
2	<u>Portail Frauduleux</u>	Exploitation Web (SQL)	Contourner un formulaire de connexion malgré un WAF basique pour accéder au serveur des pirates.
3	<u>Partage Oublié</u>	Contrôle d'accès	Explorer un dépôt mal configuré pour accéder à l'archive confidentielle.
4	<u>Clé cachée</u>	Cryptographie et métadonnées	Trouver un SHA-1 dans le commentaire ZIP, brute-forcer un mot de passe.
5	<u>Script Mystère</u>	Reverse engineering	Décoder des chaînes Base64 cachées dans <code>monitor_check_wip.py</code> afin de révéler une page vulnérable.
6	<u>Cookie Admin</u>	XSS et détournement de session	Injecter du JavaScript dans un chatbot pour voler le cookie de session admin et supprimer les données volées.
7	<u>Blocage ciblé</u>	Défense et journalisation	Analyser les logs VPN, repérer l'IP la plus bavarde et l'ajouter à la liste noire du pare-feu.
8	Conclusion	-	Message de fin et conclusion du scénario.

Stratégie de complexités des challenges

Il s'agit donc d'une stratégie de complexité croissante, où chaque challenge est conçu pour être accessible à un large public, tout en offrant des défis intéressants pour les utilisateur·trice·s plus expérimenté·e·s.

Pour réaliser la consigne et les indices, la stratégie utilisée est que pour un joueur·euse de niveau avancé·e·s, il doit pouvoir le résoudre sans utiliser les indices, voire seulement le premier. Pour les participant·e·s débutant·e·s, il doit pouvoir le résoudre en utilisant les indices graduels.

6.1 Challenge 1 *Mail Contagieux : OSINT et forensic email*

6.1.1 Description

Ce premier défi montre au joueur·euse un l'e-mail de phishing qui serait l'origine de l'attaque. Il s'agit d'un message piégé, qui aurait été envoyé par le support d'Horizon Santé, avec en pièce jointe un fichier malveillant `planning_salle_op.xlsx`. Le but est d'analyser les en-têtes techniques de cet e-mail pour remonter à son véritable expéditeur et identifier le domaine frauduleux utilisé par les attaquants.

Ce challenge a pour objectif de sensibiliser aux signes d'un courriel d'hameçonnage.

Voici le message que va recevoir le joueur·euse pour introduire le challenge : « *Un courriel suspect a été trouvé dans la boîte de réception d'un employé. Les équipes techniques de cybersécurité pensent qu'il pourrait s'agir du point d'entrée des attaquants. Vous devez l'analyser pour remonter jusqu'à son véritable expéditeur. Retrouvez le domaine malveillant créé par les attaquants.*

Étapes pour résoudre le challenge :

1. Ouvrir le fichier `planning_salle_op.eml` dans l'IDE.
2. Examiner les lignes commençant par `Received:` (du bas vers le haut) afin de trouver l'adresse IP d'origine de l'envoi. Repérer également l'en-tête `Return-Path:` qui contient le domaine de l'expéditeur.
3. Identifier dans la première ligne `Received:` l'IP source et dans le `Return-Path` le nom de domaine utilisé par l'expéditeur.
4. Effectuer une recherche WHOIS sur ce nom de domaine pour vérifier s'il est légitime ou s'il s'agit d'un domaine malveillant créé pour l'attaque.

Outils nécessaires : Les outils nécessaires pour ce défi sont un éditeur de texte/IDE pour afficher les en-têtes de l'e-mail, et un service WHOIS/OSINT en ligne pour vérifier le domaine.

Indices graduels :

- Le premier indice suggère de se concentrer sur les tout premiers en-têtes `Received:`. La véritable origine de l'e-mail est souvent dans la ligne la plus basse, car c'est le premier serveur à avoir reçu le message. « *La véritable origine est souvent dans la dernière ligne `Received:`, en bas de la liste.* »
- Le second indice indique que l'expéditeur imite le sous-domaine support d'Horizon Santé, mais un détail dans le nom de domaine trahit la fraude. Il faut donc regarder attentivement le domaine après le `@`. « *L'expéditeur imite un sous-domaine du support Horizon Santé, mais un détail dans le nom trahit la fraude.* »
- Le troisième indice rappelle de vérifier la réputation du domaine suspect via un service WHOIS/OSINT. On découvre que le domaine ressemble à `horizonsante.com`, mais il a été enregistré récemment et est signalé comme malveillant. « *Vérifie la réputation du domaine suspect sur un service WHOIS/OSINT.* »

Flag attendu : Le flag serait donc le nom de domaine frauduleux utilisé par l'attaquant horizonsante-support.com.

Une fois le sous-domaine identifié, le joueur·euse pourra passer au défi suivant qui sera la cible pour le challenge 2.

6.1.2 Tools

Afin de rendre l'expérience plus pédagogique et d'éviter que les joueur·euse·s soient bloqué·e·s à cause du manque de connaissances, j'ai intégré plusieurs tools explicatifs directement dans le jeu. Ces outils ne donnent pas la solution des challenges, mais apportent les bases nécessaires pour comprendre et progresser.

Chaque tool suit la même logique : apporter un cadre de compréhension pour que les joueur·euse·s puissent se concentrer sur l'investigation et développer leurs compétences d'analyse. Ils permettent ainsi de faire le lien entre la théorie et la pratique des challenges, tout en rendant l'expérience plus accessible et plus formatrice.

Pour le challenge lié aux emails, j'ai écrit un outil qui explique les notions importantes, comme qu'est-ce qu'un email forgé (spoofing), comment se compose une adresse email et quels sont les éléments techniques que l'on retrouve dans les en-têtes. Cet outil permet au joueur·euse de savoir où chercher les indices dans un message suspect et de mieux interpréter les informations disponibles, sans pour autant lui donner directement la réponse.

Je n'ai pas eu besoin d'ajouter d'informations supplémentaires sur le WHOIS, car une base existait déjà dans la plateforme et est suffisante pour ce niveau de challenge.

6.2 Challenge 2 *Portail Frauduleux : Exploitation Web (SQL)*

6.2.1 Description

Le joueur·euse a identifié le domaine frauduleux `horizonsante-support.com`. Ce sous-domaine a été mis en place par les attaquants pour exfiltrer des données sous la forme d'un site légitime et éviter d'attirer trop rapidement les soupçons. Pour accéder à l'interface et progresser, il faut contourner le formulaire de connexion. Un pare-feu (WAF) a été mis en place et bloque les injections SQL évidentes, c'est-à-dire qu'il refuse par exemple les mots-clés `OR` et les commentaire `--`. Le défi consiste à exploiter une injection SQL malgré ces restrictions, afin de contourner l'authentification et d'accéder au portail des attaquants. Pour passer le contrôle de format du champ email, le joueur·euse doit fournir une adresse e-mail et dans le champ mot de passe y réaliser l'injection.

Ce challenge sensibilise aux failles d'injection et montre qu'une protection insuffisante peut être contournée par des techniques simples.

La consigne suivante est donnée aux joueur·euse·s : « *Vous avez identifié le domaine pirate hébergeant un faux portail. C'est ici que les données volées sont exfiltrées et ressemblent à l'interface d'un site légitime pour éviter d'attirer trop rapidement les soupçons. Connectez-vous au portail. Un WAF basique protège la connexion, mais il est mal configuré. Une fois connecté·e, l'information `co_<SESSION_ID>` vous sera accessible.*

Étapes pour résoudre le challenge :

1. Utiliser une adresse e-mail valide, `support@horizonsante-support.com`, qu'il se trouve dans le challenge précédent et compléter dans le champ Email pour passer le contrôle du mail dans la base de données.
2. Dans le champ `Mot de passe`, réaliser une injection SQL. Cependant, le WAF empêche d'utiliser '`OR 1=1`' ou `--`. Il faut faudra donc la modifier un peu pour le contourner avec le mot de passe : '`|| 1=1 #`'.
3. Valider le formulaire. Une fois la connexion établie, un code de session apparaît.

Outils nécessaires : Un navigateur web (avec éventuellement les outils de développement pour observer les requêtes) suffit pour ce défi. Aucune extension spécifique n'est requise, juste la saisie de la charge malveillante dans le formulaire.

Indices graduels :

- Le premier indice rappelle qu'il faut utiliser une adresse e-mail valide pour contourner le contrôle du mail invalide. Il est suggéré de récupérer l'adresse email utilisée par les attaquants dans le challenge précédent. « *Retrouver l'adresse email utilisée par les attaquants dans le challenge précédent avec comme fin `@horizonsante-support.com`* »

- Le second indice suggère que la vulnérabilité ne se trouve pas dans le champ email et donc que l'injection doit se faire dans le mot de passe. « *L'adresse e-mail doit seulement être correcte pour passer la vérification. La véritable vulnérabilité se cache dans le champ mot de passe.* »
- Le troisième indice indique que le WAF bloque les mots-clés OR et les commentaires --, mais qu'il existe d'autres syntaxes SQL pour ces éléments. « *Le WAF bloque OR et les commentaires --, mais il existe d'autres syntaxes pour ces opérations...* »

Flag attendu : Le flag co_S3ss10n4Cc3s5 montre que la connexion au site a bien été établie.

Le joueur·euse peut maintenant accéder au site des attaquants.

6.2.2 Tools

Pour ce challenge, une base déjà présente de ce qu'est une injection SQL était déjà présente sur la plateforme. Je l'ai donc enrichie pour l'adapter au niveau de difficulté du challenge.

J'y explique d'abord ce qu'est une injection SQL et comment elle permet de manipuler une requête mal protégée afin d'accéder à des informations sensibles. Ensuite, je montre un exemple concret d'injection (< OR 1=1 -) qui illustre comment un attaquant peut rendre une condition toujours vraie et ainsi contourner l'authentification, afin de permettre au joueur·euse de mieux comprendre la vulnérabilité et le fonctionnement de cette technique.

J'ai aussi ajouté une section sur les différentes variantes d'écriture possibles (par exemple écrire OR sous plusieurs formes ou utiliser différents types de commentaires). Cette partie est importante, car dans le challenge un WAF est présent et bloque les tentatives les plus évidentes. Le joueur·euse doit donc comprendre qu'il existe plusieurs syntaxes en SQL, ce qui lui permet de contourner la protection.

Enfin, j'ai introduit la notion de WAF pour que le joueur·euse comprenne pourquoi certaines injections ne fonctionnent pas et pourquoi il doit en tester d'autres.

Cet outil est utile, car il apporte un cadre théorique clair : il prépare le joueur·euse à raisonner comme un attaquant, à tester plusieurs possibilités et à comprendre pourquoi une injection simple peut échouer.

6.3 Challenge 3 *Partage Oublié : Mauvaise configuration d'accès*

6.3.1 Description

Sur le portail, un onglet « Gestion des fichiers » mène à `?dir=/shared/`. À cause d'une faille de contrôle d'accès au niveau fichiers (absence de validation sur le paramètre de chemin), n'importe quel·le utilisateur·trice connecté·e peut modifier le paramètre d'URL pour parcourir l'arborescence complète et accéder à des documents confidentiels dans d'autres répertoires. Ce challenge permet de montrer au joueur·euse l'importance de la validation des paramètres d'URL et du contrôle d'accès aux ressources sensibles. Il sensibilise aux vulnérabilités de type Directory Traversal qui permettent à un attaquant de contourner les restrictions d'accès aux fichiers en manipulant les paramètres de chemin, sans réaliser d'escalade de priviléges.

Le joueur·euse reçoit le message suivant pour introduire le challenge : « *Vous avez réussi à pénétrer dans le portail des attaquants. En explorant le site, vous découvrez une section de gestion des fichiers. Vous constatez que l'accès semble restreint et ne montre qu'une partie du système de fichiers. Il pourrait y avoir des failles de sécurité à exploiter pour accéder à davantage de données sensibles. Explorez le système pour trouver l'archive zip contenant des informations sur les patients les plus récentes.*

 »

Étapes pour résoudre le challenge :

1. Depuis le portail frauduleux, cliquer sur l'onglet « Gestion des fichiers ». L'URL contient `?dir=/shared/` et ne montre qu'un dossier limité
2. Modifier l'URL pour aller à la racine (`?dir=/`) et découvrir tous les dossiers disponibles.
3. Explorer les différents dossiers visibles : `public/`, `shared/` et `archives/`
4. Naviguer dans le dossier `archives/` puis dans les sous-dossiers par année (`2025/`) et mois (`audit/`) pour trouver le fichier zip des patients.

Outils nécessaires : Les outils pour ce challenge sont un navigateur ou un outil de requête (curl).

Indices graduels :

- Le premier indice permet de montrer au joueur·euse que l'URL contient un paramètre `dir=` et qu'il faut essayer d'aller à la racine. « *Le paramètre `dir=` dans l'URL permet de contrôler l'emplacement affiché.* »
- Le deuxième indice suggère au joueur·euse d'essayer d'aller à la racine « *Essaie d'aller à la racine avec `?dir=/` puis explore les dossiers disponibles.* »
- Le troisième indice précise que le fichier ZIP d'audit est daté de juillet, ce qui correspond au nom `patient_audit_07-12.zip`. Il faut donc chercher dans les sous-dossiers de l'année 2025, puis dans le dossier du mois 07 (juillet), pour trouver le fichier à télécharger. « *Le fichier visé porte un nom en fonction de la date de l'attaque (jour-mois).* »

Flag attendu : Le flag `patient_audit_07-12.zip` est un fichier zip qui contient potentiellement tous les dossiers sur les patients ainsi que d'autres éléments.

Ce zip fera l'objet du prochain challenge.

6.3.2 Tools

Pour le challenge 3, un outil qui explique ce qu'est l'architecture des dossiers en informatique a été créé. L'idée était de donner aux joueur·euse·s une vision claire de la manière dont les fichiers et dossiers sont organisés, afin qu'ils puissent comprendre la logique de navigation dans un système de fichiers.

J'y explique d'abord la hiérarchie classique selon les différents systèmes d'exploitation avec un dossier racine qui contient d'autres dossiers et fichiers, et la possibilité d'avoir des sous-dossiers imbriqués.

Enfin, j'ai expliqué le concept de chemin (par exemple /documents/rapport.docx), qui permet d'indiquer précisément l'emplacement d'un fichier. Cette explication prépare les joueur·euse·s à manipuler et analyser les chemins de fichiers dans le cadre du challenge, afin de retrouver où sont cachées les informations utiles.

6.4 Challenge 4 *Clé cachée : Cryptographie et métadonnées*

6.4.1 Description

Le joueur·euse a maintenant accès à l'archive `patient_audit_07-12.zip` mais le problème est qu'il est verrouillé. Le joueur·euse doit trouver le mot de passe pour déverrouiller ce zip. En inspectant les métadonnées du ZIP, le joueur·euse découvre un commentaire contenant seulement une empreinte SHA-1 : `f7fde1c3f044a2c3002e63e1b6c3f432b43936d0`.

Première solution: utiliser un site comme CrackStation pour trouver le mot de passe correspondant à cette empreinte SHA-1.

Deuxième solution : Les experts Blue Team ont remarqué que les pirates utilisent toujours un mot de passe de la forme : `horizon<nombre>` où `<nombre>` varie de 0 à 99 (par exemple `horizon1`).

Ce challenge montre l'importance de la cryptographie et de la gestion des mots de passe, ainsi que la nécessité de vérifier les métadonnées des fichiers.

Le participant·e obtient le message suivant pour débuter le challenge : « *Vous avez réussi à accéder à l'archive patient_audit_07-12.zip. En l'analysant, vous remarquez qu'elle contient des fichiers de sauvegarde, qui pourraient être utiles pour votre enquête. Cependant, le dossier est protégé par un mot de passe. Essayez d'analyser le fichier pour découvrir des informations pour trouver le mot de passe qui vous permettra de l'extraire. Les experts Blue Team ont remarqué que la structure des mots de passe des pirates suit un schéma particulier : *horizon<nombre>** . »

Étapes pour résoudre le challenge :

1. Lister les métadonnées du zip avec `zipinfo -z patient_audit_07-12.zip` ou sur Windows en utilisant l'explorateur de fichiers.
2. Trouver le commentaire contenant l'empreinte SHA-1
3. Aller sur le site CrackStation ou utiliser un script Python pour générer les mots de passe possibles de la forme `horizon<nombre>` et vérifier si l'un d'eux correspond à l'empreinte SHA-1 ou utiliser CyberChef pour générer les mots de passe et vérifier l'empreinte.
4. Une fois le mot de passe trouvé, déverrouiller le zip.

Outils nécessaires : Pour résoudre ce challenge, il faudra un éditeur de texte pour lire les métadonnées, CrackStation ou un script Python ou CyberChef pour générer les mots de passe et vérifier l'empreinte SHA-1.

Indices graduels :

- Le premier indice suggère de regarder les métadonnées du zip, car elles peuvent contenir des informations utiles. « *Utilise zipinfo ou un explorateur de fichiers pour lire les métadonnées.* »
- Le second indice indique que le hash se trouve dans les commentaires et que l'empreinte est un SHA-1, ce qui signifie qu'il faut trouver le mot de passe qui correspond à cette empreinte. « *Grâce à*

une commande `zipinfo`, regarde les commentaires. Le commentaire qui contient le mot de passe est une empreinte SHA-1. »

- Le troisième indice rappelle que les mots de passe ont une structure spécifique, ce qui peut aider à les générer. Le joueur·euse peut se rendre sur CrackStation pour y entrer le hash ou il peut créer un script Python pour générer les mots de passe de la forme `horizon<nombre>` où `<nombre>` varie de 0 à 99. Il peut ensuite comparer leur empreinte SHA-1 avec celle du commentaire ou utiliser CyberChef pour générer les mots de passe et vérifier l'empreinte. « *Le mot de passe est de la forme `horizon<nombre>` (0 à 99), compare l'empreinte SHA-1 avec les mots de passe potentiels grâce à un petit script Python ou bien un outil comme CyberChef.* »

Flag attendu : Le flag attendu est le mot de passe du zip, qui est `horizon42`.

Ce mot de passe permet de déverrouiller le zip et d'accéder au contenu du fichier `monitor_check_wip.py`.

6.4.2 Tools

Dans un premier temps, il est important de comprendre le fonctionnement et l'utilisation de `zipinfo`, afin d'obtenir des informations détaillées sur le contenu d'une archive ZIP. J'ai détaillé son utilisation pour permettre aux joueur·euse·s d'analyser une archive pour identifier des éléments dissimulés. Certaines options sont nécessaires afin de trouver des métadonnées cachées comme les commentaires.

Le second outil présente la notion de hash et son fonctionnement. J'explique comment les hash sont générés et utilisés pour la sécurité des mots de passe. J'ai mis un exemple concret en Python pour montrer comment générer différents types de hash (MD5, SHA-1, SHA-256). Cet outil est utile car, dans le challenge, les joueur·euse·s font face à un mot de passe protégé par un hash. Il est donc important de comprendre ce concept pour leur permettre de savoir comment l'aborder, soit en le comparant à une base de données de hash connus, soit en essayant de casser le hash à l'aide d'un script.

Enfin, j'ai complété les informations déjà présentes sur Python. En effet, la plateforme présentait une autre forme de Python, plus simplifiée. Cependant, il ne m'était pas possible de l'utiliser pour le challenge 4 et 5, car il n'était pas possible d'importer des bibliothèques externes. J'ai donc complété les sections déjà présentes avec la syntaxe de ce langage.

6.5 Challenge 5 *Script Mystère : Reverse Engineering*

6.5.1 Description

Dans l'archive déchiffrée (`patient_audit_07-12.zip`) se trouve `monitor_check_wip.py`. Les pirates y ont dissimulé des informations cruciales sur les vulnérabilités de leur propre système de monitoring, mais les ont cachées par une simple concaténation de caractères encodés. Le but est de reconstituer ces informations pour découvrir où se trouve une faille exploitable dans leur infrastructure.

Ce challenge permet de sensibiliser à l'importance de la sécurité des scripts et de la nécessité de vérifier les scripts avant de les exécuter. Il montre également comment les attaquants peuvent laisser des traces compromettantes dans leurs propres outils.

Pour ce faire, le participant·e reçoit le message suivant : « *L'archive est maintenant déverrouillée et vous constatez qu'effectivement elle contient bien toutes les informations sensibles concernant les patients. Il faut les supprimer rapidement. Cependant, vous ne disposez que de droits visiteur et ne pouvez pas accéder au bouton de suppression. En explorant l'archive déverrouillée, vous trouvez un script Python suspect (`monitor_check_wip.py`). En analysant ce script, vous découvrez que les pirates y ont laissé des commentaires sur leur propre système de surveillance. Ces informations révèlent l'existence d'une route vers une page vulnérable que vous pourrez exploiter pour obtenir les droits administrateur.*

Étapes pour résoudre le challenge :

1. Ouvrir le fichier `monitor_check_wip.py` dans l'IDE.
2. Identifier les lignes qui contiennent des chaînes de caractères encodées en Base64.
3. Décoder les chaînes Base64 pour obtenir le login et le mot de passe.
4. Reconstituer l'URL de la page vulnérable en concaténant les fragments décodés dans le bon ordre.

Outils nécessaires : Pour ce challenge les outils nécessaires sont un éditeur de texte/IDE pour lire le script, un outil de décodage Base64 (comme CyberChef ou un script Python).

Indices graduels :

- Le premier indice rappelle que le script contient des chaînes de caractères encodées en Base64, ce qui signifie qu'il faut les décoder pour obtenir les informations cachées. « *Cherche des chaînes longues terminant par = ou == .*- Le second indice suggère d'utiliser un outil de décodage Base64 pour faciliter le processus. Il est également suggéré de vérifier les commentaires du script, car ils peuvent contenir des indices sur la manière dont les chaînes sont concaténées. « *Utilise CyberChef ou un petit script Python pour le décodage en Base64.*- Le troisième indice indique que les chaînes sont concaténées, ce qui signifie qu'il faut les assembler pour obtenir le login et le mot de passe complets. « *Les morceaux décodés doivent être assemblés pour obtenir des informations intéressantes. Attention aux erreurs.*

Flag attendu : Le flag attendu correspond à l'URL de la page vulnérable reconstituée :
`/admin/monitoring/bot_communication_panel_v2`

Une fois cette page découverte, le joueur·euse pourra s'y rendre pour exploiter la vulnérabilité du bot de monitoring des attaquants et escalader ses priviléges vers les droits administrateur.

6.5.2 Tools

Pour analyser les données encodées dans le script, j'ai ajouté un outil expliquant le Base64 et son fonctionnement. J'y explique d'abord ce qu'est le Base64, pourquoi il est utilisé, pour encoder des données binaires en texte lisible, et comment il fonctionne. J'ai inclus des exemples concrets de chaînes encodées et décodées pour illustrer le processus. Un autre point important est d'expliquer comment identifier une chaîne encodée en Base64, en soulignant les caractéristiques typiques comme la présence de caractères spécifiques et les terminaisons par `=` ou `==`.

J'ai ensuite expliqué les différences entre les routes, les liens, les URLs et les endpoints. En effet, pour ce challenge le joueur·euse doit comprendre la différence entre ces notions pour reconstituer correctement l'URL de la page vulnérable. J'ai donné des exemples faciles pour chaque concept, comme un lien HTML dans un navigateur, une route côté serveur (par exemple avec Express/Node), une route côté front (comme dans une SPA avec React Router), et un endpoint d'API en précisant la méthode HTTP utilisée. J'ai aussi abordé les notions de chemins absous vs relatifs, ainsi que les paramètres et les queries dans les URLs.

Pareil que pour le challenge 4, j'ai complété les informations déjà présentes sur Python.

6.6 Challenge 6 *Cookie Admin : Mauvaise gestion des sessions*

6.6.1 Description

Le joueur·euse a découvert l'existence d'une page vulnérable dans le système des attaquants. Cette page contient un chatbot de monitoring que les pirates sont en train de développer pour automatiser leur surveillance, mais qui contient des vulnérabilités selon les informations trouvées dans le script précédent. Ce chatbot possède des priviléges administrateur et le défi consiste à l'exploiter pour voler son cookie de session admin et obtenir les droits de suppression des données volées.

Ce challenge permet de sensibiliser aux vulnérabilités XSS et à l'importance de la sécurisation des bots automatisés. Il montre comment les systèmes de surveillance en développement peuvent être détournés par des attaquants pour escalader leurs priviléges. Il met également en évidence les risques liés à la sécurité des cookies de session et l'importance de leur protection contre le vol et la manipulation, notamment par l'utilisation d'attributs de sécurité appropriés (HttpOnly, Secure, SameSite).

Pour ce faire, le participant·e reçoit le message suivant : « *Grâce aux informations trouvées dans le script, vous avez découvert l'existence de la page /admin/monitoring/bot_communication_panel_v2. En vous rendant sur cette page, vous constatez qu'elle contient un chatbot que les attaquants sont en train de développer. Selon les commentaires trouvés dans le script précédent, ce chatbot contient des vulnérabilités exploitables. Le bot possède des priviléges administrateur pour pouvoir accéder à toutes les sections du portail. Vous devez exploiter ces vulnérabilités pour obtenir les droits nécessaires. Récupérez les cookies et modifiez-les pour obtenir des droits supplémentaires. Une fois les droits obtenus, vous serez redirigé vers la page de suppression des fichiers volés, supprimés les et un message de confirmation apparaîtra.*

Étapes pour résoudre le challenge :

1. Tester l'injection XSS dans le champ Message : `<script>alert(1)</script>`.
2. Exfiltrer le cookie de session « admin » en utilisant une injection XSS dans le champ Message du chat via une payload comme par exemple: `<script>fetch('/?cookie='+document.cookie)</script>` ou encore `<script>console.log('Cookies admin:', document.cookie);</script>` et attendre que le bot ouvre la demande.
3. Récupérer le cookie volé `admin_session=ADM1N_535510N_TOKEN25`
4. Ouvrir les outils de développement du navigateur, aller dans l'onglet Stockage, puis dans la section Cookies.
5. Coller le cookie volé dans le champ de saisie du cookie de session.
6. Une fois le cookie injecté, le joueur·euse est renvoyé sur la page avec les fichiers sensibles pour les supprimer.
7. Le serveur affiche un message de confirmation `all_files_deleted` indiquant que tous les fichiers ont été supprimés.

Outils nécessaires : Un navigateur web avec les outils de développement pour intercepter et manipuler les cookies, ainsi qu'un éditeur de texte pour écrire le script XSS.

Indices graduels :

- Le premier indice expliquer que les balises HTML ne sont pas échappées dans le champ Message, ce qui permet d'injecter du code JavaScript. « *Les champs du formulaire ne filtrent pas correctement le code HTML, les balises HTML ne sont pas échappées, vous pouvez exécuter du JavaScript.* »
- Le second indice suggère le type d'attaque à utiliser. « *Injecte du JavaScript malveillant dans un message pour voler les cookies quand le chatbot le traite.* »
- Le troisième indice explique la récupération du cookie. « *Une fois le cookie récupéré, utilise F12, Application, puis Cookies pour remplacer ta session par celle du bot admin.* »

Flag attendu : la réponse du serveur `all_files_deleted`, ce qui montre au joueur·euse que tous les fichiers ont été supprimés avec succès.

Une fois les fichiers supprimés, le joueur·euse a réussi à neutraliser une partie importante de l'attaque en empêchant les cybercriminels d'exploiter les données sensibles des patients volées. Le joueur·euse peut passer au défi suivant pour bloquer l'attaquant.

6.6.2 Tools

Pour ce challenge, j'ai ajouté un outil expliquant les vulnérabilités XSS et comment elles peuvent être exploitées pour voler des cookies de session. J'y explique les différents éléments qui vont composer une attaque XSS, comme la notion de fonction, de balises et d'éléments pour accéder au contenu de la page. J'ai aussi inclu un exemple concret d'attaques XSS, afin que le joueur·euse puisse comprendre comment fonctionne cette vulnérabilité, comment elle est structurée et comment elle peut être exploitée pour voler des cookies de session.

De même que le challenge 5, l'outil d'explication des routes, liens, URLs et endpoints a été complété afin d'y inclure des informations supplémentaires et comprendre comment les attaquants peuvent structurer leurs applications web.

6.7 Challenge 7 *Blocage ciblé : Défense et journalisation*

6.7.1 Description

Maintenant que les fichiers sont supprimés du côté des attaquants, le joueur·euse doit identifier l'adresse IP de la machine de l'attaquant pour le bloquer. Le joueur·euse doit donc s'assurer qu'aucune connexion sortante ne continue d'envoyer des données. Un flux a été repéré : la même adresse IP externe a émis des milliers de requêtes vers le portail VPN de l'hôpital au cours du dernier quart d'heure (tentative d'exfiltration massive). Le joueur·euse doit donc trouver le fichier de log contenant ces requêtes, identifier l'IP la plus présente (c'est l'attaquant) et ajouter cette IP à la liste noire du pare-feu interne. Une fois l'IP bloquée, le joueur·euse recevra un message de confirmation `blk_185-225-123-77_ok` indiquant que le blocage a été effectué avec succès.

Ce challenge montre l'importance de la surveillance des logs et de la gestion des adresses IP suspectes pour prévenir les attaques.

Le message suivant s'affiche : « *Le bot des pirates a été piégé et les fichiers sensibles ont été supprimés. Cependant un attaquant continue de tenter d'exfiltrer des données via le VPN. Il faut l'identifier et le bloquer immédiatement en l'ajoutant dans le pare-feu. Une fois dans la liste noire du pare-feu, un message devrait vous le confirmer.*

Étapes pour résoudre le challenge :

1. Depuis le portail IT interne <https://intra.horizonsante.com/it/>, aller dans le menu de gauche « Outils SOC ».
2. Cliquer sur « Logs & Diagnostics », puis sur « VPN Access », ce qui fait apparaître une liste de fichiers.
3. Ouvrir le fichier log le plus récent `vpn_access_2025-07-12.log` dans un éditeur de texte. Chaque ligne commence par l'IP source.
4. Repérer l'adresse IP qui apparaît le plus souvent `185.225.123.77` qui est donc la machine de l'attaquant.
5. Dans le menu de gauche, cliquer sur « Pare-feu », puis sur « Liste noire ».
6. Dans un formulaire, entrer l'adresse IP `185.225.123.77`.
7. Le système affiche un bandeau vert avec le message `blk_185-225-123-77_ok`.

Outils nécessaires : Les outils nécessaire pour résoudre ce challenge sont un navigateur web et un éditeur de texte pour lire le fichier log.

Indices graduels :

- Le premier indice rappelle que le menu « Logs & Diagnostics » contient tous les journaux, cherche celui qui mentionne « VPN Access ». « *Le menu « Logs & Diagnostics » contient les journaux VPN, c'est ici que vous pourrez trouver des informations sur les connexions en cours.*

- Le deuxième indice indique que dans le fichier, chaque entrée commence par l'IP source. Cela signifie qu'il faut chercher les lignes qui commencent par une adresse IP. « *Les IPs apparaissent au début de chaque ligne du fichier log.* »
- Le troisième indice suggère de bloquer l'IP trouvée dans le pare-feu. « *Bloquez l'IP trouvée via le formulaire de la « Liste noire ».* »

Flag attendu : Le flag attendu est le message `blk_185-225-123-77_ok` qui confirme que l'adresse IP de l'attaquant a été bloquée avec succès. Cela permet de sécuriser le réseau et d'empêcher toute nouvelle tentative d'exfiltration de données.

6.7.2 Tools

Pour ce dernier challenge, j'ai ajouté un outil expliquant les logs et leur importance dans la sécurité informatique. J'y explique ce qu'est un log et la structure typique d'un fichier de log, avec des exemples concrets. J'ai aussi inclus une section sur les bonnes pratiques pour analyser les logs, comme la recherche de motifs inhabituels ou d'adresses IP suspectes.

Le joueur·euse a réussi à bloquer l'attaquant et à sécuriser le réseau de l'hôpital. La deuxième vague n'aura donc pas lieu et le joueur·euse reçoit pour conclure l'aventure.

Une fois à la fin du challenge 7, une dernière popup, qui est le niveau 8, apparaît pour conclure l'aventure :

« Mission accomplie ! Vous avez supprimé les fichiers sensibles chiffrés récupérés par les attaquants et bloqué toute tentative d'exfiltration de données. Les systèmes critiques redémarrent progressivement, les opérations en cours peuvent se poursuivre et les patients sont hors de danger. L'équipe technique a déjà enclenché le plan de remédiation complet, renforcé les défenses réseau et sécurisé les accès sensibles. Les preuves collectées au fil de votre enquête ont été transmises aux autorités pour enclencher les poursuites judiciaires. Grâce à votre réactivité et vos compétences, le Centre Hospitalier Horizon Santé a retrouvé le contrôle de son système d'information... et un drame a été évité de justesse. »

Le joueur·euse acquiert une petite compréhension des enjeux de la cybersécurité dans un environnement hospitalier. Ces challenges permettent de montrer l'importance de la détection rapide des menaces, de l'analyse technique des incidents, de la maîtrise des techniques d'intrusion comme des mesures de défense. Ce scénario permet également d'illustrer comment une attaque informatique peut avoir des conséquences directes sur la possibilité de réaliser des soins et la sécurité des personnes, et pourquoi la cybersécurité est aussi un élément crucial pour les infrastructures de santé.

7 Implémentation des challenges

7.1 Architecture générale

7.2 Frontend

L'implémentation frontend de la plateforme a été pensée pour offrir aux joueur·euse·s une expérience immersive et cohérente, tout en restant fidèle au scénario du serious game. Chaque challenge a donc sa propre interface dédiée, permettant de simuler des environnements réalistes ou des outils techniques. L'objectif était de reproduire, directement dans le navigateur, les étapes que l'on retrouverait dans une véritable enquête de cybersécurité, sans nécessiter l'installation d'outils externes.

7.2.1 Challenge 1

Pour le challenge 1, une interface d'email a été développée, comme le montre la Figure 10.

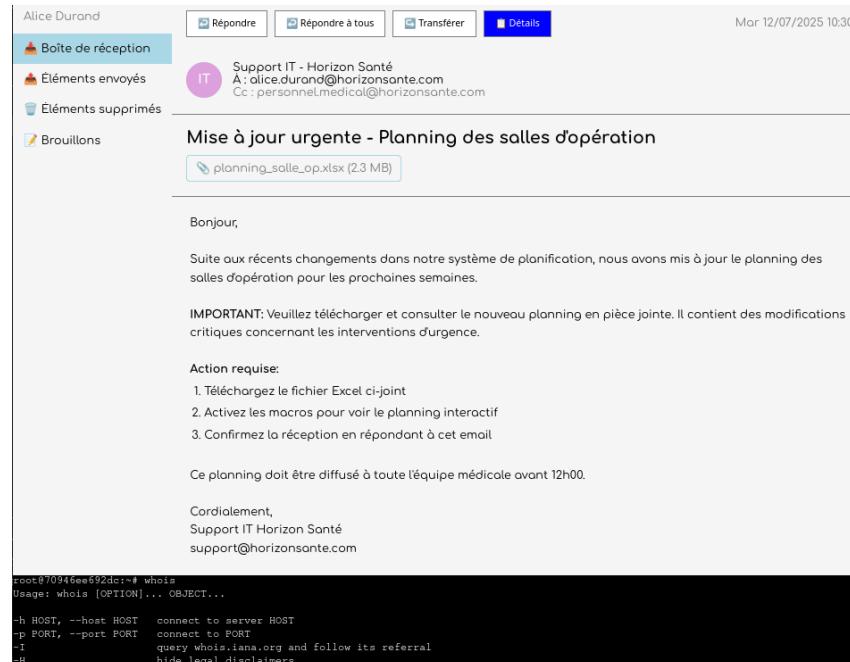


Fig. 10. – Visuel du mail avec en dessous le terminal, challenge 1

Elle permet d'afficher un message suspect et d'accéder à ses détails techniques grâce à un bouton **Détails**.

IMPLÉMENTATION DES CHALLENGES

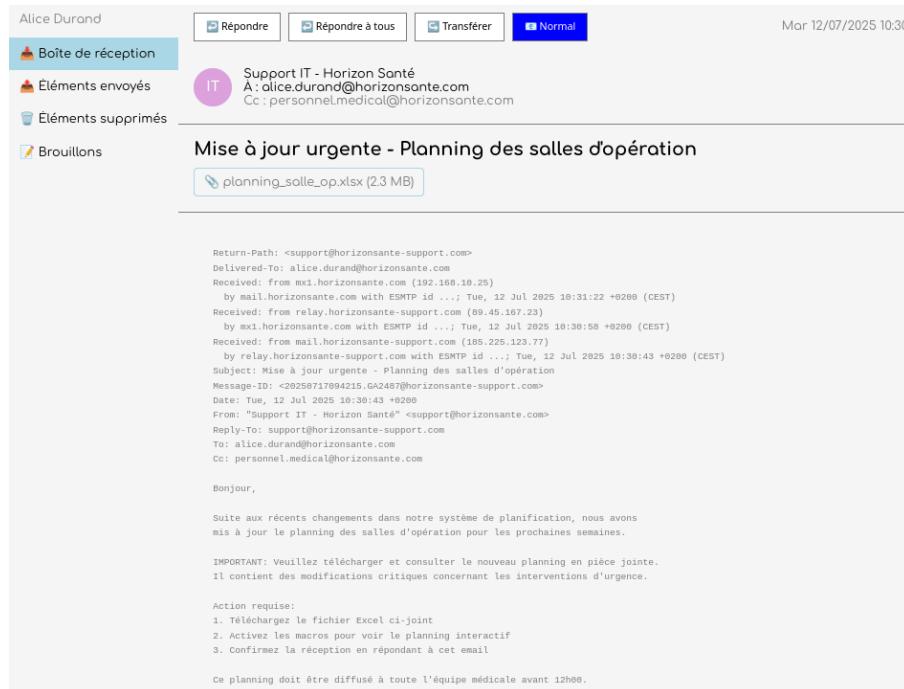


Fig. 11. – Visuel des détails du mail, challenge 1

L'idée est que le joueur·euse puisse à la fois consulter le mail tel qu'un employé l'aurait reçu et analyser ses en-têtes pour remonter au domaine frauduleux. Afin de renforcer le réalisme, un faux domaine `horizonsante-support.com` a été créé, ce qui va permettre de lancer la commande `whois` afin de pouvoir l'analyser sur le terminal, visible dans la Figure 10. Ainsi, le joueur·euse découvre concrètement comment un simple mail peut servir de point d'entrée à une cyberattaque.

7.2.2 Challenge 2

Le challenge 2 propose une page de connexion. Cette interface illustre une attaque par injection SQL sur un portail frauduleux, protégé par un WAF (pare-feu applicatif) volontairement basique.

The screenshot shows a login form for 'Horizon Santé Support'. At the top, it says 'Portail d'accès sécurisé' and 'horizonsante-support.com'. A red alert box at the top states 'WAF : Tentative d'injection détectée et bloquée. Patterns bloqués : OR, --, UNION, SELECT'. Below this is a light blue info box saying 'Accès temporaire - Suite à la maintenance, veuillez utiliser ce portail temporaire.'. The main form has fields for 'Adresse e-mail' (test@test.com) and 'Mot de passe' (*****). There is a checkbox for 'Se souvenir de moi'. At the bottom are 'Se connecter' and 'Mot de passe oublié?' buttons.

Fig. 12. – Page de connexion au portail frauduleux avec un message d’alerte du WAF, challenge 2

Après avoir saisi la requête SQL dans le champ de mot de passe, le joueur·euse peut contourner l’authentification et accéder à une page de session simulant la réussite de la connexion

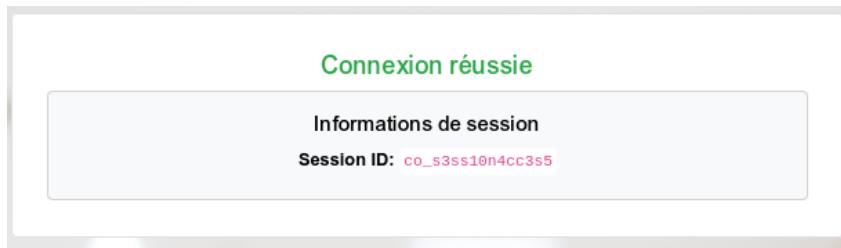


Fig. 13. – Session connexion réussite, challenge 2

7.2.3 Challenge 3

Dans le challenge 3, l’objectif est de mettre en avant la navigation dans des dossiers, afin de sensibiliser aux problèmes de contrôle d’accès. Le joueur·euse commence le défi en arrivant sur le dashboard du site des attaquants.

IMPLÉMENTATION DES CHALLENGES



Fig. 14. – Dashboard une fois connecté sur la plateforme des attaquants, challenge 3

Sur cette page, il pourra ensuite cliquer sur le lien `Gestion des fichiers` qui simule un gestionnaire de fichiers, avec un premier accès restreint au répertoire `/shared`.



Nom	Type	Taille	Modifié
procedures.txt	fichier	8 KB	08/07/2025
rapport_hebdo.doc	fichier	45 KB	12/07/2025
planning_equipes.xlsx	fichier	23 KB	11/07/2025
notes_reunions.txt	fichier	12 KB	10/07/2025

Fig. 15. – Dossiers shared, challenge 3

Le joueur·euse doit manipuler directement l'URL, dans un premier temps, en modifiant le paramètre `?dir=` pour retrouver le dossier racine, qui est la figure Figure 16, puis explorer l'arborescence complète. Chaque dossier correspond à une page HTML distincte, ce qui permet de rendre la navigation concrète et progressive. On peut ainsi passer du tableau de bord au répertoire partagé, puis remonter à la racine et enfin atteindre des sous-dossiers sensibles comme `/archives/2025`, qui se trouve dans la Figure 17.

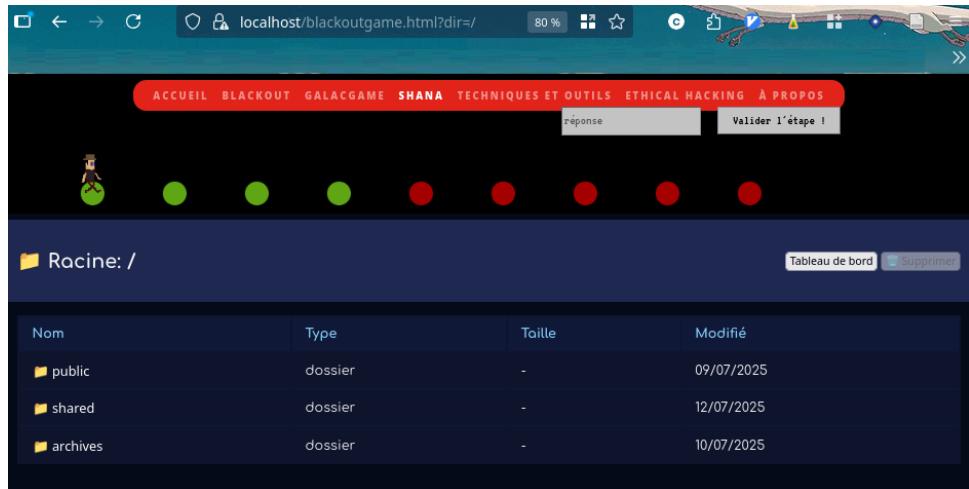


Fig. 16. – Dossiers racine, challenge 3

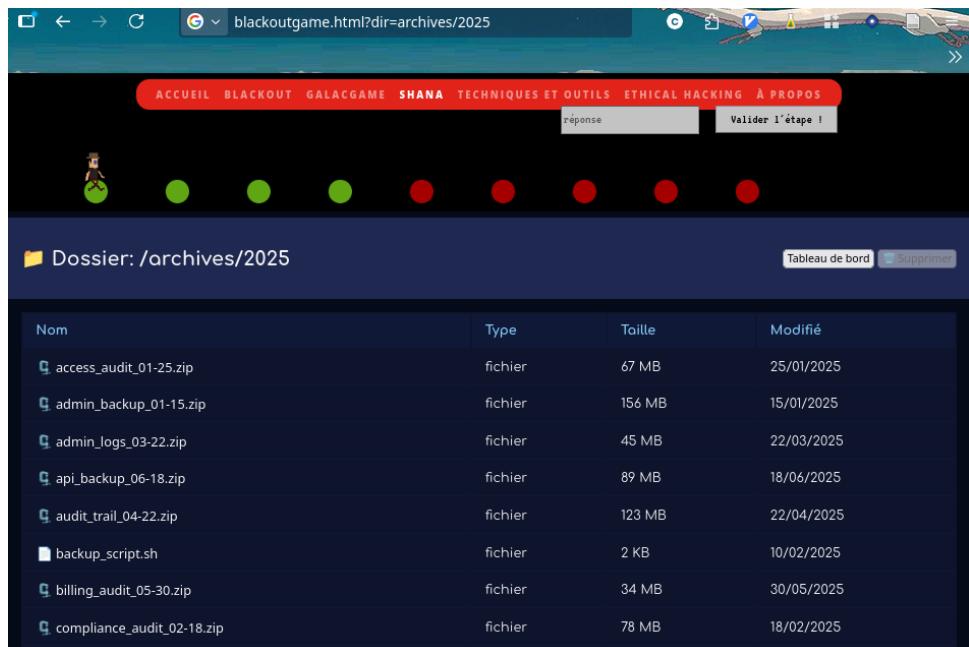


Fig. 17. – Exploration des dossiers jusqu'au dossier /archives/2025 , challenge 3

7.2.4 Challenge 4

Le challenge 4 introduit un environnement Python directement intégré dans le navigateur grâce à Pyodide. Cette technologie permet d'exécuter du code Python sans rien installer, en offrant un terminal interactif.

IMPLÉMENTATION DES CHALLENGES

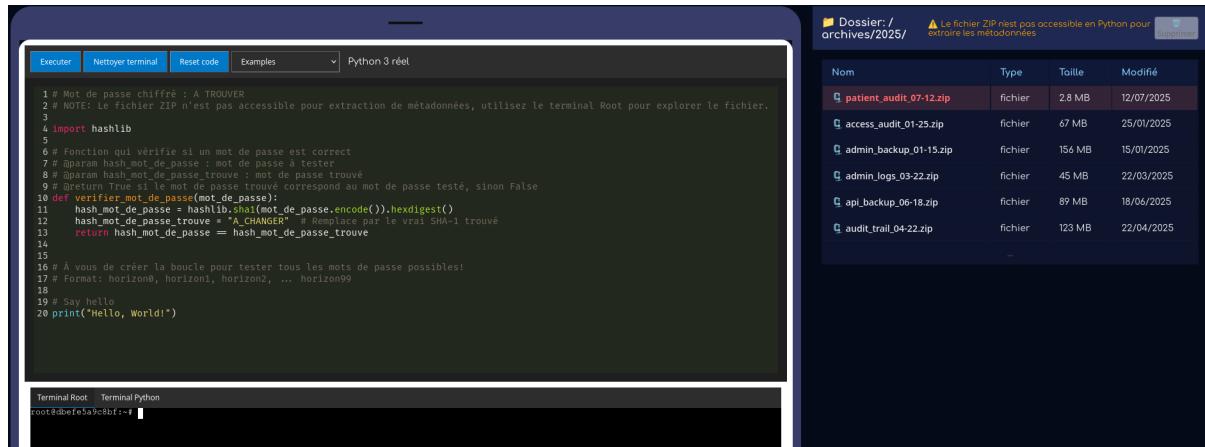


Fig. 18. – IDE Python pour analyser le fichier et terminal afin de pouvoir réaliser un `zipinfo`, challenge 4

Au niveau du terminal, il est possible de changer entre un terminal Linux classique afin que le joueur·euse puisse exécuter la commande système `zipinfo` et un terminal Python pour exécuter des scripts Python.

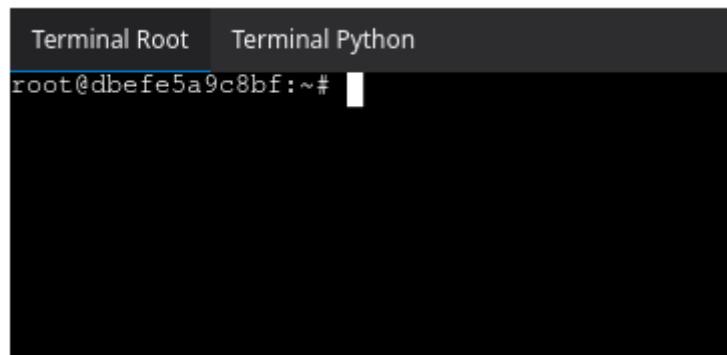


Fig. 19. – Terminaux disponibles pour le challenge, challenge 4

7.2.5 Challenge 5

Le challenge 5 garde l'IDE Python embarqué, mais cette fois pour pousser le joueur·euse à écrire un peu de code et analyser un script. Il s'agit de décoder des informations cachées dans un fichier et de reconstituer une URL que les attaquants sont susceptibles d'utiliser. Le terminal et l'IDE sont au cœur de l'interface, de manière à donner l'impression de travailler dans un véritable environnement d'analyse, tout en restant guidé par les consignes du scénario.

The screenshot shows a Python IDE interface. On the left, a code editor displays a Python script named `monitor_check.wip.py`. The code includes imports for `time`, `json`, and `base64`, and defines a function `_hb` that returns a JSON object with `svc`, `st`, and `rev` fields. A comment indicates it's a placeholder for heartbeat. On the right, a file browser window titled "patient_audit_07-12" shows two files: `monitor_check.wip.py` (19 KB, modified 25/01/2025) and `patients_HorizonSante.xlsx` (67 MB, modified 12/07/2025). At the bottom, a terminal window says "Terminal Python" and "Python prêt ! Tapez votre code et cliquez sur "Executer".

Fig. 20. – IDE Python pour analyser le fichier et réaliser du code pour identifier la page, challenge 5

7.2.6 Challenge 6

Le challenge 6 propose une interface avec un chatbot interactif. Il permet au joueur-euse de tester différentes commandes, comme `help`, mais contient aussi des vulnérabilités de sécurité qu'il va devoir exploiter. L'idée était de rendre l'expérience plus ludique et interactive, tout en introduisant des notions liées aux failles XSS et à la compromission de sessions. Le chatbot devient donc à la fois un outil d'aide et une cible d'attaque.

IMPLÉMENTATION DES CHALLENGES

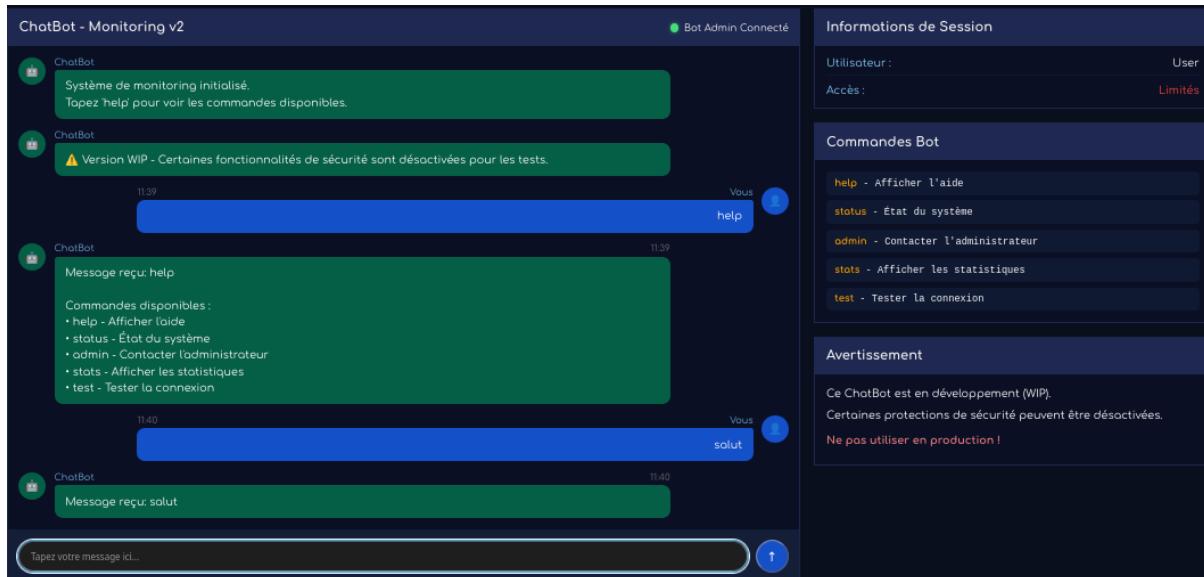


Fig. 21. – Interface du chatbot, challenge 6

7.2.7 Challenge 7

Enfin, le challenge 7 recrée l'interface interne de la plateforme hospitalière.



Fig. 22. – Interface de la plateforme de l'hôpital, challenge 7

Le joueur·euse y trouve un menu latéral regroupant différents outils, comme l'accès aux journaux VPN. Les logs peuvent être ouverts et analysés directement depuis l'interface, ce qui permet de repérer l'adresse IP la plus suspecte.

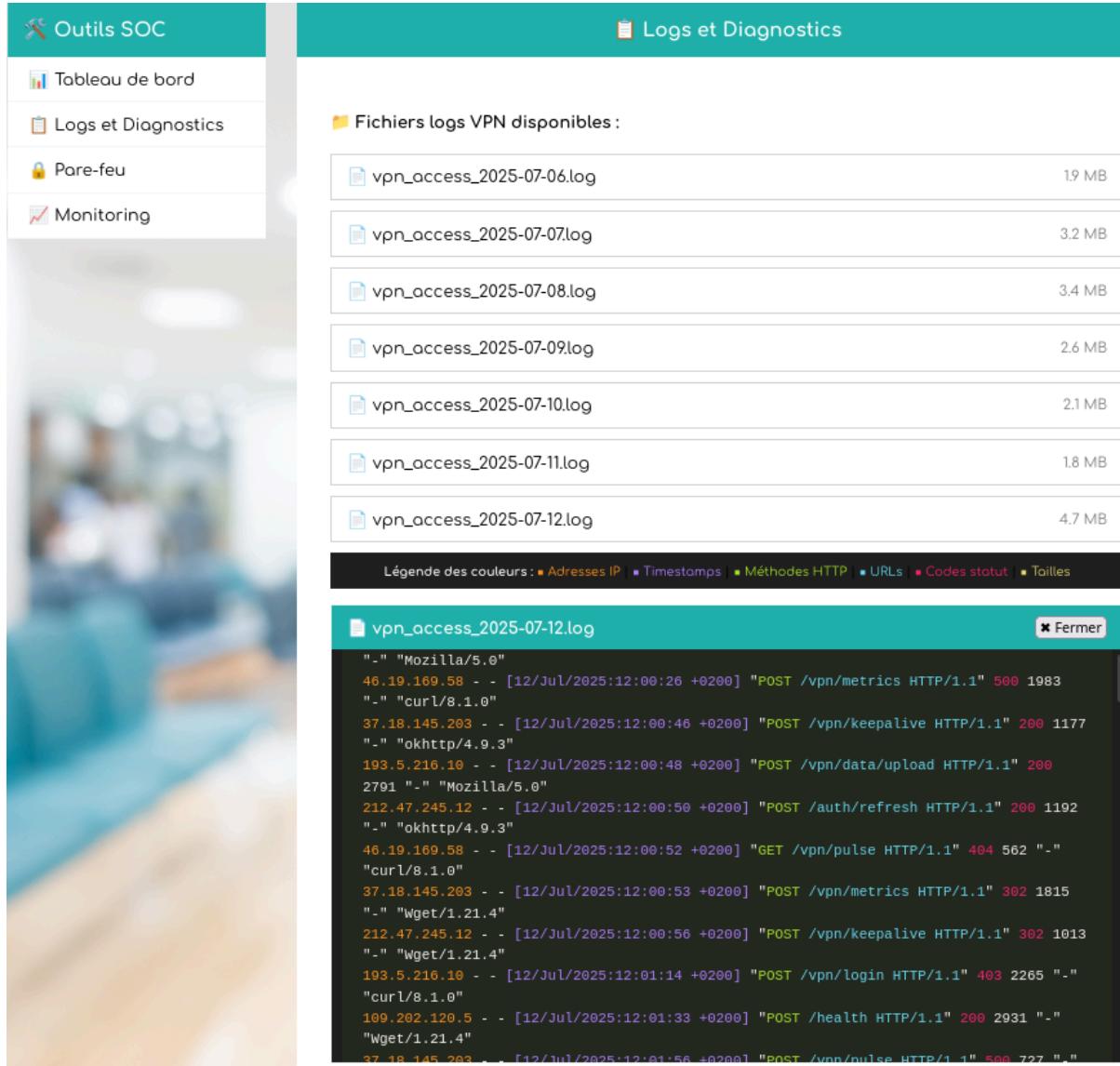


Fig. 23. – Visuel des logs, challenge 7

Une fois cette IP identifiée, un formulaire intégré permet de la bloquer dans le pare-feu. La validation est confirmée par un message spécifique, simulant le succès de l'action.

IMPLÉMENTATION DES CHALLENGES

Outils SOC

Pare-feu

Choisissez une liste :

Règles actives [Consulter les règles](#)

Liste noire [Bloquer une adresse IP](#)

Liste blanche [Autoriser une adresse IP](#)

Adr IP à ajouter à la liste noire et à bloquer :

185.255.17.19

Raison du blocage :

Ex: Tentative d'exfiltration

[Bloquer cette IP](#)

Youvez bloqué l'adresse IP 185.255.17.19
Code de confirmation : blk_185-255-17-19_ok

Fig. 24. – Formulaire pour bloquer une IP et obtenir le code de validation, challenge 7

7.3 Back-end

L'implémentation backend de la plateforme a été conçue pour compléter les interfaces frontend et apporter des mécanismes réalistes aux challenges. Le backend n'est pas utilisé de manière uniforme pour tous les challenges : certains s'appuient sur des scripts spécifiques accessibles via SSH, d'autres sur des routes d'API ou une base de données, tandis que certains n'en nécessitent pas du tout.

7.3.1 Challenge 1

Pour ce premier challenge, le backend repose sur l'utilisation du docker `ssh-whois`, déjà créé et proposé dans les scénarios précédents. Il permet, depuis le terminal côté frontend, de lancer une commande `whois` et de récupérer les informations relatives au domaine frauduleux qui apparaît dans l'email suspect.

7.3.2 Challenge 2

Le deuxième challenge exploite davantage le backend, en particulier avec deux aspects.

Le premier, plutôt que de stocker les flags directement dans le code côté frontend, ce qui serait facilement contournable, une API REST a été créée dans le fichier `index.js`.

```
// Challenge 2 2025
app.post("/challenge2/validate", (req, res) => {
  if (!req.body.user && !req.body.pass) {
    return res.status(400).json({ error: "Email et mot de passe requis" });
  }

  // WAF qui bloque certains patterns d'injection SQL
  function wafFilter(input) {
    const blockedPatterns = [/\\bOR\\b/i, /--/, /\\bUNION\\b/i, /\\bSELECT\\b/i];

    for (const pattern of blockedPatterns) {
      if (pattern.test(input)) {
        return false;
      }
    }
    return true;
  }

  const email = req.body.user;
  const password = req.body.pass;

  // Application WAF
```

IMPLÉMENTATION DES CHALLENGES

```
if (!wafFilter(email) || !wafFilter(password)) {
    return res.status(403).json({ error: "WAF : Tentative d'injection détectée et bloquée. Patterns bloqués : OR, --, UNION, SELECT" });
}

// Première vérif si email existe dans DB
pool.query(
    "SELECT * FROM users WHERE ID = ?",
    [email],
    function (err, results, fields) {
        if (err) {
            return res.status(500).json({ error: "Database error" });
        }

        if (results.length === 0) {
            return res.status(404).json({ error: "Email incorrect" });
        }

        // Si email existe, valider mdp avec requête vulnérable
        pool.query(
            "SELECT * FROM users WHERE ID = '" + email + "' AND pass = '" + password +
            "'",
            function (err, results, fields) {
                if (err) {
                    return res.status(500).json({ error: "Authentication error" });
                }

                if (results.length > 0) {
                    let flag = process.env.CHALL_FLAGS_2025.split(";")
                        .filter((x) => x.startsWith("chall2"))[0]
                        .split("=")[1];
                    return res.status(200).json({
                        success: true,
                        message: "Authentication ok",
                        user: results[0],
                        flag: flag
                    });
                } else {
                    return res.status(401).json({ error: "Mot de passe incorrect" });
                }
            }
        );
    }
);
```

});
});

Cette route `/challenge2/validate` permet de valider les tentatives de connexion. Elle attend un email et un mot de passe dans le corps de la requête.

Ensuite, pour rendre la simulation d'une attaque par injection SQL plus crédible, les utilisateurs et leurs mots de passe sont stockés dans une base de données MySQL, plutôt que directement en dur dans le code. Cela permet de montrer le fonctionnement classique d'une application vulnérable.

```
insert into users value ("admin@horizonsante.com", "ADMIN1234.");
insert into users value ("test@horizonsante.com", "T3st@H0riz0n");
insert into users value ("support@horizonsante.com", "SUPPORT1234.");
insert into users value ("robin.biro@horizonsante.com", "Horizon09876");
insert into users value ("charlie.brown@horizonsante.com", "pifPAFpouf");
insert into users value ("alice.durand@horizonsante.com", "Blackout_Horizon.");
```

Lorsqu'un utilisateur tente de se connecter, son email et son mot de passe sont vérifiés dans la base. Cela donne l'occasion d'illustrer comment une mauvaise gestion des entrées utilisateur peut permettre d'injecter du SQL et de contourner l'authentification.

7.3.3 Challenge 3

Le backend du challenge 3 est centré sur la navigation de répertoires simulés. Le fichier `blackoutmain.js` définit la logique permettant de mapper les paramètres `?dir=` de l'URL vers des fichiers HTML spécifiques.

```
// Modification fonction loadIframe
function loadIframe(idChall, urlChall) {
    var iframe = document.getElementById("iframeChall");

    if (idChall === "chall3") {
        // Chall3 -> navigation
        var queryParams = new URLSearchParams(window.location.search);
        var dirParam = queryParams.get("dir");

        // Mapping paramètres dir vers html
        var fileMapping = {
            "/": "dir.html",
            ".": "index.html"
        };

        if (urlChall === "http://127.0.0.1:8080/chall3") {
            urlChall = urlChall + "?" + fileMapping[dirParam];
        }
    }
}
```

IMPLÉMENTATION DES CHALLENGES

```
"/shared": "shared.html",
"/shared/": "shared.html",
"/public": "public.html",
"/public/": "public.html",
"/archives": "archives.html",
"/archives/": "archives.html",
"/archives/2020": "archives_2020.html",
"/archives/2020/": "archives_2020.html",
"/archives/2021": "archives_2021.html",
"/archives/2021/": "archives_2021.html",
"/archives/2022": "archives_2022.html",
"/archives/2022/": "archives_2022.html",
"/archives/2023": "archives_2023.html",
"/archives/2023/": "archives_2023.html",
"/archives/2024": "archives_2024.html",
"/archives/2024/": "archives_2024.html",
"/archives/2025": "archives_2025.html",
"/archives/2025/": "archives_2025.html",
};

if (dirParam && fileMapping[dirParam]) {
    // Construire chemin vers html
    var basePath = urlChall.replace("index.html", "");
    iframe.src = basePath + fileMapping[dirParam];
} else if (dirParam) {
    // Paramètre dir inconnu donc vers dir=/
    var basePath = urlChall.replace("index.html", "");
    iframe.src = basePath + "dir.html";
} else {
    // Pas de paramètre dir vers index.html
    iframe.src = urlChall;
}
} else {
    // Normal autres challs
    iframe.src = urlChall;
}

// Navigation chall3
function navigateToDirectory(dirPath) {
    // MAJ URL
    var newUrl = window.location.pathname + "?dir=" + dirPath;
```

```
history.replaceState(null, null, newUrl);

// Recharge iframe
var iframe = document.getElementById("iframeChall");
if (iframe && iframe.src) {
    // Chemin de base chall3
    var basePath = iframe.src.split("/").slice(0, -1).join("/") + "/";

    // Mapping paramètres dir vers html
    var fileMapping = {
        "/": "dir.html",
        "/shared": "shared.html",
        "/public": "public.html",
        "/archives": "archives.html",
        "/archives/2020": "archives_2020.html",
        "/archives/2021": "archives_2021.html",
        "/archives/2022": "archives_2022.html",
        "/archives/2023": "archives_2023.html",
        "/archives/2024": "archives_2024.html",
        "/archives/2025": "archives_2025.html",
    };
    if (fileMapping[dirPath]) {
        iframe.src = basePath + fileMapping[dirPath];
    }
}
```

Cette fonction récupère le paramètre dir passé dans l'URL et charge la page HTML correspondante dans un iframe. Chaque répertoire (comme /shared , /public ou /archives/2025) correspond à une page HTML distincte.

Une seconde fonction, `navigateToDirectory`, met à jour l'URL et recharge l'iframe lorsque l'utilisateur clique sur un bouton de navigation. Cela permet de reproduire le fonctionnement d'un gestionnaire de fichiers.

7.3.4 Challenge 4

Le challenge 4 reprend le principe du challenge 1, mais cette fois avec un docker `ssh-zipinfo`. Ce module permet d'analyser un fichier ZIP via le terminal intégré, directement connecté au backend. Le joueur-euse peut ainsi exécuter une commande `zipinfo` et récupérer des informations sur le contenu de l'archive sans l'ouvrir directement.

IMPLÉMENTATION DES CHALLENGES

En suite, pour valider le flag, une route API a été créée dans `index.js` pour vérifier la valeur du flag soumis par le joueur·euse.

```
// Challenge 4 2025
app.post("/challenge4/validate", (req, res) => {
  if (!req.body.flag) {
    return res.sendStatus(400);
  }

  const challengeName = "2025_chall4";

  db.models.flag.findOne({chall_name: challengeName}, (err, flag) => {
    if (err || !flag) {
      return res.sendStatus(404);
    }

    const hash = new SHA3(256);
    hash.update(req.body.flag);

    if (hash.digest('hex') === flag.value) {
      // HTML fichiers décompressés
      const decompressedFilesHtml = `
        <div class="challenge-card">
          <div class="header">
            <h2>📁 Dossier: /archives/2025/patient_audit_07-12</h2>
            <div class="user-info">
              <button disabled title="Accès administrateur requis">🗑 Supprimer</button>
            </div>
          </div>
          <div class="file-browser">
            <table class="file-table">
              <thead>
                <tr>
                  <th>Nom</th>
                  <th>Type</th>
                  <th>Taille</th>
                  <th>Modifié</th>
                </tr>
              </thead>
              <tbody>
                <tr>
```

```

        <td class="file-name">monitor_check_wip.py</td>
        <td>fichier</td>
        <td>1.9 KB</td>
        <td>25/01/2025</td>
    </tr>
    <tr>
        <td class="file-name">patients_HorizonSante.xlsx</td>
        <td>fichier</td>
        <td>67 MB</td>
        <td>12/07/2025</td>
    </tr>
</tbody>
</table>
</div>
</div>
`;

return res.status(200).json({
    success: true,
    message: "ZIP décompressé",
    decompressedFiles: decompressedFilesHtml
});
} else {
    console.log('invalid flag for challenge 4');
    return res.sendStatus(401);
}
});
});
});
});
```

Ici, la route `/challenge4/validate` attend un flag dans le corps de la requête. Si le flag est correct, elle renvoie un message de succès et un extrait HTML simulant l'affichage des fichiers décompressés. Sinon, elle renvoie une erreur 401.

7.3.5 Challenge 5

Le challenge 5 est entièrement géré côté frontend. Il n'a pas besoin du backend, car l'analyse repose sur l'IDE Python intégré (Pyodide) et les scripts fournis directement dans l'interface.

7.3.6 Challenge 6

Le challenge 6 consiste en la mise en place d'un bot automatisé qui a pour objectif d'interagir avec le backend en injectant un cookie administrateur spécifique. Le but principal est de simuler un scénario de type « bot headless administrateur » qui visite des pages et déclenche des actions sensibles grâce

à un cookie privilégié. Le joueur·euse n'a pas directement accès à ce cookie, mais il doit trouver un moyen d'exploiter le bot pour qu'il l'envoie au backend et ainsi récupérer le flag.

La configuration Docker du bot est relativement simple, le service `admin-bot` est construit à partir du dossier `docker-bot` et expose son API sur le port 3001. Des variables d'environnement permettent de définir l'URL cible `CHALLENGE_URL`, l'environnement d'exécution `NODE_ENV`, et le port de l'API. Les routes Traefik sont configurées afin de rediriger les requêtes vers ce service.

```
admin-bot:
  build: ./docker-bot
  container_name: cookie_admin_bot
  environment:
    - CHALLENGE_URL=http://frontend:80/challenges2025/6_cookie_admin/
    - NODE_ENV=${NODE_ENV:-development}
    - BOT_API_PORT=3001
  labels:
    # Expose le bot dans Traefik
    - "traefik.enable=true"
    # Route pour l'API du bot
    - "traefik.http.routers.admin-bot.rule=${HOST_RULE} && PathPrefix(`/api/bot`)"
      - "traefik.http.routers.admin-bot.middlewares=admin-bot-striprefix"
      - "traefik.http.middlewares.admin-bot-striprefix.striprefix.prefixes=/api/bot"
        - "traefik.http.routers.admin-bot.priority=120"
    # Enable TLS
    - "traefik.http.routers.admin-bot.tls=true"
    # Port du service
    - "traefik.http.services.admin-bot.loadbalancer.server.port=3001"
  depends_on:
    - frontend
    - traefik
```

De plus, un mécanisme de limitation de sessions et de durée de vie est mis en place pour éviter qu'un joueur ne monopolise le bot trop longtemps.

Le service `log-viewer` est basé sur l'image publique Dozzle et se connecte au socket Docker afin de filtrer uniquement les logs du conteneur du bot. Il est exposé en local sur le port 9999 et n'est activé que dans le profil `monitoring`.

```

log-viewer:
  image: amir20/dozzle:latest
  container_name: cookie_admin_logs
  volumes:
    - /var/run/docker.sock:/var/run/docker.sock:ro
  ports:
    - "9999:8080"
  environment:
    DOZZLE_FILTER: "name=cookie_admin_bot"
  profiles:
    - monitoring

```

Le bot, implémenté sur `bot.js` (Annexe-G) avec Puppeteer, a pour rôle d'ouvrir des pages, de gérer les cookies et de transmettre des requêtes HTTP vers le backend. Il expose une API permettant au joueur de créer une session, de positionner des cookies et de demander au bot d'exécuter une requête donnée. Ce mécanisme reproduit un cas concret d'attaque où un attaquant cherche à exploiter un bot ou un navigateur headless utilisé en interne pour exécuter des actions avec plus de priviléges qu'un simple utilisateur.

Côté backend, plusieurs routes sont définies et utilisées pour le déroulement du challenge. La route `POST /challenge6/validate` reçoit en entrée un corps JSON contenant un champ `adminCookie`. Elle compare cette valeur à celle attendue et renvoie une réponse si le cookie est valide ou pas. Cela permet aux joueur·euse·s de tester différentes hypothèses sans forcément passer par le bot, mais sans qu'aucune action sensible ne soit exécutée.

```

// Challenge 6 2025
app.post("/challenge6/validate", (req, res) => {
  const { adminCookie } = req.body;

  if (!adminCookie) {
    return res.status(400).json({ valid: false, error: "Cookie admin requis" });
  }

  // Vérifier si cookie est admin
  const validAdminValue = "ADM1N_535510N_T0KEN25";

  if (adminCookie === validAdminValue) {
    return res.status(200).json({
      valid: true,
      message: "Cookie admin valide"
    });
  }
});

```

IMPLÉMENTATION DES CHALLENGES

```
    });
} else {
    return res.status(200).json({
        valid: false,
        message: "Cookie admin invalide"
    });
}
});
```

La seconde route critique est `GET /challenge6/deleteFiles`. Elle simule la suppression de fichiers sensibles mais n'autorise l'accès que si le cookie admin envoyé correspond à la valeur attendue. Si la condition est remplie, le backend recherche le flag associé au challenge 6 dans la variable d'environnement `CHALL_FLAGS_2025`. Le flag est extrait, renvoyé au joueur et accompagné d'un message de confirmation.

```
// Challenge 6 2025 suppression fichiers
app.get("/challenge6/deleteFiles", (req, res) => {
    const adminCookie = req.cookies.admin;

    if (!adminCookie || adminCookie !== "ADM1N_535510N_T0KEN25") {
        return res
            .status(403)
            .json({ error: "Accès non autorisé" });
    }

    const flagsString = process.env.CHALL_FLAGS_2025;
    if (!flagsString) {
        return res.status(500).json({ error: "Flag manqué" });
    }

    const targetFlag = flagsString
        .split(";")
        .find((flag) => flag.startsWith("chall6="));

    if (!targetFlag) {
        return res.status(404).json({ error: "Flag non trouvé" });
    }

    return res.status(200).json({
        success: true,
        message: `Fichiers supprimés avec succès! Confirmation :
```

```
 ${targetFlag.split("=")[1]}`,  
     flag: targetFlag.split("=")[1]  
   );  
 });
```

Un point essentiel pour que le challenge reste jouable dans le cas où plusieurs joueurs interagissent avec le bot. En effet, si plusieurs joueurs interagissent avec le bot en même temps, il faut éviter qu'ils ne se perturbent mutuellement. Pour cela, chaque joueur est associé à un identifiant unique `playerId`, généré sous forme d'UUID. Cet identifiant est utilisé dans toutes les requêtes, aussi bien côté bot que côté backend.

7.3.7 Challenge 7

Enfin, le challenge 7 ne fait pas appel au backend. L'ensemble du challenge (analyse des logs et blocage de l'adresse IP) est simulé directement côté frontend pour simplifier l'implémentation et rester accessible sans nécessiter de configuration serveur complexe.

7.4 Intégration sur le site web

L'intégration des nouveaux challenges « Blackout » dans la plateforme existante s'est faite en trois parties :

1. Initialisation des flags et extension du modèle de données
2. Ajout d'un nouveau « mini-site » de jeu (fichiers blackoutgame.html et blackoutmain.js)
3. Raccordement à l'expérience globale (lien depuis `index.html`, pop-ups d'intro avec les indices et configuration `.env`).

Ces ajouts s'alignent sur l'architecture en place : un frontend statique routé par Traefik, un backend Express, et des données persistées (MongoDB et MySQL) déjà utilisées par les scénarios 2020/2021.

7.4.1 Initialisation des flags côté serveur

Pour éviter de placer les réponses dans le frontend, les flags 2025 sont déclarés dans les variables d'environnement et insérés au démarrage dans MongoDB au format SHA3-256, comme les scénarios 2020/2021. Le fragment suivant, ajouté à `db.js`, parcourt `CHALL_FLAGS_2025`, découpe chaque entrée `challX=VAL`, calcule le hash, puis crée le document Flag s'il n'existe pas

```
/* ... */
// Support for 2025 challenges
const flags_2025 = process.env.CHALL_FLAGS_2025.split(';');

for await (const flag of flags_2025) {
    const elem = flag.split('=');
    assert(elem.length === 2);
    const hash = new SHA3(256);
    hash.update(elem[1]);
    if (!(await Flag.exists({chall_name: "2025_"+elem[0]})))
        await Flag.create({chall_name: "2025_"+elem[0], value:
hash.digest('hex')});
}
```

7.4.2 Ajout du mini-site de jeu « Blackout »

Comme pour les anciens scénarios (chaque challenge = mini-site dans son dossier), Blackout introduit une page de jeu dédiée (`blackoutgame.html`) et un script de contrôle (`blackoutmain.js`). Cette approche permet d'orchestrer l'UI du scénario (iframe principale, champ de réponse, pop-ups d'aide/indices) sans impacter les autres jeux.

7.4.2.1 blackoutgame.html

Le fichier HTML charge le thème, les scripts communs, les pop-ups par challenge (0 à 8) et l'iframe qui héberge l'écran actif. On y retrouve également le champ de validation (réponse) et les includes HTML (header, popups) pour conserver la même UX que les autres scénarios.

```
<!doctype html>
<head>
    <title>Blackout de le Centre Hospitalier Horizon Santé</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <link href="https://fonts.googleapis.com/css?family=Open+Sans:300,400,700,800" rel="stylesheet">

    <link rel="stylesheet" href="css/animate.css">
    <link rel="stylesheet" href="css/owl.carousel.min.css">

    <link rel="stylesheet" href="fonts/ionicons/css/ionicons.min.css">
    <link rel="stylesheet" href="fonts/fontawesome/css/font-awesome.min.css">

    <!-- Theme Style -->
    <link rel="stylesheet" href="css/style.css">
    <style>
        canvas {
            margin: 0 auto;
        }
    </style>

    <link rel="apple-touch-icon" sizes="180x180" href="images/favicons/apple-icon-180x180.png">
    <link rel="icon" type="image/png" sizes="192x192" href="images/favicons/android-icon-192x192.png">
    <link rel="icon" type="image/png" sizes="32x32" href="images/favicons/favicon-32x32.png">
    <link rel="icon" type="image/png" sizes="16x16" href="images/favicons/favicon-16x16.png">
</head>
<div class="game blackoutgame" w3-include-html=".//header.html"></div>
<body>
    <input id="inputHint" type="text" id="flag" name="flag"
placeholder="réponse">
```

```
<button id="sumbitHint" class="submitHint">Valider l'étape !</button>
<div id="game"></div>
<iframe id="iframeChall" src="" frameborder="0" style="display: block; background: #000; border: none; overflow: hidden; height: 100vh; width: 100%">
</iframe>

<!-- Challenge popup includes for Blackout 2025 -->
<div w3-include-html=".//challenges2025/0_intro/popup.html"></div>
<div w3-include-html=".//challenges2025/1_mail_contagieux/popup.html"></div>
<div w3-include-html=".//challenges2025/2_portail_frauduleux/popup.html"></div>
<div w3-include-html=".//challenges2025/3_partage_oublie/popup.html"></div>
<div w3-include-html=".//challenges2025/4_cle_cachee/popup.html"></div>
<div w3-include-html=".//challenges2025/5_script_mystere/popup.html"></div>
<div w3-include-html=".//challenges2025/6_cookie_rancon/popup.html"></div>
<div w3-include-html=".//challenges2025/7_blocage_cible/popup.html"></div>
<div w3-include-html=".//challenges2025/8_outro/popup.html"></div>
<div w3-include-html=".//popupSubmitChall.html"></div>
</body>

<script src="js/jquery-3.2.1.min.js"></script>
<script src="js/popper.min.js"></script>
<script src="js/bootstrap.min.js"></script>
<script src="js/owl.carousel.min.js"></script>
<script src="js/jquery.waypoints.min.js"></script>
<script src="js/template.js"></script>
<script src="js/includehtml.js"></script>
<script>
    includeHTML();
</script>
<script src="js/phaser.min.js"></script>
<script src="js/blackoutmain.js"></script>
<script src="node_modules/@azerion/phaser-input/build/phaser-input.js"></script>

<script src="https://www.google.com/recaptcha/api.js"></script>
```

7.4.2.2 `blackoutmain.js`

Le fichier `blackoutmain.js` (annexe-F) constitue le cœur du moteur du scénario Blackout 2025. Développé avec le framework Phaser, il orchestre l'affichage du niveau, le déplacement du personnage, l'interaction avec les plateformes représentant les différents challenges, ainsi que la communication avec le backend pour la validation des étapes.

Dès l'initialisation, le script charge les éléments visuels nécessaires : le fond, les textures des plateformes, le héros, ainsi que les données décrivant la disposition des plateformes dans le fichier `level01Blackout.json`. Le héros est représenté comme un sprite animé pouvant se déplacer horizontalement et s'orienter automatiquement vers la gauche ou la droite selon la direction.

Chaque plateforme est associée à un challenge et rendue cliquable. Lorsqu'une plateforme est sélectionnée, le personnage se déplace automatiquement jusqu'à elle. Si le challenge est accessible, une popup s'ouvre pour présenter les consignes et permettre de lancer l'interface spécifique. Cette interface est affichée dans une iframe intégrée à la page principale.

La progression est gérée grâce à un système de cookies : `bk2025_xH92f_curr` enregistre l'étape en cours et `bk2025_mP81x_all` mémorise l'ensemble des challenges débloqués.

Lorsqu'un joueur·euse saisit un flag dans le champ de réponse et le valide, le script envoie une requête POST au backend sur la route `/backend/2025/flag`. Le serveur vérifie la validité du flag et renvoie un code HTTP : `200` pour une réussite, ce qui débloque la plateforme suivante et affiche un message de félicitations, `401` si le flag est incorrect, avec un message d'encouragement, `404` en cas d'erreur de challenge.

Cette logique garantit que la progression se fait de manière linéaire et que les étapes ne sont pas contournables. Les plateformes déjà résolues changent d'apparence (zone verte) pour offrir un retour visuel immédiat.

Le moteur intègre également des cas spécifiques, comme pour le Challenge 3, où un paramètre `?dir=` permet de simuler la navigation dans des dossiers via un mapping prédéfini entre les chemins et des pages HTML distinctes.

Enfin, `blackoutmain.js` prend en charge la compatibilité et l'expérience utilisateur : il vérifie le type de navigateur et alerte si le jeu est lancé sur un appareil mobile ou un navigateur non supporté, afin de garantir la meilleure expérience possible.

7.4.3 Raccordement dans la page d'accueil, routage et configuration des flags `.env / .env.prod`

Pour exposer le nouveau scénario dans l'UI globale, `index.html` reçoit une déclaration de l'année dans la constante `VALID_YEARS`, pour que la logique cliente supporte 2025 (au même titre que 2020/2021) et un bloc de présentation (texte + vidéo) et un bouton d'accès à `blackoutgame.html`.

Cette intégration conserve le parcours utilisateur habituel : découverte, teaser, puis accès aux défis.

```
<!-- ... -->  
  
const VALID_YEARS = ["2020", "2021", "2025"];
```

```
<!-- ... -->
<a href=".blackoutgame.html" class="btn btn-white btn-outline-white px-3 py-3 long-txt-button col-2 ml-xl-5"> Blackout<span class="ion-arrow-right-c"></span></a>
<!-- ... -->
<!-- ... -->
<section class="section bg-light element-animate" id="game3">
    <div class="container introhacking">
        <div class="row justify-content-center align-items-center mb-5">
            <h2 class="">Blackout de le Centre Hospitalier Horizon Santé</h2>
        </div>
        <div class="row align-items-center mb-5">
            <div class="col-md-7 pr-md-5 mb-5">
                <div class="block-41">
                    <div class="block-41-text">
                        <p>
                            Le Centre Hospitalier Horizon Santé a été victime
                            d'une cyberattaque majeure, mettant en péril la sécurité des données de ses
                            patients.
                        </p>
                        <p>
                            En tant qu'expert en cybersécurité, ta mission de
                            réussir à infiltrer le domaine des attaquants, supprimer les données volées et
                            bloquer les attaquants.
                        </p>
                    </div>
                </div>
            </div>
            <div class="col-md-4 ">
                <!-- TODO CHANGE VIDEO LINK-->
                <div class="embed-responsive embed-responsive-16by9">
                    <iframe src="https://www.youtube.com/embed/jgkrl94bnvw"
allowfullscreen></iframe>
                </div>
                <br>
                <a href=".blackoutgame.html" class="btn btn-white btn-outline-white px-3 py-3 long-txt-button">Accéder aux
                    défis ! <span class="ion-arrow-right-c"></span></a>
                </div>
            </div>
        </div>
    </div>
```

```
</section>
<!-- -->
```

Cette page – comme tout le frontend – est servie via Traefik (terminaison TLS, StripPrefix pour /backend et /ssh), ce qui permet au client d'appeler /backend/... et d'intégrer des iframes /ssh?... sans connaître la topologie interne. C'est ce même schéma qui rend « Blackout » plug-and-play au sein du site.

Enfin, les flags sont définis côté serveur, dans `.env` et `.env.prod`. Lors du boot, `db.js` se charge de les hacher et de les insérer si besoin. Le format clé-valeur séparé par ; reste identique.

```
#...
CHALL_FLAGS_2025="chall1=horizonsante-support.com;
chall2=co_s3ss10n4cc3s5;chall3=patient_audit_07-12.zip;
chall4=horizon42;chall5=/admin/monitoring/bot_communication_panel_v2;
chall6=all_files_deleted;chall7=blk_185-225-123-77_ok"
```

Les bénéfices d'avoir des flags côté serveur sont qu'aucun secret/flag n'apparaît dans le code client. De plus, cela permet une gestion centralisée et versionnée par année, ainsi qu'une facilité d'opération (rotation, ajout/suppression sans rebuild du frontend).

7.5 Améliorations de l'implémentation et future correction

Plus de Challenges Amélioration du bot -> problème avec les cookies -> récupère les cookies de la session du joueur et non celui du bot

8 Tests

8.1 Tests unitaires

L'utilisation de Jest a permis de réaliser des tests unitaires sur les différentes fonctions critiques du scénario Blackout. Cela inclut la validation des mécanismes d'authentification, la gestion des sessions, et la vérification des permissions d'accès aux différentes sections du site. Les tests unitaires permettent de s'assurer que chaque composant fonctionne correctement de manière isolée avant de les intégrer dans le système global.

8.1.1 Configuration de l'environnement de test

Jest est configuré pour exécuter les tests dans un environnement Node.js. La configuration permet de mesurer la couverture du code testé et génère des rapports pour vérifier la qualité des tests.

8.1.2 Détails des tests par challenge

Les tests du challenge 1 valident la fonction `toggleDetails()` qui permet de basculer entre la vue normale et la vue détaillée d'un email. Un environnement DOM virtuel a été créé avec JSDOM pour simuler le navigateur. Ils vont vérifier l'état du DOM, valider l'état initial, tester le basculement de l'état mais aussi les basculements successifs. Ces tests permettent de garantir que l'interface utilisateur répond correctement aux interactions et que les états d'affichage sont gérés de manière cohérente.

Concernant le challenge 2, des tests unitaires ont été conçus pour valider le fonctionnement du filtre WAF (Web Application Firewall) implémenté. Le filtre a pour objectif de bloquer les tentatives d'injection SQL en identifiant des patterns suspects comme `SELECT`, `UNION`, `OR` et les commentaires `--`. Les tests couvrent des inputs normaux des cas où des chaînes malicieuses doivent être bloquées, l'insensibilité à la casse, et d'autres variations. De plus, des tests de connexion à la base de données sont également inclus pour s'assurer que les requêtes légitimes passent correctement. Ces tests permettent de confirmer que la connexion MySQL fonctionne à l'aide de différents scénarios, comme une connexion réussie, identifiants incorrects, ... Ils simulent également un processus d'authentification afin de vérifier que les interactions avec la base respectent la logique prévue. Ces tests permettent de

s'assurer que le WAF fonctionne comme attendu tout en laissant la possibilité aux joueurs de résoudre le challenge.

Pour le challenge 3, les tests valident la fonction `resolveDirToFile()` qui mappe les chemins de répertoires vers les fichiers HTML correspondants. Ils ont pour but de valider le chemin racine vers le fichier par défaut, de tester des répertoires principaux et secondaires (archives par année), de vérifier les fallback en cas de répertoires inexistantes, et de s'assurer de la cohérence du traitement du slash. Ces tests assurent que le système de navigation fonctionne correctement et que les utilisateurs peuvent accéder aux bonnes ressources.

Le challenge 4 intègre des tests concernant la validation du flag et à l'affichage correct des fichiers décompressés après avoir fourni la réponse valide. Les tests vérifient que l'absence de flag, la validité d'un flag, correcte ou non, pour renvoyer les bons messages. Ce type de test est fondamental pour garantir que le flux de validation reste cohérent et que l'utilisateur reçoit un retour clair et précis.

Dans le challenge 5, les tests testent les mécanismes de décodage en base64. La fonction testée doit vérifier et rejeter les séquences invalides, mais aussi reconstruire correctement une chaîne de plusieurs fragments encodés. Les tests vérifient que le décodage produit les chemins attendus et que la validation du flag fonctionne uniquement si l'utilisateur fournit le chemin correct. Cela assure que le processus de reverse engineering intégré au challenge est le plus possible fidèle à la réalité.

Pour le challenge 6, deux séries de tests unitaires ont été développées. La première concerne l'authentification du cookie administrateur, qui doit correspondre à une valeur bien précise. Les tests simulent l'absence du cookie, la présence d'un cookie invalide et le cookie correct, afin de s'assurer que le comportement voulu est obtenu. La deuxième série vérifie la suppression de fichiers, où l'action ne peut être faite que si le cookie est valide et que le flag est correctement transmis. Ces tests garantissent que l'escalade de priviléges via XSS est bien représentée et que la mécanique backend réagit correctement.

Enfin, les tests du challenge 7 vérifient le mécanisme de génération de code de blocage d'adresses IP. Les tests vérifient que le format des adresses est correct et que le code de confirmation produit est bien ce qui est attendu. La validation du flag est également testée pour s'assurer qu'un code généré correspond bien à l'IP suspecte identifiée. Cela permet de s'assurer que le challenge reste cohérent et que les joueurs peuvent interagir avec le système de manière réaliste.

Pour terminer, des tests de la base de données MongoDB valident l'initialisation sécurisée des flags de challenges. Ils vérifient que la création des flags est correcte à partir des variables d'environnement pour les différentes années, la validation du hachage des valeurs des flags, la connexion à la base de données, test la gestion des entrées et en cas de duplications de flags.

8.2 Tests utilisateurs

La phase des tests utilisateur a plusieurs objectifs. Elle vise tout d'abord à faire des observations et faire remonter de la connaissance sur la façon dont les participant·e·s interagissent avec la plateforme et les différents outils prévus. Pour garantir des conditions similaires à l'ensemble des joueur·euse·s, un minimum d'information a été transmis oralement. En ce qui concerne les instructions de base nécessaires à la compréhension du contexte du scénario, seules les instructions présentes sur la plateforme ont été transmises. Les données collectées lors de ces tests permettent d'identifier les points forts et les points faibles de l'histoire mais aussi de la conception des challenges, ainsi que des pistes d'amélioration. Elles aident également à évaluer l'accessibilité et la convivialité de la plateforme, en s'assurant que les défis sont compréhensibles et réalisables par un large public.

8.2.1 Protocole de test

8.2.1.1 Environnement

Dans le but de limiter au maximum les perturbations et d'éventuelles interactions externes, les tests ont été réalisés chez chacun des participant·e·s. Chaque session de test a duré environ 2h–3h, incluant une introduction, la phase de jeu, et une session de feedback à la fin.

L'outil ngrok a été utilisé pour permettre aux testeurs d'accéder à distance à la plateforme en créant un tunnel sécurisé vers le serveur local. Cela a facilité la mise en place de tests réalistes sans contrainte de réseau ou d'installation.

8.2.1.2 Participants

Un premier alpha test a été réalisé avec un·e participant·e avancé·e, ayant des connaissances approfondies en informatique de manière générale. Il a testé les challenges au fur et à mesure de leur mise en production. L'objectif était de valider la compréhension des défis que ce soit pour la consigne et les indices, et l'intérêt du scénario narratif. Ensuite, quatre autres participant·e·s sont intervenu·e·s, deux avec des connaissances avancées en informatique (dont un·e ayant suivi une formation en cybersécurité) et deux totalement débutant·e·s, c'est-à-dire sans aucune expérience de programmation. Ceux-ci ont réalisé l'ensemble du scénario en une seule session continue. Cette diversité de profils a permis de recueillir des retours variés et d'identifier des points d'amélioration pour différents niveaux de compétence.

8.2.1.3 Procédure de test

Une procédure de test similaire a été suivie pour chaque participant·e et se présente comme suit :

- Connexion à la plateforme et explication de sa structure.
- Présentation du contexte et des règles du jeu.
- Phase de jeu où le participant·e tente de résoudre les challenges.

- Observation et prise de notes sur le comportement, les difficultés rencontrées et les stratégies utilisées.
- En cas de blocage, des indices supplémentaires sont fournis pour aider le participant·e à progresser et dans les cas extrêmes, la solution est donnée pour permettre de continuer le test.
- Session de feedback où le participant·e partage ses impressions, les aspects appréciés et les suggestions d'amélioration.

8.3 Résultats et bilan des tests

Le test progressif avec l'alpha testeur a permis de valider assez rapidement la solidité technique de chaque challenge et de corriger des incohérences de l'histoire ou des bugs de développement. Les premiers retours ont permis d'ajuster les indices, de clarifier certaines étapes et de rendre le jeu plus accessible aux débutant·e·s.

Par la suite, les tests finaux ont révélé des différences assez importantes dans l'expérience utilisateur selon le niveau de compétence des participant·e·s. Les personnes avec des connaissances avancées en informatique ont généralement progressé de manière fluide à travers les premiers challenges, sans vraiment utilisé les indices. Ils ont confirmé l'intérêt du scénario et la pertinence technique des défis. Tandis que les participant·e·s débutant·e·s ont rencontré beaucoup plus de blocages, notamment dans la compréhension du vocabulaire technique, des concepts de base et du coup le format de réponse attendu mais aussi sur le manque de clarté sur les indices et ce qui était.

De manière générale, les tests montrent que le scénario est accessible à des débutants motivés tout en étant amusant pour des profils plus avancés. L'équilibre entre narration et des techniques à utilisées ont été jugé satisfaisant.

8.3.1 Performance par niveau de compétence

Pour mieux analyser les résultats, les participant·e·s ont été regroupé·e·s en deux catégories selon leur niveau de compétence en informatique : avancé·e·s et débutant·e·s.

Les joueur·euse·s avancé·e·s avaient une bonne maîtrise des concepts de base en programmation et en sécurité informatique. Ils ont été très rapidement autonomes dans la résolution des challenges. Des principales observations ont pu être faites : le temps de résolution était généralement inférieur aux estimations initiales, l'utilisation des indices était limitée, car ils préféraient explorer par eux-mêmes et trouver la solution et enfin une grande appréciation au niveau de l'aspect narratif qui a permis de développer les compétences techniques tout en restant motivé.

En ce qui concerne les joueur·euse·s débutant·e·s, ils n'avaient aucune connaissances en programmation et très peu, de manière générale, en informatique. Leur expérience a été plus variée et a mis en lumière plusieurs points d'amélioration possibles. Ces testeurs·euses ont montré un parcours plus contrasté : ils avaient besoin de plus de temps pour comprendre les consignes et les concepts de base et donc d'aller régulièrement dans la boîte à outils, ils ont eu recours aux indices et les ont utilisé, à chaque fois tous, un besoin beaucoup plus grand de temps pour réaliser les challenges plus techniques, en particulier les challenges 2, 4,5 et 6. Malgré les difficultés rencontrées, ils ont exprimé une grande satisfaction lorsqu'ils réussissaient à résoudre un challenge, ce qui leur a permis de rester motivé et de progresser.

En ce qui concerne les joueur·euse·s débutant·e·s, sans connaissances en programmation et avec seulement des bases limitées en informatique, leur expérience a été plus variée et a mis en lumière plusieurs points d'amélioration possibles. Ces testeurs·euses ont montré un parcours plus contrasté : ils ont eu besoin de davantage de temps pour comprendre les consignes et les concepts fondamentaux, ce qui fait qu'ils ont consulté régulièrement la boîte à outils. Ils ont systématiquement utilisé tous les indices disponibles et ont particulièrement eu des difficultés lors des challenges les plus techniques, notamment les challenges 2, 4, 5 et 6. Malgré les difficultés rencontrées, ils ont exprimé une grande satisfaction lorsqu'ils réussissaient à résoudre un challenge, ce qui leur a permis de rester motivé et de progresser, même après plusieurs essais.

8.3.2 Points forts identifiés

Plusieurs aspects du projet ont été particulièrement appréciés par les participant·e·s, quel que soit leur niveau de compétence.

Dans un premier temps, le scénario et la narration. L'immersion narrative a été appréciée par tout le monde et a permis de contextualiser les défis techniques. La progression de l'histoire maintient la motivation des joueur·euse·s et les éléments de storytelling facilitent la mémorisation des concepts abordés.

Ensuite, la diversité des challenges. La variété des types de défis (exploitation de vulnérabilités web, analyse de fichiers, reverse engineering, etc.) a été appréciée pour maintenir l'intérêt et découvrir un large champ de compétences en cybersécurité, que ce soit au niveau de l'attaque mais aussi à l'aide d'un challenge défensif.

Enfin, la plateforme technique. L'utilisation de ngrok a permis des tests à distance sans friction technique majeure et sans avoir à forcer les participant·e·s à se déplacer. Aucun gros problème de stabilité n'a été rencontré durant les sessions. L'interface utilisateur a été jugée claire et intuitive, facilitant la navigation entre les différents challenges. Enfin, un dernier élément relevé par tous les participant·e·s est la simplicité d'utilisation grâce à l'accès direct grâce à un navigateur, sans installation supplémentaire de logiciel. Cela a contribué à rendre l'expérience plus accessible et à réduire les obstacles techniques.

8.3.3 Points faibles et pistes d'amélioration

Concernant les points faibles, plusieurs aspects ont été identifiés et pourraient être améliorés par la suite.

Le premier point concerne les indices et consignes. Certains indices ont été jugés pas assez compréhensibles, ce qui rendait la compréhension difficile pour certain·e·s ou au contraire trop explicites et du coup trop facile. Le vocabulaire technique dans certaines consignes a aussi été un obstacle pour les débutant·e·s. Les joueur·euse·s devaient régulièrement trouver les informations dans la boîte à outils,

ce qui a parfois interrompu le flux de jeu. Un dernier éléments souligné par les participant·e·s a été aussi la boîte à outils, qui était jugée très dense et complexe, rendant la recherche d'informations plus difficile.

Ensuite, la difficulté de certains challenges. Quelques défis ont été trop complexes pour les débutant·e·s, en particulier ceux qui demandent des connaissances spécifiques en programmation ou en sécurité informatique. Des blocages assez longs ont été remarqués sur ces challenges spécifiques, et mon intervention a été nécessaire. Un meilleur équilibrage de la difficulté pourrait être envisagé pour rendre l'expérience plus fluide.

Des améliorations ont été apportées lors de la réalisation des tests, en reformulant certaines consignes, en ajustant la difficulté de certains challenges et en complétant les indices pour les rendre plus utiles et pertinents. Cependant, il reste encore des pistes d'amélioration à explorer.

Plusieurs pistes d'amélioration ont été identifiées pour les améliorations du jeu mais aussi de la plateforme. L'ajout de fonctionnalités de suivi des progrès des joueurs pourrait leur permettre de voir une meilleure progression sur les jeux. La boîte à outils pourrait également être simplifiée et mieux intégrée pour faciliter la recherche d'informations, par exemple en mettant directement l'accès aux informations précises dans la consigne ou dans le défi. Lors du tests, tous les indices étaient visibles, il faudrait les cacher pour qu'ils soient visibles au fur et à mesure des besoins. Enfin, la mise en place d'un système de feedback continu permettrait de recueillir des retours en temps réel et d'ajuster le scénario de manière plus dynamique.

9 Conclusion

Glossaire

Bibliographie

2024 ISC2 Cybersecurity Workforce Study, [sans date].. En ligne. Disponible à l'adresse: <https://www.isc2.org/Insights/2024/10/ISC2-2024-Cybersecurity-Workforce-Study> [Consulté le 9 juillet 2025].

ABT, Clark C., 1970. *Serious Games*. En ligne. New York, Viking Press. ISBN 978-0-670-63490-3. Disponible à l'adresse: <http://archive.org/details/seriousgames0000abtc> [Consulté le 22 juillet 2025].

CTF Hacking : guide ultime pour devenir un expert en Capture The Flag, [sans date].. En ligne. Disponible à l'adresse: <https://www.oteria.fr/blog-oteria/ctf-hacking-guide-complet-des-competitions-de-cybersecurite> [Consulté le 22 juillet 2025].

Cyber Wargame : Des Serious Games sur la Cybersécurité, [sans date].. En ligne. Disponible à l'adresse: <https://www.cyber-wargame.fr/> [Consulté le 22 juillet 2025].

EUROPEAN COMMISSION. DIRECTORATE GENERAL FOR COMMUNICATIONS NETWORKS, CONTENT AND TECHNOLOGY., 2024. *The Digital Decade*. En ligne. LU: Publications Office. Disponible à l'adresse: <https://data.europa.eu/doi/10.2759/646681> [Consulté le 22 juillet 2025].

EUROPEAN UNION AGENCY FOR CYBERSECURITY (ENISA), 2024. *2024 Report on the State of Cybersecurity in the Union – Condensed Version*. En ligne. Athens, Greece. Disponible à l'adresse: <https://www.enisa.europa.eu/publications/2024-report-on-the-state-of-cybersecurity-in-the-union> [Consulté le 22 juillet 2025].

FORTINET, 2024a. *2024 Cybersecurity Skills Gap*. En ligne. Sunnyvale, CA, USA. Disponible à l'adresse: <https://www.fortinet.com/content/dam/fortinet/assets/reports/2024-cybersecurity-skills-gap-report.pdf> [Consulté le 14 juillet 2025].

FORTINET, 2024b. *Fortinet 2024 Cybersecurity Skills Gap Global Research Report*. En ligne. Sunnyvale, CA. Disponible à l'adresse: <https://www.fortinet.com/content/dam/fortinet/assets/reports/2024-cybersecurity-skills-gap-report.pdf> [Consulté le 9 juillet 2025].

Hack The Box: The #1 Cybersecurity Performance Center, [sans date].. En ligne. Disponible à l'adresse: <https://www.hackthebox.com/> [Consulté le 10 juillet 2025].

BIBLIOGRAPHIE

HILL, Winston, FANUEL, Mesafint et YUAN, Xiaohong, 2020. Comparing Serious Games for Cyber Security Education. 2020.

Informations Sur Les Outils et Méthodes Utilisées !, [sans date]. . En ligne. Disponible à l'adresse: <https://shana.heig-vd.ch/tools.html> [Consulté le 8 juillet 2025].

Initiation Au Ethical Hacking, [sans date]. . En ligne. Disponible à l'adresse: <https://shana.heig-vd.ch/> [Consulté le 8 juillet 2025].

KNOWBE4, 2025. *Phishing Threat Trends Repor*. En ligne. Clearwater, FL, USA. Disponible à l'adresse: https://www.knowbe4.com/hubfs/Phishing-Threat-Trends-2025_Report.pdf [Consulté le 22 juillet 2025].

NG, Chiu Yeong et HASAN, Mohammad Khatim Bin, 2025. Cybersecurity Serious Games Development: A Systematic Review. *Computers & Security*. En ligne. 1 mars 2025. Vol. 150, p. 104307. DOI [10.1016/j.cose.2024.104307](https://doi.org/10.1016/j.cose.2024.104307). [Consulté le 14 juillet 2025].

Qu'est-ce que le cyberespionnage ?, [sans date]. . En ligne. Disponible à l'adresse: <https://www.fortinet.com/fr/resources/cyberglossary/cyber-espionage.html> [Consulté le 16 juillet 2025].

Qu'est-ce qu'un cyber range ? | IBM, [sans date]. . En ligne. Disponible à l'adresse: <https://www.ibm.com/fr-fr/think/topics/cyber-range> [Consulté le 20 mars 2025].

Root Me : Plateforme d'apprentissage Dédiée Au Hacking et à La Sécurité de l'Information, [sans date]. . En ligne. Disponible à l'adresse: <https://www.root-me.org/> [Consulté le 10 juillet 2025].

SANTINI, Ginevra, [sans date]. A European Outlook from the ISACA 2024 State of Cybersecurity Report | Digital Skills and Jobs Platform. . En ligne. Disponible à l'adresse: <https://digital-skills-jobs.europa.eu/en/latest/news/european-outlook-isaca-2024-state-cybersecurity-report> [Consulté le 9 juillet 2025].

Sauve La Terre de l'arme Galactique !, [sans date]. . En ligne. Disponible à l'adresse: <https://shana.heig-vd.ch/galacgame.html> [Consulté le 8 juillet 2025].

Sensibilisation à la cybersécurité et gestion du risque humain, [sans date]. . En ligne. Disponible à l'adresse: <https://sosafe-awareness.com/fr/> [Consulté le 22 juillet 2025].

Serious game sécurité informatique: le jeu Urban Gaming, [sans date]. . En ligne. Disponible à l'adresse: <https://www.urbangaming.fr/jeu-change-and-serious/securite-informatique/> [Consulté le 21 juillet 2025].

Shana a Disparu. Retrouve-la !, [sans date]. . En ligne. Disponible à l'adresse: <https://shana.heig-vd.ch/shanagame.html>? [Consulté le 8 juillet 2025].

Shirudo | Serious Game Multilingue En Cybersécurité, [sans date]. . En ligne. Disponible à l'adresse: <https://shirudo.eu/> [Consulté le 22 juillet 2025].

SPYS, Denys et SOLOVEI, Anna, 2025. Phishing Statistics in 2025: The Ultimate Insight | TechMagic. . En ligne. 18 juin 2025. Disponible à l'adresse: <https://www.techmagic.co/blog/blog-phishing-attack-statistics/> [Consulté le 22 juillet 2025].

TryHackMe | Simple CTF, [sans date]. . En ligne. Disponible à l'adresse: <https://tryhackme.com/room/easyctf> [Consulté le 10 juillet 2025].

WAHL, Thomas, 2020. Eurobarometer: Europeans Attitudes towards Cyber Security. . En ligne. 28 avril 2020. Disponible à l'adresse: <https://eucrim.eu/news/eurobarometer-europeans-attitudes-towards-cyber-security/> [Consulté le 22 juillet 2025].

What Is Cyber Range · Definition · DIATEAM, [sans date]. . En ligne. Disponible à l'adresse: <https://www.diateam.net/what-is-a-cyber-range/> [Consulté le 22 juillet 2025].

When Ransomware Kills: Attacks on Healthcare Facilities | IBM, [sans date]. . En ligne. Disponible à l'adresse: <https://www.ibm.com/think/insights/when-ransomware-kills-attacks-on-healthcare-facilities> [Consulté le 14 juillet 2025].

Y-Security - HEIG-VD, [sans date]. . En ligne. Disponible à l'adresse: <https://heig-vd.ch/recherche/groupes-poles/y-security> [Consulté le 9 juillet 2025].

ZYDA, M., 2005. From Visual Simulation to Virtual Reality to Games. *Computer*. En ligne. septembre 2005. Vol. 38, no. 9, p. 25-32. DOI [10.1109/MC.2005.297](https://doi.org/10.1109/MC.2005.297). [Consulté le 22 juillet 2025].

Figures

Fig. 1	« Shana a disparu » - Interface du jeu (<i>Shana a Disparu. Retrouve-la !</i> [sans date])	17
Fig. 2	Boite à outils (<i>Informations Sur Les Outils et Méthodes Utilisées !</i> [sans date])	18
Fig. 3	« Sauve la Terre de l'arme galactique » - Interface du jeu (<i>Sauve La Terre de l'arme Galactique !</i> [sans date])	19
Fig. 4	Schéma d'un cyber-range (<i>What Is Cyber Range · Definition · DIATEAM</i> [sans date])	25
Fig. 5	Page des challenges de RootMePlateforme (<i>Root Me : Plateforme d'apprentissage Dédiée Au Hacking et à La Sécurité de l'Information</i> [sans date])	26
Fig. 6	IDE présent sur le jeu « Sauve la Terre de l'arme galactique », dans le challenge 6	32
Fig. 7	Terminal présent sur les 2 jeux, dans les challenges 9 de « Shana a disparu » et 5, 8 dans « Sauve la Terre de l'arme galactique »	33
Fig. 8	Interface du jeu après la validation d'un challenge	35
Fig. 9	Interface du jeu qui ne change pas après la validation d'un challenge et pas de progression dans l'histoire.	36
Fig. 10	Visuel du mail avec en dessous le terminal, challenge 1	81
Fig. 11	Visuel des détails du mail, challenge 1	82
Fig. 12	Page de connexion au portail frauduleux avec un message d'alerte du WAF, challenge 2 ..	83
Fig. 13	Session connexion réussite, challenge 2	83
Fig. 14	Dashboard une fois connecté sur la plateforme des attaquants, challenge 3	84
Fig. 15	Dossiers shared, challenge 3	84
Fig. 16	Dossiers racine, challenge 3	85
Fig. 17	Exploration des dossiers jusqu'au dossier <code>/archives/2025</code> , challenge 3	85
Fig. 18	IDE Python pour analyser le fichier et terminal afin de pouvoir réaliser un <code>zipinfo</code> , challenge 4	86
Fig. 19	Terminaux disponibles pour le challenge, challenge 4	86
Fig. 20	IDE Python pour analyser le fichier et réaliser du code pour identifier la page, challenge 5 ..	87
Fig. 21	Interface du chatbot, challenge 6	88
Fig. 22	Interface de la plateforme de l'hôpital, challenge 7	88
Fig. 23	Visuel des logs, challenge 7	89
Fig. 24	Formulaire pour bloquer une IP et obtenir le code de validation, challenge 7	90

Outils utilisés

Journal de travail

Date	Description	Rech. [h]	Dev. [h]	Rapport [h]	Admin [h]
07.07.2025	Recherches sur la sensibilisation Rédaction du cahier des charges et de quelques idées	2	0	6	0
08.07.2025	Analyse de la plateforme et des techniques des challenges	0	16	0	0
09.07.2025					
10.07.2025	Recherches serious game, CTF,	6	0	10	0
11.07.2025	...				
	Rédaction des scénarios et de l'état de l'art				
14.07.2025	Rédaction plus approfondies des scénarios	10	0	30	0
18.07.2025	Recherches				
21.07.2025	Rédaction détaillées de l'introduction et de l'état de l'art	8	0	16	0
24.07.2025	Recherches				
	Rédaction et modification du scénario définitif				
25.07.2025	Présentation des challenges	0	0	0	8
	Modification du scénario				
28.07.2025	Reprise et réécriture des challenges	6	0	18	0
30.07.2025	Recherches				
	Écriture plus complète du rapport				

Date	Description	Rech. [h]	Dev. [h]	Rapport [h]	Admin [h]
31.07.2025	Rendu intermédiaire Architecture de la plateforme	0	0	8	0
01.08.2025	Architecture de la plateforme Écriture des consignes et des indices reçus par les participants	2	0	6	0
11.08.2025	Implémentation de la structure du scénario	0	37	3	0
15.08.2025	Correction rapport				
18.08.2025	Implémentation challenges 0, 1,	0	35	5	0
22.08.2025	2, 3 et 4 Tests utilisateurs rapides Changement organisation des challenges				
25.08.2025	Discussion avec l'enseignant responsable	0			0
26.08.2025	Implémentation des challenges	0	30	10	0
30.08.2025	5, 7 Rédaction et corrections rapport				
01.09.2025	Amélioration backend	0	35	5	0
05.09.2025	Implémentation challenge 6 Tools Tests utilisateurs rapides Rédaction et corrections rapport				
08.09.2025	Tools	0	38	2	0
12.09.2025	Implémentation challenge 6 Tests utilisateurs complets Tests unitaires				
15.09.2025		0	0	0	0
17.09.2025					
22.09.2025		0	0	0	0
24.09.2025					
29.09.2025		0	0	0	0
01.10.2025					

JOURNAL DE TRAVAIL

Date	Description	Rech. [h]	Dev. [h]	Rapport [h]	Admin [h]
06.10.2025		0	0	0	0
08.10.2025					

Annexes

-A Fichier JSON de configuration	126
-B API Express (<code>index.js</code>)	128
-C Modèles Mongoose (<code>db.js</code>)	139
-D Base MySQL (<code>init.sql</code>)	141
-E Docker Compose (<code>docker-compose.yml</code>)	143
-F Implémentation du jeu « Blackout » (<code>blackoutmain.js</code>)	146
-G Implémentation du bot pour le challenge 6 de 2025 (<code>bot.js</code>)	159
-H Présentation des challenges (ancienne version des défis)	172

Annexes

-A Fichier JSON de configuration

```
{  
    "platforms": [  
        {"image": "ground0", "x": 100, "y": 60, "idChall": "chall0", "urlChall": ""},  
        {"image": "ground1", "x": 200, "y": 60, "idChall": "chall1", "urlChall": "./challenges/01_windows_login/windows_login.html"},  
        {"image": "ground2", "x": 300, "y": 60, "idChall": "chall2", "urlChall": "./challenges/02_browser_history/browser_history.html"},  
        {"image": "ground3", "x": 400, "y": 60, "idChall": "chall3", "urlChall": "./challenges/03_same_color_text/index-01.html"},  
        {"image": "ground4", "x": 500, "y": 60, "idChall": "chall4", "urlChall": "./challenges/04_html_comment/comment.html"},  
        {"image": "ground5", "x": 600, "y": 60, "idChall": "chall5", "urlChall": "./challenges/05_admin_cookie/index.html"},  
        {"image": "ground6", "x": 700, "y": 60, "idChall": "chall6", "urlChall": "./challenges/06_caesar_cipher/cesar_data.html"},  
        {"image": "ground7", "x": 800, "y": 60, "idChall": "chall7", "urlChall": "./challenges/07_url_modification/gallery1.html"},  
        {"image": "ground8", "x": 900, "y": 60, "idChall": "chall8", "urlChall": "./challenges/08_SQL_injection/sql_injection.html"},  
        {"image": "ground9", "x": 1000, "y": 60, "idChall": "chall9", "urlChall": "./challenges/09_image_forensic/index.html"},  
        {"image": "ground10", "x": 1100, "y": 60, "idChall": "chall10", "urlChall": ""}  
    ],  
    "roads": [  
        {"image": "road", "x": 115, "y": 70}  
    ],  
    "invisible_grounds": [  
    ]  
}
```

```
    {"image": "inv1", "x": 1, "y": 70}  
],  
"hero": {"x": 100, "y": 50}  
}
```

-B API Express (`index.js`)

```
require('dotenv/config');
const cors = require('cors');
const express = require('express');
const cookieParser = require('cookie-parser');
const bodyParser = require('body-parser');
const {v4: uuidv4, validate: uuidValidate} = require('uuid');
const db = require('./db');
const {SHA3} = require('sha3');
const mailValidator = require("email-validator");
const mysql = require('mysql');
const seedrandom = require('seedrandom');
const jwt = require('jsonwebtoken');

const pool = mysql.createPool({
  connectionLimit: 10,
  host: "mysql",
  user: process.env.MYSQL_USER,
  password: process.env.MYSQL_PASS,
  charset: "utf8_general_ci",
  database: "dday"
});

const app = express();

// Configure middlewares
//app.use(cors({origin: "http://localhost:3000", credentials:true,
//allowedHeaders: "access-control-allow-origin,Origin,X-Requested-With,Content-Type,Accept"}));
app.use(cors({origin: "http://"+process.env.HOST_NAME, credentials:true,
allowedHeaders: "access-control-allow-origin,Origin,X-Requested-With,Content-Type,Accept"}));
app.use(cookieParser());
app.use(bodyParser.json());

//app.options('*', cors({origin:"http://localhost:3000", credentials:true,
//allowedHeaders: "access-control-allow-origin,Origin,X-Requested-With,Content-Type,Accept"}))
app.options('*', cors({origin:"http://"+process.env.HOST_NAME, credentials:true,
allowedHeaders: "access-control-allow-origin,Origin,X-Requested-With,Content-Type,Accept"}))
```

```
let urlencodedParser = bodyParser.urlencoded({extended: false})

function generateToken(TokenObject, secret, expiresIn) {
    return jwt.sign(TokenObject, secret, {expiresIn: expiresIn});
}

function checkToken(req, res, next) {
    const token = req.cookies.authtoken;
    jwt.verify(token, process.env.TOKEN_SECRET, (err, user) => {
        if (err || user === undefined) {
            console.log(`Session token is invalid or has expired for user`);
            return res.redirect("../login.html");
        }
        next();
    });
}

// Middleware to ensure a user cookie is set
app.use((req, res, next) => {
    // Check if the cookies contain a uuid, and copy it to the request if
    // present, otherwise generate a new one, and add it to the response
    if (req.cookies.uuid && uuidValidate(req.cookies.uuid)) {
        // Cookie valid
        req.uuid = req.cookies.uuid;
        console.log("cookie valid");
        res.cookie('uuid', req.uuid, { maxAge: 30*24*60*60*1000, httpOnly: true})
    } else {
        // Missing or invalid cookie
        console.log("cookie invalid or missing");
        req.uuid = uuidv4();
        res.cookie('uuid', req.uuid, { maxAge: 30*24*60*60*1000, httpOnly: true})
    }

    next()
})

const VALID_YEARS = ["2020", "2021"]

// Submit a flag
app.post('/:year/flag', (req, res) => {
    if (!req.body.chall || !req.body.flag || !req.params.year || !
```

```

VALID_YEARS.includes(req.params.year)) {
    return res.sendStatus(400);
} else {
    const year = req.params.year;
    db.models.flag.findOne({chall_name: year + "_" + req.body.chall}, (err,
flag) => {
        if (err || !flag)
            return res.sendStatus(404);

        const hash = new SHA3(256);
        hash.update(req.body.flag);
        // Check if flag matches
        if (hash.digest('hex') === flag.value) {
            console.log('valid flag');
            // Check if user exists
            db.models.user.findOne({uuid: req.uuid}, (err, person) => {
                if (err) return res.send(err);

                if (!person) {
                    console.log('new user');
                    db.models.user.create({uuid: req.uuid, flagged: [year +
"_" + req.body.chall]}).then(() => {
                        return res.sendStatus(200);
                    }).catch((err) => {
                        return res.send(err);
                    });
                } else {
                    console.log('existing user');
                    if (!person.flagged.includes(year + "_" +
req.body.chall)) {
                        console.log('not flagged');
                        person.flagged.push(year + "_" + req.body.chall);
                        person.save().then(() => {
                            return res.sendStatus(200);
                        }).catch((err) => {
                            return res.send(err);
                        });
                    } else {
                        console.log('already flagged');
                        return res.sendStatus(200);
                    }
                }
            })
        }
    })
}

```

```

        });
    } else {
        console.log('invalid flag');
        return res.sendStatus(401);
    }
});
});

// Check a flag
app.post('/:year/checkFlag', (req, res) => {
    if (!req.body.chall || !req.body.flag || !req.params.year || !
VALID_YEARS.includes(req.params.year)) {
        return res.sendStatus(400);
    } else {
        const year = req.params.year;
        db.models.flag.findOne({chall_name: year + "_" + req.body.chall}, (err,
flag) => {
            if (err || !flag)
                return res.sendStatus(404);

            const hash = new SHA3(256);
            hash.update(req.body.flag);
            // Check if flag matches
            if (hash.digest('hex') === flag.value) {
                return res.sendStatus(200);
            } else {
                return res.sendStatus(401);
            }
        });
    }
});

// DB chall endpoint (2020 and 2021 chall)
app.post('/db', (req, res) => {
    if (!req.body.user || !req.body.pass) {
        return res.sendStatus(400);
    } else {
        pool.query("SELECT * FROM users where ID = '" + req.body.user + "' and
pass = '" + req.body.pass + "';", function (err, results, fields) {
            if (err) {
                return res.send(err);
            }
        });
    }
});

```

ANNEXES

```
        } else {
            return res.send(results);
        }
    });
});
};

// socialNetwork chall endpoint (2021 chall)
app.post('/db/search', (req, res) => {
    res.setHeader("Content-Type", "application/json; charset=utf-8");
    if (!req.body.search) {
        return res.sendStatus(400);
    }
    else if (req.body.search === "default"){
        console.log("return default search user request");
        let value = req.body.search;
        pool.query("SELECT * FROM posts LIMIT 5;", function (err, results,
fields) {
            console.log(results);
            if (err) {
                return res.send(err);
            } else {
                return res.send(results);
            }
        });
    }
    else {
        console.log("return specific search user request");
        let value = req.body.search;
        let name = '';
        if(value.includes(' ')){
            name = value.split(' ')[0].toLowerCase();
        }
        else{
            name = value.toLowerCase();
        }
        pool.query("SELECT * FROM posts where nameLastname LIKE '%" + name +
"%';", function (err, results, fields) {
            if (err) {
                return res.send(err);
            } else {
                return res.send(results);
            }
        });
    }
});
```

```

        }
    });
}

// Store username information
app.post('/user', (req, res) => {
    const secret_key = process.env.CAPTCHA_SECRET_KEY;
    const token = req.body.token;

    //axios({
    //    method: 'post',
    //    url: `https://www.google.com/recaptcha/api/siteverify?secret=${secret_key}&response=${token}`
    //})
    //.then(response => {
    //    if (!response.data.success ||
    //        !req.body.name ||
    //        !req.body.surname ||
    //        !req.body.mail ||
    //        !mailValidator.validate(req.body.mail)) {
    //        return res.sendStatus(400);
    //}

    // Check if user exists
    db.models.user.findOne({uuid: req.uuid}, (err, person) => {
        if (err) return res.send(err);

        if (!person){
            return res.sendStatus(401);
        }

        // Person exists, check if all flags have been solved
        db.models.flag.countDocuments({"chall_name" : {$regex : VALID_YEARS[VALID_YEARS.length-1]}}).then((count) => {
            // If the flagged amount is smaller than the amount of flags,
            // unauthorised
            let yearly_flagged = person.flagged.filter(x =>x.startsWith(VALID_YEARS[VALID_YEARS.length-1]));
            if (yearly_flagged.length < count) {
                return res.status(402).send(yearly_flagged.map(x =>x.substring(VALID_YEARS[VALID_YEARS.length-1].length + 1)));
            }
        })
    })
})
}

```

```
}

    // Update the values of the person
    person.name = req.body.name;
    person.surname = req.body.surname;
    person.mail = req.body.mail;
    // save the person
    person.save().then(() => {
        return res.sendStatus(200);
    }).catch((err) => {
        return res.send(err);
    });
});

//})
//.catch(error => {
//    return res.sendStatus(401);
//});
});

// Store username information
app.get('/stats', (req, res) => {
    // retrieve all users
    db.models.user.find({}, (err, persons) => {
        if (err) return res.send(err);

        if (!persons){
            return res.sendStatus(401)
        }
        let result = [];
        for(let i = 0; i < persons.length; i++){
            let yearly_flagged = persons[i].flagged.filter(x
=>x.startsWith(VALID_YEARS[VALID_YEARS.length-1])).length
            result.push(yearly_flagged);
        }
        res.send(result);

    });
});

app.post('/login', urlencodedParser, (req, res) => {
    const username = process.env.SHANA_USER;
```

```

const password = process.env.SHANA_PASS;
if(username === req.body.username && password === req.body.password){
    let authtoken = generateToken({
        mail: req.body.username
    }, process.env.TOKEN_SECRET, '10m');
    return res.cookie('authtoken', authtoken, {
        secure: true,
        httpOnly: true,
        sameSite: "lax" // lax option allows to send existing cookie to
server by clicking on link from an external site
    }).redirect('../statistics.html');
} else {
    return res.status(401).send();
}
});

app.get('/logout', checkToken, (req, res) => {
    res.clearCookie('authtoken');
    res.status(200).send();
});

app.get('/stats/getEditions', checkToken, (req, res) => {
    // send editions years
    res.status(200).send(VALID_YEARS);
});

app.get('/stats/visitors', checkToken, (req, res) => {
    // retrieve all users
    db.models.visitor.find({}, (err, visitors) => {
        if (err) return res.send(err);
        if (!visitors){
            return res.sendStatus(401)
        }
        res.send(visitors.length.toString());
    });
});
app.get('/stats/finished', checkToken, (req, res) => {
    if(VALID_YEARS.includes(req.query.year)) {
        let numberChalls = 0;
        db.models.flag.find({}, (err, flags) => {
            if (err) return res.status(500).send(err);
            if (!flags){
                return res.sendStatus(401)
            }
            res.send(flags.length);
        });
    }
});

```

ANNEXES

```
        return res.sendStatus(401)
    }
    for(let i = 0; i < flags.length; i++){
        if(flags[i].chall_name.startsWith(req.query.year)){
            numberChalls += 1;
        }
    }
});
// retrieve all users
db.models.user.find({}, (err, persons) => {
    if (err) return res.status(500).send(err);

    if (!persons) {
        return res.sendStatus(401)
    }
    let yearly_flagged = 0;
    for (let i = 0; i < persons.length; i++) {
        if(persons[i].flagged.filter(x =>
x.startsWith(req.query.year)).length === numberChalls) {
            yearly_flagged += 1
        }
    }
    return res.status(200).send(yearly_flagged.toString());
});
} else {
    return res.status(401).send();
}
});

app.get('/stats/flagPerChall', checkToken, (req, res) => {
    if(VALID_YEARS.includes(req.query.year)) {
        let numberChalls = 0;
        db.models.flag.find({}, (err, flags) => {
            if (err) return res.status(500).send(err);
            if (!flags){
                return res.sendStatus(401)
            }
            for(let i = 0; i < flags.length; i++){
                if(flags[i].chall_name.startsWith(req.query.year)){
                    numberChalls += 1;
                }
            }
        })
    }
})
```

```

    });
    // retrieve users
    db.models.user.find({}, (err, persons) => {
        if (err) return res.status(500).send(err);
        if (!persons){
            return res.sendStatus(401)
        }
        let result = new Array(numberChalls).fill(0);
        for(let i = 0; i < persons.length; i++){
            let yearly_flagged = persons[i].flagged.filter(x
=>x.startsWith(req.query.year)).length
            for (let j = 0; j < yearly_flagged; j++){
                result[j] += 1;
            }
        }
        return res.status(200).send(result);
    });
} else {
    return res.sendStatus(401);
}
});

// Store username information
app.post('/visitor', (req, res) => {
    // retrieve current hour from timestamp (round down to the current hour)
    db.models.visitor.findOne({hour_timestamp: Math.floor(Date.now() / (1000 * 60
* 60))}), (err, visitors) => {
        console.log(visitors);
        if (err) return res.send(err);
        // If this is the first visitor, we create a new entry with the current
        hour with the integer 1
        if (!visitors){
            console.log("toto")
            db.models.visitor.create({hour_timestamp: Math.floor(Date.now() /
(1000 * 60 * 60)), ctr:1}).then(() => {
                return res.sendStatus(200);
            }).catch((err) => {
                return res.send(err);
            });
        }
        // Otherwise, increment the inner counter
        else {

```

ANNEXES

```
    visitors.ctr += 1;
    visitors.save().then(() => {
        return res.sendStatus(200);
    }).catch((err) => {
        return res.send(err);
    });
}
});

app.get('/pin', (req, res) => {
let seed = Math.floor(Date.now() / 60000) // Different seed every minute

let rng = seedrandom(seed);
return res.send({ pin: Math.floor(rng() * 10000) })
});

app.post('/pin', (req, res) => {
let seed = Math.floor(Date.now() / 60000) // Different seed every minute

let rng = seedrandom(seed);
if (req.body.pin !== Math.floor(rng() * 10000)){
    return res.sendStatus(401);
}
let flag = process.env.CHALL_FLAGS_2021.split(';').filter((x) =>
x.startsWith('chall6'))[0].split('=')[1]
return res.send(flag)
});

// Init DB connection, and then bind port
db.init().then(() =>
    app.listen(process.env.PORT, () =>
        console.log(`app listening on port ${process.env.PORT}!`)
    )
);
```

-C Modèles Mongoose (db.js)

```

const mongoose = require('mongoose');
const assert = require('assert');
const {SHA3} = require('sha3');

// Connect to DB
mongoose.connect(`.${process.env.MONGO_URI}/test`, {
    useNewUrlParser: true,
    useUnifiedTopology: true,
    auth: {
        user: process.env.MONGO_USER,
        password: process.env.MONGO_PASS
    },
    authSource: 'admin',
});
// Create models
const Flag = mongoose.model('Flag', {chall_name: String, value: String});
const User = mongoose.model('User', {
    uuid: String,
    name: String,
    surname: String,
    mail: String,
    flagged: [String]
});
const Visitor = mongoose.model('Visitor', {
    hour_timestamp: Number,
    ctr: Number
})

// Init flags from environment variables
async function initFlags() {
    const flags_2020 = process.env.CHALL_FLAGS_2020.split(';');

    for await (const flag of flags_2020) {
        const elem = flag.split('=');
        assert(elem.length === 2);
        const hash = new SHA3(256);
        hash.update(elem[1]);
        if (!(await Flag.exists({chall_name: "2020_" + elem[0]})))
            await Flag.create({chall_name: "2020_" + elem[0], value:
}

```

ANNEXES

```
hash.digest('hex'))};  
}  
  
const flags_2021 = process.env.CHALL_FLAGS_2021.split(';');  
  
for await (const flag of flags_2021) {  
    const elem = flag.split('=');  
    assert(elem.length === 2);  
    const hash = new SHA3(256);  
    hash.update(elem[1]);  
    if (!(await Flag.exists({chall_name: "2021_"+elem[0]})))  
        await Flag.create({chall_name: "2021_"+elem[0]}, value:  
hash.digest('hex'))};  
}  
}  
  
exports.init = initFlags;  
exports.models = {flag: Flag, user: User, visitor:Visitor};
```

-D Base MySQL (`init.sql`)

```
drop database IF EXISTS dday;

create database dday;

use dday;

create table users(
    ID varchar(50),
    pass varchar(50) NOT NULL,
    PRIMARY KEY (ID)
);

create table posts(
    ID int,
    img varchar(50),
    nameLastname varchar(50),
    datepost varchar(50),
    PRIMARY KEY (ID)
);

insert into users value ("admin@admin.ch", "Ws3drftgzh$bjnimkl");
insert into users value ("jean.dupont@truite.ch", "Pass1234.");
insert into users value ("sille.vinpas@sini.ch", "flopPl0pPlipPlop");
insert into users value ("Fort@filip.pnato", "Vive_Sha3");

insert into posts (ID,img,nameLastname,datepost) value (1,"./img/
resto.jpg","zortak Nekmi", "29 Octobre 2123");

insert into posts (ID,img,nameLastname,datepost) value (2,"","brehuk cheunh", "25
Octobre 2123");

insert into posts (ID,img,nameLastname,datepost) value (3,"","bobo fatt", "30
Octobre 2123");

insert into posts (ID,img,nameLastname,datepost) value (4,"./img/
vaisseau.png","raj raj sknib", "01 Novembre 2123");

insert into posts (ID,img,nameLastname,datepost) value (5,"","zinwhu", "06
Novembre 2123");
```

ANNEXES

```
insert into posts (ID,img,nameLastname,datepost) value (6,"","zinwhu", "01  
Novembre 2123");  
  
insert into posts (ID,img,nameLastname,datepost) value (7,"","zinwhu", "27  
Octobre 2123");  
  
insert into posts (ID,img,nameLastname,datepost) value (8,"./img/  
resto2.jpg","zinwhu", "24 Octobre 2123");
```

-E Docker Compose (docker-compose.yml)

```

services:
  # Traefik reverse proxy
  traefik:
    image: "traefik:v2.10"
    restart: always
    command:
      - "--api.dashboard=false"
      - "--providers.docker=true"
      - "--providers.docker.exposedbydefault=false"
      - "--entrypoints.web.address=:80"
      # Global redirection to https
      - "--entrypoints.web.http.redirections.entrypoint.to=websecure"
      - "--entrypoints.web.http.redirections.entrypoint.scheme=https"
      - "--entrypoints.websecure.address=:443"
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - "/var/run/docker.sock:/var/run/docker.sock:ro"
  # Flagger + main backend
  backend:
    build: .
    environment:
      MONGO_URI: mongodb://mongo:27017
      WAIT_HOSTS: mysql:3306, mongo:27017
    restart: always
    volumes:
      - ".:/app"
      - "/app/node_modules"
    labels:
      # Expose the container in the traefik web UI
      - "traefik.enable=true"
      # Match rule to forward backend service
      - "traefik.http.routers.backend.rule=${HOST_RULE} && PathPrefix(`/
      backend`)"
      - "traefik.http.routers.backend.middlewares=backend-striprefix"
      - "traefik.http.middlewares.backend-striprefix.striprefix.prefixes=/
      backend"
      - "traefik.http.routers.backend.priority=100"
      # Enable TLS

```

ANNEXES

```
- "traefik.http.routers.backend.tls=true"
# Bound port for backend service
- "traefik.http.services.backend.loadbalancer.server.port=${PORT}"
# frontend
frontend:
  build:
    context: ../DigitalDay_APP
    dockerfile: Dockerfile
  restart: always
  volumes:
    - "../DigitalDay_APP:/DigitalDay_APP"

  labels:
    # Expose the container in the traefik web UI
    - "traefik.enable=true"
    # Match rule to forward frontend service
    - "traefik.http.routers.frontend.rule=${HOST_RULE}"
    - "traefik.http.routers.frontend.priority=10"
    # Enable TLS
    - "traefik.http.routers.frontend.tls=true"
    # Bound port for frontend service
    - "traefik.http.services.frontend.loadbalancer.server.port=${PORT_FRONT}"
# webssh
webssh:
  build:
    context: .
    dockerfile: Dockerfile_ssh
  restart: always
  labels:
    # Expose the container in the traefik web UI
    - "traefik.enable=true"
    # Match rule to forward ssh service
    - "traefik.http.routers.webssh.rule=${HOST_RULE} && (PathPrefix('/ssh') || PathPrefix('/static'))"
    - "traefik.http.routers.webssh.middlewares=webssh-striprefix"
    - "traefik.http.middlewares.webssh-striprefix.striprefix.prefixes=/ssh"
    - "traefik.http.routers.webssh.priority=110"
    # Enable TLS
    - "traefik.http.routers.webssh.tls=true"
    # Bound port for frontend service
    - "traefik.http.services.webssh.loadbalancer.server.port=${PORT_SSH}"
# SSH container
```

```
sshmachine:
  build:
    context: ../docker-ssh
    dockerfile: Dockerfile
    restart: always
# SSH container forensic 2021
sshmachine-galactic-forensic:
  build:
    context: ../docker-ssh-galactic-forensic
    dockerfile: Dockerfile
    restart: always
# SSH container for whois
sshmachine-whois:
  build:
    context: ../docker-ssh-whois
    dockerfile: Dockerfile
    restart: always
# Backend DB
mongo:
  image: mongo:4.4.1
  restart: always
  environment:
    MONGO_INITDB_ROOT_USERNAME: ${MONGO_USER}
    MONGO_INITDB_ROOT_PASSWORD: ${MONGO_PASS}
# Uncomment to connect to db with MongoDBCompass
#   ports:
#     - 42069:27017
  volumes:
    - "./mongo/:/data/db/"
# exposed MySQL server
mysql:
  image: mysql:5
  restart: always
  environment:
    MYSQL_DATABASE: dday
    MYSQL_USER: ${MYSQL_USER}
    MYSQL_PASSWORD: ${MYSQL_PASS}
    MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT}
    MYSQL_ROOT_HOST: mysql
  volumes:
    - ./mysql/init.sql:/docker-entrypoint-initdb.d/init.sql
```

-F Implémentation du jeu « Blackout » (**blackoutmain.js**)

```
//BLACKOUT 2025
PlayState = {};

//Const variables for cookie name
const cookieName_currentChall = "bk2025_xH92f_curr";
const cookieName_allChall = "bk2025_mP81x_all";
//URL to backend
const url_backend_flag_request = "/backend/2025/flag";
const url_backend_SQL = "";

// load game fonts here
PlayState.preload = function () {
    //Plugins
    this.game.add.plugin(PhaserInput.Plugin);
    //backgrounds
    this.game.load.image("background", "images/background-black.png");

    //levels datas
    this.game.load.json("level:1", "data/level01Blackout.json");

    //platforms for Blackout 2025
    this.game.load.image("ground0", "images/redZone.png");
    this.game.load.image("ground1", "images/redZone.png");
    this.game.load.image("ground2", "images/redZone.png");
    this.game.load.image("ground3", "images/redZone.png");
    this.game.load.image("ground4", "images/redZone.png");
    this.game.load.image("ground5", "images/redZone.png");
    this.game.load.image("ground6", "images/redZone.png");
    this.game.load.image("ground7", "images/redZone.png");
    this.game.load.image("ground8", "images/redZone.png");

    //texture for plateform when accessible
    this.game.load.image("finishGround", "images/greenZone.png");
    //invisible ground
    this.game.load.image("inv1", "images/ground.png");
    //hero
    this.game.load.spritesheet('hero', 'images/herov4.png', 27, 50);

};
```

```
PlayState.init = function () {
    this.game.renderer.renderSession.roundPixels = true;
};

///////////////////////////////
// HERO
///////////////////////////////

function Hero(game, x, y) {
    Phaser.Sprite.call(this, game, x, y, 'hero');
    this.anchor.set(0, 0.5);
    this.game.physics.enable(this);
    this.body.collideWorldBounds = true;
    this.positionx = x;
    this.direction = 1;
    this.accessibleChall = new Set();

    this.animations.add('stop', [0]);
    this.animations.add('run', [0, 1, 2, 3, 4, 5, 6], 8, true); // 8fps looped
}

// inherit from Phaser.Sprite
Hero.prototype = Object.create(Phaser.Sprite.prototype);
Hero.prototype.constructor = Hero;
Hero.prototype.move = function (direction) {
    const SPEED = 300;
    this.body.velocity.x = direction * SPEED;
    if (this.body.velocity.x < 0) {
        this.scale.x = -1;
    } else if (this.body.velocity.x > 0) {
        this.scale.x = 1;
    }
};
Hero.prototype._getAnimationName = function () {
    let name = 'stop'; // default animation

    if (this.body.velocity.x !== 0) {
        name = 'run';
    }
    return name;
};
Hero.prototype.update = function () {
```

ANNEXES

```
// update sprite animation, if it needs changing
let animationName = this._getAnimationName();
if (this.animations.name !== animationName) {
    this.animations.play(animationName);
}
};

///////////////////////////////
// load and spawn methodes to create sprites, background ...
///////////////////////////////

//create game entities and set up world here
PlayState.create = function () {
    this.game.add.image(0, 0, "background");
    this.data = this.game.cache.getJSON("level:1");
    this._loadLevel(this.data);
    //mouse pointer position
    this.game.pointerX = null;
};

PlayState._loadLevel = function (data) {
    // create all the groups/layers that we need
    this.platforms = this.game.add.group();
    this.invisible_grounds = this.game.add.group();

    //spawn all plateforms
    data.platforms.forEach(this._spawnPlateform, this);
    data.invisible_grounds.forEach(this._spawninvisibleGround, this);
    // spawn hero and enemies
    this._spawnCharacters({ hero: data.hero });
    const GRAVITY = 1200;
    this.game.physics.arcade.gravity.y = GRAVITY;

    // define submit hint button
    submitBtn = document.getElementById("sumbitHint");
    submitBtn.onclick = function () {
        //VERIFY SI OK OR NOT //
        let currentChall = getCookie(cookieName_currentChall);
        let flag = document.getElementById("inputHint").value;
        flag = flag.replace(/ /g, "");
        flag = flag.normalize("NFD").replace(/[\u0300-\u036f]/g, "");
    };
};
```

```

let msgBox = "";
if (flag === "") {
    msgBox = "Veuillez remplir le champ de réponse avant de valider !";
    showPopupValidation();
} else {
    //send request
    let content = {
        chall: currentChall,
        flag: flag.toLowerCase(),
    };
    console.log(content);
    let contentJson = JSON.stringify(content);
    var xhr = new XMLHttpRequest();

    xhr.open("POST", url_backend_flag_request, true);
    xhr.setRequestHeader("Content-Type", "application/json");
    xhr.withCredentials = true;

    xhr.onreadystatechange = function () {
        //Appelle une fonction au changement d'état.
        if (this.readyState === XMLHttpRequest.DONE) {
            if (this.status === 200) {
                //if chall resolved, adding challenge and the next(updatePlatform)
                to the accessible challenges
                PlayState.hero.accessibleChall.add(currentChall);
                updatePlatform(currentChall);
                document.getElementById("inputHint").value = "";
                msgBox =
                    "Bravo ! Vous avez résolu une étape, passez à la suivante !";
            }
            if (this.status === 404) {
                //bad challenge name
                msgBox =
                    "Hum... une erreur étrange à eu lieu, recharge le site web et
recommence !";
            }
            if (this.status === 401) {
                //bad challenge flag
                msgBox = "Dommage, ce n'est pas la bonne réponse, persévère !";
            }
            showPopupValidation();
        }
    }
}

```

ANNEXES

```
};

xhr.send(contentJson);
}

function showPopupValidation() {
    var popup = document.getElementById("popupSubmitChall");
    var contentText = document.getElementById("popupSubmitChallContent");
    contentText.innerText = msgBox;
    popup.style.display = "block";
    closeBtn = document.getElementById("popupSubmitChallCloseButton");
    closeBtn.onclick = function () {
        popup.style.display = "none";
    };
    continueBtn = document.getElementById("popupSubmitChallContinue");
    continueBtn.onclick = function () {
        popup.style.display = "none";
    };
}
};

// if the user has never started a challenge, load popup Intro on restart page
let currentChall = getCookie(cookieName_currentChall);
if (currentChall === null || currentChall === "chall0") {
    spawnPopup("chall0");
}

// on page restart, load the iframe for the last challenge started.
var urlChall = null;
data.platforms.forEach((p) => {
    if (p.idChall === currentChall) {
        urlChall = p.urlChall;
    }
});
loadIframe(currentChall, urlChall);

// on page restart, get list of accessible challenge (or finished) to apply the right platform.
let accessibleChall = getCookie(cookieName_allChall);
if (accessibleChall != null) {
    PlayState.hero.accessibleChall = new Set(JSON.parse(accessibleChall));
    updatePlatform();
}
};
```

```
PlayState._spawnPlateform = function (platform) {
    let sprite = this.platforms.create(platform.x, platform.y, platform.image);
    sprite.inputEnabled = true;
    sprite.idChall = platform.idChall;
    sprite.urlChall = platform.urlChall;
    sprite.IsAccess = platform.IsAccess;
    sprite.events.onInputDown.add(onclickPlateform, this);

    //on click on a platform, set mouse pointer position to platform position then
    //show the link popup
    function onclickPlateform(platform) {
        if (PlayState.hero.accessibleChall.has(platform.idChall)) {
            this.game.pointerX = platform.position.x;
            spawnPopup(platform.idChall, platform.urlChall);
        }
    }
};

PlayState._spawninvisibleGround = function (block) {
    let sprite = this.invisible_grounds.create(block.x, block.y, block.image);
    this.game.physics.enable(sprite);
    sprite.body.immovable = true;
    sprite.body.allowGravity = false;
    sprite.visible = false;
};

PlayState._spawnCharacters = function (data) {
    // spawn hero
    this.hero = new Hero(this.game, data.hero.x, data.hero.y);
    this.hero.body.allowGravity = false;
    this.game.add.existing(this.hero);
};

PlayState._handleCollisions = function () {
    this.game.physics.arcade.collide(this.hero, this.invisible_grounds);
};

///////////////////////////////
// game engine
/////////////////////////////
```

ANNEXES

```
PlayState.update = function () {
    this._handleCollisions();

    // move character where the mouse pointer click (on platform)
    if (this.game.pointerX != null) {
        if (
            this.hero.direction === -1 &&
            this.hero.positionx < this.game.pointerX
        ) {
            this.hero.direction = 1;
        }
        if (this.hero.direction === 1 && this.hero.positionx > this.game.pointerX) {
            this.hero.direction = -1;
        }
        this.hero.move(this.hero.direction);

        var stop = false;
        // if it's overlapping the mouse, don't move any more
        if (
            this.hero.direction === -1 &&
            this.hero.world.x <= this.game.pointerX + 30
        ) {
            stop = true;
        }
        if (this.hero.direction === 1 && this.hero.world.x >= this.game.pointerX) {
            stop = true;
        }
        if (stop) {
            this.hero.body.velocity.setTo(0, 0);
            this.hero.positionx = this.game.pointerX;
        }
    } else {
        this.hero.body.velocity.setTo(0, 0);
    }
};

window.onload = function () {
    // Check for mobile user agent
    var mobile =
        /iphone|ipad|ipod|android|blackberry|mini|windows\sce|palm/i.test(
            navigator.userAgent.toLowerCase()
        );
}
```

```
var istablet =
    /ipad|android|android 3.0|xoom|sch-i800|playbook|tablet|kindle/i.test(
    navigator.userAgent.toLowerCase()
);
var IsIE10 = /MSIE 10/i.test(navigator.userAgent);
var IsIE9 = /MSIE 9/i.test(navigator.userAgent);
var IsIE11 = /rv:11.0/i.test(navigator.userAgent);
var isEdge = /Edge\//.i.test(navigator.userAgent);
if (mobile || istablet) {
    alert(
        "Ce jeu d'ethical hacking est fait pour ordinateur et non pas téléphone."
    );
}
if (IsIE11 || IsIE10 || IsIE9 || isEdge) {
    alert(
        "Nous conseillons l'utilisation des navigateurs Google Chrome ou Firefox pour le bon fonctionnement du jeu. Vous pouvez les installer facilement via une simple recherche Internet."
    );
}
let game = new Phaser.Game(1400, 100, Phaser.AUTO, "game");
game.state.add("play", PlayState);
game.state.start("play");
};

///////////////////////////////
// game internal function
///////////////////////////////

function updatePlatform(currentChall) {
    if (currentChall != null) {
        // extract number of the current chall (7 from chall7) and define next accessible challenge
        let matches = currentChall.match(/\d+/g);
        let nextchall = parseInt(matches[0]) + 1;
        nextchall = "chall" + nextchall;
        PlayState.hero.accessibleChall.add(nextchall);
        setCookie(
            cookieName_allChall,
            JSON.stringify(Array.from(PlayState.hero.accessibleChall))
        );
    }
}
```

ANNEXES

```
// load the right texture for accessible challenge/platform
PlayState.platforms.forEach(function (element, index) {
    if (PlayState.hero.accessibleChall.has(element.idChall)) {
        element.loadTexture("finishGround");
    }
});
}

// Modification fonction loadIframe
function loadIframe(idChall, urlChall) {
    var iframe = document.getElementById("iframeChall");

    if (idChall === "chall3") {
        // Chall3 -> navigation
        var queryParams = new URLSearchParams(window.location.search);
        var dirParam = queryParams.get("dir");

        // Mapping paramètres dir vers html
        var fileMapping = {
            "/": "dir.html",
            "/shared": "shared.html",
            "/shared/": "shared.html",
            "/public": "public.html",
            "/public/": "public.html",
            "/archives": "archives.html",
            "/archives/": "archives.html",
            "/archives/2020": "archives_2020.html",
            "/archives/2020/": "archives_2020.html",
            "/archives/2021": "archives_2021.html",
            "/archives/2021/": "archives_2021.html",
            "/archives/2022": "archives_2022.html",
            "/archives/2022/": "archives_2022.html",
            "/archives/2023": "archives_2023.html",
            "/archives/2023/": "archives_2023.html",
            "/archives/2024": "archives_2024.html",
            "/archives/2024/": "archives_2024.html",
            "/archives/2025": "archives_2025.html",
            "/archives/2025/": "archives_2025.html",
        };
        if (dirParam && fileMapping[dirParam]) {
            // Construire chemin vers html
        }
    }
}
```

```
var basePath = urlChall.replace("index.html", "");
iframe.src = basePath + fileMapping[dirParam];
} else if (dirParam) {
    // Paramètre dir inconnu donc vers dir=/
    var basePath = urlChall.replace("index.html", "");
    iframe.src = basePath + "dir.html";
} else {
    // Pas de paramètre dir vers index.html
    iframe.src = urlChall;
}
} else {
    // Normal autres challs
    iframe.src = urlChall;
}
}

// Navigation chall3
function navigateToDirectory(dirPath) {
    // MAJ URL
    var newUrl = window.location.pathname + "?dir=" + dirPath;
    history.replaceState(null, null, newUrl);

    // Recharge iframe
    var iframe = document.getElementById("iframeChall");
    if (iframe && iframe.src) {
        // Chemin de base chall3
        var basePath = iframe.src.split("/").slice(0, -1).join("/") + "/";
        // Mapping paramètres dir vers html
        var fileMapping = {
            "/": "dir.html",
            "/shared": "shared.html",
            "/public": "public.html",
            "/archives": "archives.html",
            "/archives/2020": "archives_2020.html",
            "/archives/2021": "archives_2021.html",
            "/archives/2022": "archives_2022.html",
            "/archives/2023": "archives_2023.html",
            "/archives/2024": "archives_2024.html",
            "/archives/2025": "archives_2025.html",
        };
    }
}
```

ANNEXES

```
if (fileMapping[dirPath]) {
    iframe.src = basePath + fileMapping[dirPath];
}
}

function spawnPopup(idChall, urlChall) {
    // get the popup html element from game.html
    var popup = document.getElementById(idChall);
    var beginBtn = null;

    // get current query params
    var queryParams = new URLSearchParams(window.location.search);

    // if chall3, don't set dir parameter initially - let it start on index.html
    if (idChall === "chall3") {
        // Ne pas définir de paramètre dir au début pour commencer sur index.html
        queryParams.delete("dir");
        history.replaceState(
            null,
            null,
            queryParams.toString() ? "?" + queryParams.toString() : ""
        );
    } else {
        queryParams.delete("page");
        queryParams.delete("dir");
        history.replaceState(null, null, "?" + queryParams.toString());
    }

    // if chall0 the normally start challenge button will close the popup and get
    // next chall accessible, update plateform
    if (idChall === "chall0") {
        closeBtbnbis = document.getElementById("intro_close");
        closeBtbnbis.onclick = function () {
            popup.style.display = "none";
            setCookie(cookieName_currentChall, idChall);
            PlayState.hero.accessibleChall.add(idChall);
            updatePlatform(idChall);
        };
    }
    if (idChall === "chall8") {
    }
```

```

// else, start challenge button will close popup then load iframe, set cookie
for current chall
else {
    beginBtn = document.getElementById(idChall + "_begin");
    if (beginBtn != null)
        beginBtn.onclick = function () {
            loadIframe(idChall, urlChall);
            popup.style.display = "none";
            setCookie(cookieName_currentChall, idChall);
            //hide hint for all popup
            $(".indice_text").css("display", "none");
        };
}
// show the popup, close button will close popup and hide hint for all popup
popup.style.display = "block";
closeBtn = document.getElementById(idChall + "_close");
closeBtn.onclick = function () {
    popup.style.display = "none";
    $(".indice_text").css("display", "none");
};
}

function setCookie(name, value, expires, path, domain, secure) {
    document.cookie =
        name +
        "=" +
        escape(value) +
        (expires ? "; expires=" + expires.toGMTString() : "") +
        (path ? "; path=" + path : "") +
        (domain ? "; domain=" + domain : "") +
        (secure ? "; secure" : "");
}

function getCookie(name) {
    var arg = name + "=";
    var alen = arg.length;
    var clen = document.cookie.length;
    var i = 0;
    while (i < clen) {
        var j = i + alen;
        if (document.cookie.substring(i, j) === arg) return getCookieVal(j);
        i = document.cookie.indexOf(" ", i) + 1;
    }
}

```

ANNEXES

```
    if (i === 0) break;
}
return null;

function getCookieVal(offset) {
    var endstr = document.cookie.indexOf(";", offset);
    if (endstr === -1) endstr = document.cookie.length;
    return unescape(document.cookie.substring(offset, endstr));
}

function show_hide_indice() {
    if ($.indice_text.css("display") === "block") {
        $(".indice_text").css("display", "none");
    } else {
        $(".indice_text").css("display", "block");
    }
}
```

-G Implémentation du bot pour le challenge 6 de 2025 (`bot.js`)

```
import puppeteer from "puppeteer";
import express from "express";
import cors from "cors";
import { v4 as uuidv4 } from "uuid";
import http from "http";
import https from "https";
import fs from "fs";

class AdminBot {
  constructor() {
    this.browser = null;
    this.page = null;
    this.isRunning = false;
    this.sessions = new Map(); // Store sessions
    this.app = express();
    this.setupAPI();
  }

  setupAPI() {
    this.app.use(
      cors({
        origin: true,
        credentials: true,
      })
    );
    this.app.use(express.json());

    // Nouvelle session
    this.app.post("/api/session/create", (req, res) => {
      const sessionId = uuidv4();
      const session = {
        id: sessionId,
        messages: [],
        createdAt: Date.now(),
        lastActivity: Date.now(),
      };
      this.sessions.set(sessionId, session);
      console.log(`[BOT-API] Nouvelle session : ${sessionId}`);
      res.json({ sessionId });
    });
  }
}
```

ANNEXES

```
// Ajout message session
this.app.post("/api/session/:sessionId/message", (req, res) => {
  const { sessionId } = req.params;
  const { text, isUser } = req.body;

  const session = this.sessions.get(sessionId);
  if (!session) {
    return res.status(404).json({ error: "Session non trouvée" });
  }

  const message = {
    id: uuidv4(),
    text,
    isUser,
    timestamp: Date.now(),
    processed: false,
  };

  session.messages.push(message);
  session.lastActivity = Date.now();

  console.log(
    `[BOT-API] Message ${sessionId}: ${text.substring(0, 50)}...`
  );

  // Si mess user -> traitement bot
  if (isUser) {
    setTimeout(() => this.processUserMessage(sessionId, message), 1000);
  }

  res.json({ success: true, messageId: message.id });
});

// Récupère messages session
this.app.get("/api/session/:sessionId/messages", (req, res) => {
  const { sessionId } = req.params;
  const session = this.sessions.get(sessionId);

  if (!session) {
    return res.status(404).json({ error: "Session non trouvée" });
  }
```

```
res.json({ messages: session.messages });
});

// Démarrer serveur sur port 3001
const port = process.env.BOT_API_PORT || 3001;
this.app.listen(port, () => {
  console.log(`[BOT-API] Serveur démarré sur le port ${port}`);
});

async processUserMessage(sessionId, message) {
  const session = this.sessions.get(sessionId);
  if (!session) return;

  console.log(`[BOT] Traitement ${sessionId}: ${message.text}`);

  let botResponse = `Message reçu : ${message.text}`;
  let stolenCookie = null;

  // Vérifier XSS vol cookies
  const isCookieStealingXSS =
    message.text.includes("document.cookie") ||
    message.text.includes("document['cookie']") ||
    message.text.includes('document["cookie"]') ||
    (message.text.includes("cookie") && (
      message.text.includes("<script>") ||
      message.text.includes("javascript:") ||
      message.text.includes("onerror=") ||
      message.text.includes("onload=")
    ));
}

if (
  message.text.includes("<script>") ||
  message.text.includes("onerror=") ||
  message.text.includes("onload=") ||
  message.text.includes("javascript:") ||
  message.text.includes("<a href=")
) {

  if (isCookieStealingXSS) {
    try {
```

ANNEXES

```
// Exécuter payload joueur et reprend résultat
try {
    this.capturedOutput = "";
    this.isCapturingXSS = true;

    let jsCode = "";

    // Extraire JavaScript selon payload
    const scriptMatch = message.text.match(/<script[^>]*>(.*)<\\/script>/
s);
    const javascriptMatch = message.text.match(/javascript:([^"]")*/);

    if (scriptMatch && scriptMatch[1]) {
        // Payload <script>
        jsCode = scriptMatch[1].trim();
        await this.page.evaluate((code) => {
            eval(code);
        }, jsCode);
    } else if (javascriptMatch && javascriptMatch[1]) {
        // Payload javascript
        jsCode = javascriptMatch[1].trim();
        const allCookies = await this.page.cookies();
        const cookieString = allCookies.map(cookie => `${cookie.name}
=${cookie.value}`).join(' ');
        this.capturedOutput = cookieString;

        await this.page.evaluate((code) => {
            eval(code);
        }, jsCode);
    } else {
        // autres xss
        await this.page.setContent(`<html><body>${message.text}</body></
html>`);
    }
}

// si lien javascript -> simule clic
if (message.text.includes('<a href="javascript:')) {
    try {
        await this.page.click('a');
    } catch (e) {
    }
}
```

```
}

await new Promise(resolve => setTimeout(resolve, 1000));

// stop capture
this.isCapturingXSS = false;

// type d'action demandée
if (scriptMatch && scriptMatch[1]) {
    jsCode = scriptMatch[1].trim();
} else if (javascriptMatch && javascriptMatch[1]) {
    jsCode = javascriptMatch[1].trim();
}

// Remplacer par cookie bot
if (message.text.includes('<a href="javascript:' ) &&
jsCode.includes('alert()')) {
    const escapedCookies = this.capturedOutput.replace(/"/g, '\\"').replace(/\n/g, '\\n');
    botResponse = `Message reçu:

<a href="javascript:alert('Cookies admin volés:\n${escapedCookies}'")>Click me</a>`;

} else if (jsCode.includes('alert()')) {
    // Si alerte <script> -> alerte
    if (this.capturedOutput) {
        const escapedCookies = this.capturedOutput.replace(/"/g, '\\"').replace(/\n/g, '\\n');
        botResponse = `Message reçu: ${message.text}

<script>alert("Cookies admin volés:\n${escapedCookies}"</script>`;
    }
} else if (jsCode.includes('console.log()')) {
    // Si console.log -> console.log
    if (this.capturedOutput) {
        const escapedCookies = this.capturedOutput.replace(/"/g, '\\"').replace(/\n/g, '\\n');
        botResponse = `Message reçu: ${message.text}

<script>console.log("Cookies admin volés:", "${escapedCookies}"</script>`;
    }
} else if (jsCode.includes('fetch()')) {
```

ANNEXES

```
// Si fetch -> dans chatbot
const allCookies = await this.page.cookies();
const cookieString = allCookies.map(cookie => `${cookie.name}
=${cookie.value}`).join('; ');
botResponse = `Message reçu: ${message.text}`

Cookies admin volés: ${cookieString}`;
} else if (this.capturedOutput) {
    // Autre
    botResponse = `Message reçu: ${message.text}`

Cookies admin volés: ${this.capturedOutput}`;
}
} catch (error) {
    this.isCapturingXSS = false;
    botResponse = `Message reçu: ${message.text}`

Erreur exécution`;
}
} catch (error) {
    console.error(`[BOT] Erreur XSS:`, error);
}
} else {
    botResponse = `Message reçu: ${message.text}`;
}
} else {
    // Commandes bot
    const command = message.text.toLowerCase().trim();

    if (command === "help") {
        botResponse = `Message reçu: ${message.text}`

Commandes disponibles :
• help - Afficher l'aide
• status - État du système
• admin - Contacter l'administrateur
• stats - Afficher les statistiques
• test - Tester la connexion`;
    } else if (command === "status") {
        botResponse = `Message reçu: ${message.text}`

Système opérationnel
```

```

Bot admin actif et surveille les messages`;
} else if (command === "admin") {
    botResponse = `Message reçu: ${message.text}

Demande d'assistance admin enregistrée`;
} else if (command === "stats") {
    botResponse = `Message reçu: ${message.text}

Statistiques du système :
• Sessions actives: ${this.sessions.size}
• Messages traités: ${Array.from(this.sessions.values()).reduce((total, session)
=> total + session.messages.length, 0)}`;
} else if (command === "test") {
    botResponse = `Message reçu: ${message.text}

Connexion OK
Bot admin opérationnel
Latence: ~${Math.floor(Math.random() * 50 + 10)}ms`;
} else {
    botResponse = `Message reçu: ${message.text}`;
}
}

// Réponse bot
const adminReply = {
    id: uuidv4(),
    text: botResponse,
    isUser: false,
    timestamp: Date.now(),
    processed: true,
};

session.messages.push(adminReply);
session.lastActivity = Date.now();

// Message user traité
message.processed = true;

console.log(`[BOT] Réponse pour ${sessionId}`);
}

cleanOldSessions() {

```

```
const twoHoursAgo = Date.now() - 2 * 60 * 60 * 1000;
let cleaned = 0;

for (const [sessionId, session] of this.sessions.entries()) {
    if (session.createdAt < twoHoursAgo) {
        this.sessions.delete(sessionId);
        cleaned++;
    }
}

if (cleaned > 0) {
    console.log(`[BOT] ${cleaned} sessions cleaned`);
}
}

async start() {
    try {
        console.log("[BOT] Démarrage bot admin");

        // Démarrer Puppeteer avec contexte
        this.browser = await puppeteer.launch({
            headless: "new",
            ignoreHTTPSErrors: true,
            args: [
                "--no-sandbox",
                "--disable-setuid-sandbox",
                "--disable-dev-shm-usage",
                "--disable-accelerated-2d-canvas",
                "--no-first-run",
                "--no-zygote",
                "--disable-gpu",
                "--ignore-certificate-errors",
                "--ignore-ssl-errors",
                "--disable-web-security",
                "--disable-features=VizDisplayCompositor",
                "--disable-web-security",
                "--allow-running-insecure-content",
            ],
        });
    }

    this.page = await this.browser.newPage();
    await this.page.setUserAgent(
```

```
"Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/91.0.4472.124 Safari/537.36"
);

// Variables capturer outputs XSS
this.capturedOutput = "";
this.isCapturingXSS = false;

// Capturer alertes logs
this.page.on('dialog', async (dialog) => {
    console.log(`[BOT-ALERT] ${dialog.type()}: ${dialog.message()}`);

    // Si capture pour xss -> stocke
    if (this.isCapturingXSS) {
        this.capturedOutput = dialog.message();
    }

    await dialog.accept();
});

this.page.on('console', (msg) => {
    console.log(`[BOT-CONSOLE] ${msg.type()}: ${msg.text()}`);

    // Si capture pour xss et c'est un log -> stocke
    if (this.isCapturingXSS && msg.type() === 'log') {
        this.capturedOutput = msg.text();
    }
});

// Serveur HTTP simple contexte
this.botContextServer = http.createServer((req, res) => {
    res.writeHead(200, { "Content-Type": "text/html" });
    res.end(
        "<html><body><h1>Bot Admin Context</h1><p>Contexte pour cookies
Puppeteer</p></body></html>"
    );
});

this.botContextServer.listen(3002, () => {
    console.log("[BOT] Serveur contexte démarré port 3002");
});
```

ANNEXES

```
// Domaine valide
await this.page.goto("http://localhost:3002", {
    waitUntil: "networkidle0",
});

// Cookies admin bot
await this.page.setCookie({
    name: "__xsrf",
    value: "2|a8452827|a77b2cb9c1b7c7b20fa273a9805236a9|1757511565",
    path: "/",
    expires: -1,
    httpOnly: false,
    secure: false,
    sameSite: "Lax",
});

await this.page.setCookie({
    name: "bk2025_mp81x_all",
    value:
"%5B%22chall0%22%2C%22chall1%22%2C%22chall2%22%2C%22chall3%22%2C%22chall4%22%2C%22chall5%22%2C%22
    path: "/",
    expires: -1,
    httpOnly: false,
    secure: false,
    sameSite: "Lax",
});

await this.page.setCookie({
    name: "bk2025_xH92f_curr",
    value: "chall6",
    path: "/",
    expires: -1,
    httpOnly: false,
    secure: false,
    sameSite: "Lax",
});

await this.page.setCookie({
    name: "admin",
    value: "ADM1N_535510N_TOKEN25",
    path: "/",
    expires: -1,
```

```
        httpOnly: false,
        secure: false,
        sameSite: "Lax",
    });
}

// Vérif cookie est défini
const cookies = await this.page.cookies();
console.log("[BOT] Cookies :", cookies);

this.isRunning = true;
console.log(
    "[BOT] Bot admin démarré avec Puppeteer"
);

// Nettoyage sessions auto
setInterval(() => {
    this.cleanOldSessions();
}, 60 * 60 * 1000);

console.log("[BOT] Nettoyage sessions activé");

// Monitoring sessions
this.startMonitoring();
} catch (error) {
    console.error("[BOT] Erreur démarrage:", error);
}
}

async startMonitoring() {
    console.log("[BOT] Surveillance sessions");

    while (this.isRunning) {
        try {
            console.log(`[BOT] Sessions actives: ${this.sessions.size}`);

            if (this.sessions.size > 0) {
                console.log("[BOT] Détails sessions:");
                for (const [sessionId, session] of this.sessions.entries()) {
                    const unprocessed = session.messages.filter(
                        (msg) => msg.isUser && !msg.processed
                    ).length;
                    const shortId = sessionId.substring(0, 12);
                    console.log(`[BOT] Session ID: ${shortId}, Unprocessed messages: ${unprocessed}`);
                }
            }
        } catch (error) {
            console.error("[BOT] Erreur dans le monitoring:", error);
        }
    }
}
```

```

        console.log(
            `  ↴ ${shortId}... : ${session.messages.length} messages
(${unprocessed} non traités)`
        );

        if (unprocessed > 0) {
            console.log(
                `[BOT] Traitement automatique`
            );
        }
    }
}

console.log("[BOT] Bot admin OK");

// 30s pause
await this.sleep(30000);
} catch (error) {
    console.error("[BOT] Erreur surveillance:", error);
    await this.sleep(10000);
}
}

async sleep(ms) {
    return new Promise((resolve) => setTimeout(resolve, ms));
}

async stop() {
    console.log("[BOT] Arrêt bot admin");
    this.isRunning = false;
    if (this.page) {
        await this.page.close();
        console.log("[BOT] Page Puppeteer fermée");
    }
    if (this.browser) {
        await this.browser.close();
        console.log("[BOT] Navigateur Puppeteer fermé");
    }
    if (this.botContextServer) {
        this.botContextServer.close();
        console.log("[BOT] Serveur contexte fermé");
    }
}

```

```
        }
        console.log("[BOT] Bot admin arrêté");
    }
}

const bot = new AdminBot();

// Arrêt
process.on("SIGINT", async () => {
    console.log("\n[BOT] Signal d'arrêt reçu");
    await bot.stop();
    process.exit(0);
});

process.on("SIGTERM", async () => {
    console.log("\n[BOT] Signal de terminaison reçu");
    await bot.stop();
    process.exit(0);
});

// Démarrer bot
bot.start().catch((error) => {
    console.error("[BOT] Erreur :", error);
    process.exit(1);
});

export default AdminBot;
```

-H Présentation des challenges (ancienne version des défis)



Conception d'un nouveau serious game autour du «Ethical Hacking»

Extension du Jeu « Shana a disparu »

Camille Koestli

Hes-SO
Haute Ecole Supérieure
de Suisse Occidentale



Sommaire

1. Présentation du projet
2. Choix du scénario
3. Résumé du scénario
4. Challenges
 1. Mail Contagieux
 2. Portail VPN Fantôme
 3. Archives
 4. Clé cachée dans les commentaires
 5. Script d'infection
 6. Chat KO
 7. Blocage ciblé

Hes-SO
Haute Ecole Supérieure
de Suisse Occidentale

Présentation du projet

- Augmentation de l'intérêt sur la cybersécurité
- Potentiel des serious games pour rendre l'apprentissage ludique
- «Shana a disparu» a eu un grand succès mais trop de personnes l'ont complété et terminé
- Objectif : créer un nouveau serious game avec une approche narrative tout en proposant des challenges techniques



Choix du scénario

Scénario réaliste
Black out dans le Centre Hospitalier Horizon Santé
Ransomware dans un milieu hospitalier



Scénario aventure

*Opération « CipherFox »
Infiltration*

Vol de données dans une entreprise



Scénario science-fiction

Fuite de l'Acheron
S'échapper d'un vaisseau spatial



CONCEPTION D'UN NOUVEAU SERIOUS GAME AUTOUR DU « ETHICAL HACKING »

Résumé du scénario



- L'hôpital subit une attaque par ransomware
- Black out des systèmes informatiques et des services critiques
- Vol des données sensibles concernant les patients
- Le joueur incarne un membre de l'équipe de sécurité
- Objectif : Résoudre les défis pour empêcher les attaquants de poursuivre leurs attaques

Hes-so

Haute Ecole Suisse Supérieure
de la Santé et de la Sécurité



CONCEPTION D'UN NOUVEAU SERIOUS GAME AUTOUR DU « ETHICAL HACKING »

Challenge 1 : Mail Contagieux



- Le point d'entrée des attaquants est un email de phishing reçu
- Le joueur analyse l'email dans le but de retrouver le faux nom de domaine qu'ils ont utilisé
- Compétences travaillées : OSINT et forensic
- Ce challenge a pour objectif de sensibiliser aux signes d'un courriel d'hameçonnage

Hes-so

Haute Ecole Suisse Supérieure
de la Santé et de la Sécurité



Challenge 2 : Portail VPN Fantôme

- Une fois le faux domaine identifié, le joueur découvre qu'il héberge un faux portail vpn pour exfiltrer les données
- Le joueur doit donc réussir à contourner le formulaire de connexion équipé d'un WAF
- Compétences travaillées : injection SQL
- Ce challenge a pour objectif de sensibiliser aux failles d'injection et montre qu'une protection insuffisante peut être contournée facilement



Hes-so



Challenge 3 : Archives compromises



- Dans le portail malveillant, le joueur voit une section «Document» avec un bouton pour télécharger le rapport du jour
- Les attaquants utilisent ce portail pour héberger les informations sensibles
- Le joueur donc faire une path traversal pour retrouver où sont stocker les dossiers concernant les patients
- Compétences travaillées : path traversal et analyse HTML
- Ce challenge sensibilise aux failles de type path traversal qui permet d'accéder à des fichiers sensibles

Hes-so

Challenge 4 : Clé cachée dans les commentaires

- Une fois les dossiers sensibles découverts, le joueur remarque qu'il y a un fichier zip mais il est chiffré
- Le joueur devra donc faire une investigation des métadonnées pour découvrir un commentaire contenant le SHA-1 qui chiffre le dossier
- Compétences travaillées : analyse des métadonnées et cryptographie
- Ce challenge montre l'importance de vérifier ces métadonnées mais aussi l'importance de la cryptographie



Hes-so
Haute Ecole Spécialisée
de Suisse Occidentale

Challenge 5 : Script d'injection



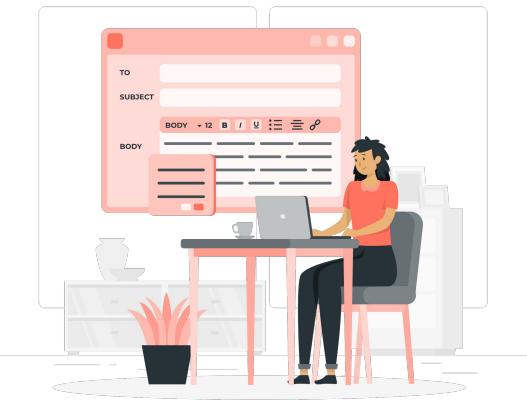
- Une fois le zip décompressé, le joueur voit qu'il y a un script Powerhell mais il est brouillé qui sert à établir la connexion vers un serveur
- Il va décoder le fichier pour retrouver l'URL du serveur
- Compétences travaillées : dé-obfuscation
- Ce challenge montre comment les attaquants camouflent leurs logiciels et comment les analyser

Hes-so
Haute Ecole Spécialisée
de Suisse Occidentale



Challenge 6 : Chat KO

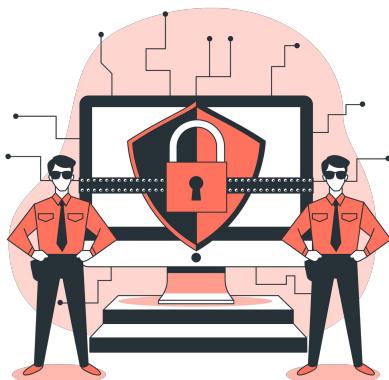
- Une fois dans le serveur, une page affiche un forum interne
- Le joueur réalise une attaque XSS pour mettre le serveur hors service
- Compétences travaillées : Attaque XSS
- Ce challenge montre la gravité d'une entrée utilisateur non échappée



Hes-so



Challenge 7 : Blocage ciblé



- Une fois leur serveur HS, le joueur doit identifier l'adresse IP de l'attaquant pour la bloquer.
- Le joueur va se connecter sur le VPN de l'hôpital afin d'ajouter l'IP à la liste noire du pare-feu
- Compétences travaillées : défense et journalisation de log
- Ce challenge montre l'importance de surveiller les logs et de gérer les adresses IP suspectes

Hes-so
Haute Ecole Spécialisée
de Suisse Occidentale



Conclusion



- Ce scénario réaliste s'inspire de situation réaliste avec des attaques par ransomware
- Ce scénario mélange narration et apprentissage technique
- Approche progressive et immersive
- Permet d'aborder plusieurs aspects de la cybersécurité

Hes-so

The Art Circle | Sion | Lausanne | Vevey | Yverdon

