Retour d'expérience sur l'organisation d'un CTF

Rétrospective de 5 ans de FCSC

Tristan Claverie, Emilien Court, Ambre Iooss, Jérémy Jean, Matthieu Olivier et Adrien Thuau
<Prénom>.<Nom>@ssi.gouv.fr

Résumé. Tous les ans depuis 2019, l'ANSSI organise le FCSC, une compétition CTF en ligne dont l'un des buts est de constituer une équipe nationale qui participera à l'ECSC organisé par l'ENISA. Depuis sa création il y a cinq ans, le FCSC a évolué, s'est structuré et quelques projets connexes comme Hackropole ont vu le jour. En plus d'être apprécié par les joueurs de CTFs, le FCSC permet également de répondre à des objectifs de l'ANSSI et mobilise chaque année une trentaine de personnes. Dans cet article, nous revenons sur cet événement, la sélection de l'équipe nationale et sur la mise en place d'archives des épreuves passées.

1 Introduction

L'Agence Nationale de la Sécurité des Systèmes d'Information (ANSSI) est l'autorité nationale française en matière de cybersécurité. Sa mission est de comprendre, prévenir et répondre aux risques liés à la sécurité informatique. Depuis 2019, l'ANSSI organise au printemps une compétition d'une dizaine de jours : le France Cybersecurity Challenge (FCSC). Cet événement est un Capture-The-Flag (CTF) en ligne, gratuit et ouvert à toutes et à tous. Un CTF est un type d'exercice de cybersécurité scénarisé sous forme de jeu où les participants sont invités à résoudre des épreuves techniques de différents domaines (e.g., cryptographie, sécurité web, exploitation binaire, etc.) pour trouver des *flags*. Un flag est généralement une simple chaîne de caractères [1] dont la connaissance atteste la réussite de l'épreuve associée.

Les motivations de l'ANSSI pour l'organisation du FCSC sont nombreuses. Il s'agit à la fois de faire découvrir les différents domaines de la sécurité informatique à un public large, expert ou non, mais également de proposer un environnement sur une courte période où les participants peuvent se mesurer individuellement les uns aux autres dans un esprit de compétition. Par ailleurs, au-delà de l'aspect purement scientifique et technique, le FCSC s'inscrit dans un écosystème plus large dans lequel d'autres établissements publics proposent également des CTF tels que

[1] Souvent de la forme reconnaissable « `FCSC{23824e6c87abcc709f6ccf85274fe968}` ».

DG'hAck, TRACS et le 404 CTF. Ces compétitions ont souvent un objectif de recrutement affiché à des degrés variables, mais cela n'a pour le moment pas été le but principal du FCSC.

L'aspect compétitif du FCSC se traduit pendant l'événement par la présence d'un classement de tous les participants. Ce classement permet ensuite à l'ANSSI de sélectionner une équipe de 10 personnes âgées de moins de 25 ans pour représenter le pays à la prochaine édition de l'European Cybersecurity Challenge (ECSC). En effet, depuis 2014 l'Agence de l'Union européenne pour la cybersécurité (ENISA), l'organisme européen chargé de la cybersécurité au sein de l'Union Européenne coordonne l'ECSC. Il s'agit d'une compétition à destination des jeunes de moins de 25 ans où des équipes de citoyens des pays membres UE/AELE s'affrontent en présentiel pendant deux jours sur des épreuves conçues pour l'occasion.

Au-delà de l'aspect compétitif, le FCSC tente également de répondre à un objectif pédagogique, en proposant des épreuves introductives qui nécessitent peu de compétences ou d'outillage pour être résolues. Le but est alors de faire découvrir les missions de l'ANSSI et les différents domaines de la sécurité informatique aux participants. Dans le prolongement de cette volonté, l'ANSSI a publié fin 2023 la majorité des épreuves proposées au FCSC depuis 2019 sur un site dédié appelé Hackropole.[2] Ce site permanent rassemble plus de 400 épreuves classées par thématiques et difficultés et propose aux visiteurs une manière simple de les rejouer, mais également de lire des solutions quand celles-ci sont disponibles.

Structure de l'article. Dans ce document, nous abordons tous les projets cités précédemment et leurs évolutions depuis 2019. Dans la section 2, nous décrivons le FCSC, son mode de fonctionnement, l'infrastructure conçue spécifiquement pour ce CTF puis nous présentons les liens avec l'ECSC dans la section 3, et notamment la constitution et l'organisation de la *Team France*. Enfin, dans la section 4, nous détaillons le développement d'Hackropole et donnons quelques statistiques d'utilisation depuis son lancement.

2 France Cybersecurity Challenge (FCSC)

Le FCSC est une compétition de type « Capture-The-Flag » (CTF), c'est-à-dire qu'il faut collecter des *flags* pour gagner des points. Il existe principalement deux types de CTF selon le mode de jeu choisi : les CTF dits « Attaque/Défense » où des équipes incarnent à la fois des attaquants

[2] `https://hackropole.fr`

Tableau 1. Les différentes phases de l'organisation du FCSC et de l'ECSC sur une année civile.

Période	**Phase**
Janvier - Début avril	Préparation des épreuves et de l'infrastructure
Fin avril	FCSC
Mai - Juin	Entretiens puis sélection de la *Team France*
Juin - Septembre	Entraînement de la *Team France*
Octobre	ECSC
Novembre - Décembre	Préparation des épreuves

et des défenseurs (*red/blue teams*), et les CTF « Jeopardy », où une série d'épreuves indépendantes sont à résoudre. Chacune se termine par une chaîne de caractères permettant d'attester de la réussite : le *flag* (un drapeau). Le FCSC suit ce mode « Jeopardy », qui est moins complexe à mettre en œuvre au niveau de l'infrastructure et qui permet également de faire un classement individuel plutôt qu'en équipe.

2.1 Organisation du FCSC

En interne, l'équipe organisatrice du FCSC travaille sur ce sujet tout au long de l'année, que ce soit pour préparer le FCSC ou en prévision de l'ECSC (tableau 1).

Comme évoqué précédemment, l'ENISA coordonne chaque année l'organisation de l'ECSC, mais laisse la liberté aux pays participants d'utiliser la méthode de sélection nationale qu'ils souhaitent, tant que les critères d'âge sont respectés. On peut citer plusieurs modèles de sélection :

- une sélection à travers l'organisation d'une compétition en ligne ;
- une sélection en plusieurs rondes, avec une phase de compétition en ligne puis une ou plusieurs phases hors ligne ;
- une sélection se reposant sur des compétitions externes ;
- la sélection des participants d'une équipe de CTF nationale déjà constituée.

Lors de la création du FCSC, une seconde phase de compétition avait lieu, hors ligne en 2019, puis en ligne en 2020 et 2021, où un second jeu d'épreuves était conçu pour les quelques dizaines de finalistes. Ce mode de fonctionnement a été délaissé, pour simplifier la logistique et avoir plus de liberté dans la composition de l'équipe. Aujourd'hui, à l'issue du classement, les meilleurs joueuses et joueurs du FCSC doivent soumettre leurs solutions écrites à deux épreuves de leur choix, et ont l'occasion de passer un entretien pour être sélectionnés dans l'équipe de France pour l'ECSC, appelée *Team France*. Une trentaine d'entretiens sont

organisés en ligne par les agents de l'ANSSI pour identifier plus finement les compétences des participants et ainsi pouvoir conjuguer les profils pour aboutir à une équipe équilibrée.

2.2 Contenu technique du FCSC

Catégories. Toutes les catégories classiques de CTF sont représentées dans les épreuves du FCSC :

— cryptographie (*crypto*),
— rétro-ingénierie (*reverse*),
— exploitation binaire (*pwn*),
— sécurité des applications web (*web*),
— investigation numérique (*forensics*).

À cela s'ajoutent quelques épreuves hors catégorie dites « *misc* » qui peuvent porter sur du réseau, du système, de la programmation algorithmique, etc. Par ailleurs, en plus de ces catégories classiques, le FCSC inclut généralement des épreuves plus originales liées à la sécurité matérielle provenant directement des domaines d'expertise des concepteurs. On peut par exemple citer l'analyse de signaux radio, l'étude de l'électronique embarquée, l'exploitation d'attaques par faute ou par canaux auxiliaires.

Chaque année, une dizaine d'épreuves sont proposées dans chacune de ces catégories, selon le temps des concepteurs et leur inventivité. De manière générale, toutes les épreuves sont conçues afin que les participants découvrent, apprennent ou manipulent un concept technique. Les points techniques abordés n'ont pas vocation à tous être d'un niveau d'expertise très élevé : certaines épreuves doivent être complexes pour permettre de différencier les participants, et notamment ceux qui tentent la sélection pour la *Team France*, mais pour la majorité des autres, l'intérêt et si possible l'originalité technique, sont les premiers critères recherchés.

Au-delà de l'aspect compétition, le FCSC a l'ambition de répondre à un objectif pédagogique, en proposant des épreuves introductives qui nécessitent peu de compétences pour être résolues. Durant l'événement, ces épreuves sont clairement identifiées et le but est de faire découvrir des domaines de la sécurité informatique et l'ANSSI aux participants, sans impact réel sur le classement général. Dans l'enseignement supérieur, certaines de ces épreuves ont par exemple été réutilisées par des enseignants pour illustrer certains concepts auprès de leurs étudiants, et cette volonté pédagogique a été fortement renforcée par la publication de Hackropole (voir section 4).

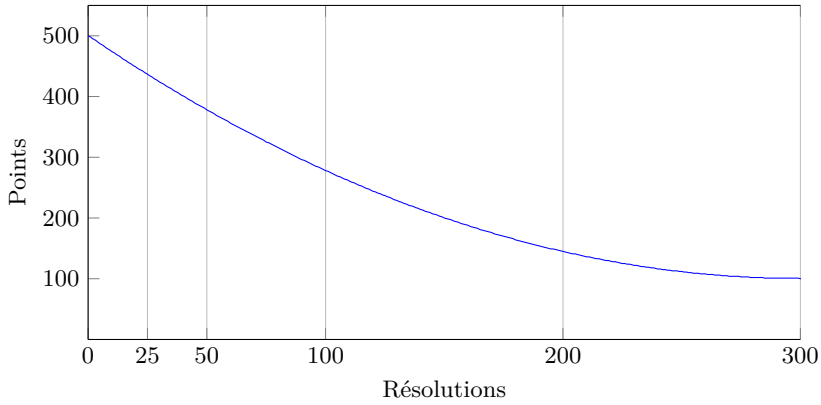

Fig. 1. Fonction pour le score dynamique utilisée au FCSC : le nombre de points que rapporte une épreuve décroît avec le nombre de résolutions.

Scores. Plusieurs approches existent pour répartir la quantité de points que la résolution d'une épreuve apporte, il est possible de jouer sur deux facteurs :

— le score *statique* : la valeur de base en points de chaque épreuve, qui peut être plus grande pour une épreuve plus difficile ;
— le score *dynamique* : l'évolution du nombre de points que vaut une épreuve en fonction du nombre de résolutions.

Au FCSC, à quelques rares exceptions près, les épreuves démarrent toutes avec un score de 500 points, quelles que soient leurs difficultés,[3] puis leur score décroît en suivant une fonction préétablie qui dépend du nombre de résolutions (voir figure 1). Cette manière de compter les points est très répandue dans les CTF, seule la fonction de décroissance peut légèrement changer. Au FCSC, la fonction f qui donne le nombre de points en fonction du nombre de résolutions x est la suivante :

$$f(x) = \begin{cases} \left\lceil s_\infty + (s_0 - s_\infty) \cdot \left(\frac{x-d}{d}\right)^2 \right\rceil & \text{if} \quad x \leq d, \\ s_\infty & \text{if} \quad x \geq d, \end{cases}$$

avec $s_0 = 500$ le score initial, et $s_\infty = 100$ le score minimal atteint après $d = 300$ résolutions.

L'intérêt principal est de dissocier l'estimation de la difficulté d'une épreuve par son concepteur de sa difficulté réelle perçue par les joueurs.

[3] Sauf pour les épreuves d'introduction qui possèdent un petit nombre de points pour ne pas trop affecter la compétition.

En effet, la connaissance des détails de l'épreuve ainsi que de sa solution biaise nécessairement l'évaluation de la difficulté : il est ainsi préférable de se fier aux nombres de résolutions pour constater ce qui est facile ou difficile pour les participants.

De plus, pour les thématiques moins prisées, certaines épreuves faciles ont peu de résolutions même après plusieurs jours et valent mécaniquement beaucoup de points : pour les joueurs qui cherchent une qualification, ces épreuves peuvent être des objectifs de choix pour être bien positionnés dans le classement et cela les force donc à s'aventurer dans des catégories dont ils ne sont pas spécialistes.

Enfin, l'ordre de résolution n'a pas d'influence sur le score : la première personne à résoudre une épreuve valant initialement 500 points obtient pour cette épreuve le même nombre de points que toutes les personnes qui valident cette épreuve par la suite. Bien que ce décompte puisse paraître légèrement déroutant étant donné que les scores peuvent augmenter, mais également diminuer dans le temps, il permet néanmoins à chacun d'organiser son temps comme il l'entend, sans dépendre des autres, du fuseau horaire, de son travail, etc.

2.3 Organisation technique

D'un point de vue technique, l'événement regroupe les éléments suivants :

- Les épreuves que les joueurs devront résoudre. Certaines doivent être résolues en ligne, d'autres doivent être résolues hors ligne après avoir téléchargé les fichiers associés.
- La plate-forme de la compétition, qui est une version modifiée de CTFd [21] afin de lister les épreuves, soumettre les *flags* et afficher le classement en temps réel.
- L'infrastructure qui héberge les épreuves, la plate-forme CTFd, la remontée des journaux, etc.

Avant l'événement

Préparation des épreuves. Chaque année, plusieurs dizaines d'épreuves sont conçues pour le FCSC. Par exemple, l'édition 2023 regroupait par exemple 73 épreuves, pour 21 concepteurs, tandis que l'édition 2024 rassemblait 97 épreuves pour 25 concepteurs. Les niveaux de difficulté identifiés sont « introduction », « facile », « moyen » et « difficile ». Cette étape de conception des épreuves est assez variable selon les années et dépend

beaucoup de la motivation des personnes impliquées, de leur disponibilité, de l'actualité, etc. Il s'agit d'une phase transverse avec beaucoup d'interactions entre des personnes de domaines potentiellement très différents, qui apporte généralement une expérience humaine et technique très enrichissante. Nous exposons ci-après quelques généralités sur les étapes de conception et de tests, mais il est difficile d'être exhaustif et faute de place, nous ne rentrerons pas dans tous les détails.

Les épreuves d'introduction ne doivent nécessiter que très peu de bagage technique, voire peuvent être résolues à la main. Les épreuves faciles vont en général se baser sur des sujets et/ou vulnérabilités bien connus et documentés. À l'inverse, les épreuves difficiles vont nécessiter la compréhension précise d'un problème ou d'un sujet difficile et leur résolution nécessitera un ou plusieurs scripts à développer soi-même.

Pour garantir à la fois la qualité et le bon fonctionnement des épreuves, des tests sont faits sur chacune des épreuves. À la conception, tous les auteurs doivent fournir le code de l'épreuve, mais également une description interne, une méthode de résolution et un script de résolution automatique. Pour une épreuve en ligne, les conditions particulières de déploiement doivent également être précisées et une recette de déploiement Docker [13] doit être fournie.

Le test des épreuves en tant que tel se fait en boîte noire dans la mesure du possible, c'est-à-dire que le testeur n'utilise que la description publique de l'épreuve pour sa résolution et n'en connaît pas au préalable le sujet. Il doit alors refaire toutes les étapes de résolution. En pratique, une partie des épreuves est testée en boîte grise par manque de temps, notamment pour les épreuves de difficulté facile, sur des sujets bien connus. Cela permet de vérifier tous les aspects de l'épreuve :

— que l'épreuve peut être résolue avec les éléments fournis ;
— que la difficulté estimée est adaptée à l'épreuve ;
— qu'il n'y a pas de contournement, rendant l'épreuve moins intéressante.

Après plusieurs années sur ce modèle, les tests en boîte noire ont confirmé leur intérêt : il y a généralement peu de problèmes liés aux épreuves pendant le FCSC. Certaines épreuves peuvent être modifiées et améliorées avant leur publication grâce au test, si par exemple le processus de résolution attendu est estimé trop alambiqué. Il y a toujours quelques modifications d'épreuves qui sont faîtes pendant le FCSC, mais il s'agit généralement de clarifications de consignes (e.g., sur le format du *flag*) ou d'erreurs mineures qui étaient passées inaperçues. En revanche, il est arrivé que certaines modifications majeures doivent être apportées en

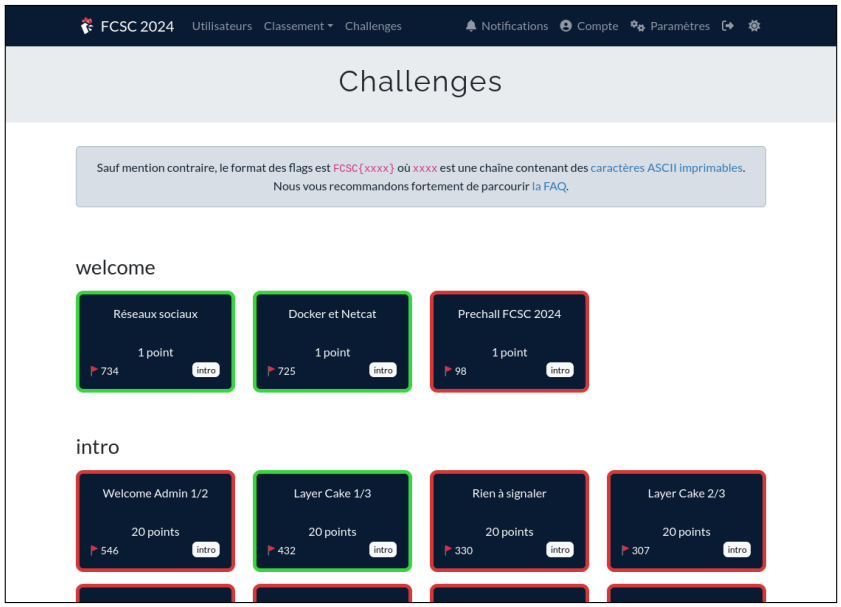

Fig. 2. Page listant les épreuves du FCSC 2024.

urgence, mais cela reste minoritaire sur les quelques 400 épreuves publiées depuis 2019.

Préparation de la plate-forme CTFd. La plate-forme utilisée pour la compétition est basée sur le logiciel libre CTFd [21], un logiciel bien connu des habitués de CTF. Cette plate-forme web permet à tous de s'enregistrer, de visualiser la liste des épreuves (figure 2) ainsi que leurs descriptions et de soumettre les *flags* trouvés pour résoudre l'épreuve. En plus des modifications graphiques, quelques extensions ont été développées et sont maintenues pour le FCSC : la matrice de résolution, le top des joueurs par catégorie et l'accès à une épreuve de teasing. Nous envoyons régulièrement des correctifs vers le code source public de CTFd.

Préparation de l'infrastructure. L'infrastructure est mise en place en parallèle du développement des épreuves par une équipe dédiée. L'ensemble des services nécessaires au FCSC est installé sur des serveurs dédiés hébergés dans des centres de données en France. Ces services sont redondés pour réaliser de la haute disponibilité et ainsi augmenter la résilience.

Nous reviendrons plus précisément sur l'infrastructure mise en place dans la suite de l'article.

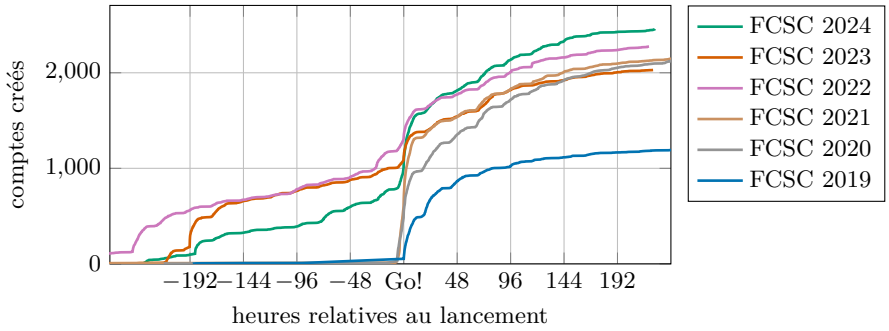

Fig. 3. Évolution du nombre de comptes créés pour chaque édition du FCSC.

Pendant l'événement

Communication. Pendant l'événement, la communication et les annonces aux joueurs se font via un serveur Discord : la principale tâche de l'organisation est donc de répondre aux questions des joueurs et de modérer le serveur Discord. En cas de bug sur une épreuve, des correctifs sont également développés et immédiatement déployés, et les participants sont tenus informés via Discord et un système de notifications interne à CTFd.

Infrastructure. Côté infrastructure, il faut veiller au bon fonctionnement des épreuves, et à ce que toutes les instances répondent et se comportent comme attendu. La charge des serveurs est particulièrement surveillée suite aux attaques par Déni de Service Distribué (DDoS) du FCSC 2021.

Afin de diminuer la vague de création des comptes lors de l'ouverture du FCSC, une épreuve en plusieurs étapes est proposée aux participants une semaine avant, pour un point symbolique. Cela a pour effet de lisser la charge à l'ouverture (figure 3, figure 4).

Après l'événement

Dès la fin de la compétition, les participants échangent leurs idées et solutions sur le serveur Discord. Les plus motivés rédigent des documents retraçant la réflexion qui leur a permis de résoudre une épreuve, aidant ainsi les autres participants à progresser. La fin de la compétition est aussi l'occasion pour les participants de remercier les créateurs d'épreuves et de manifester leur appréciation pour certaines épreuves. Un sondage est généralement proposé aux participants afin de remonter tout type de remarques, critiques et/ou remerciements à l'équipe organisatrice.

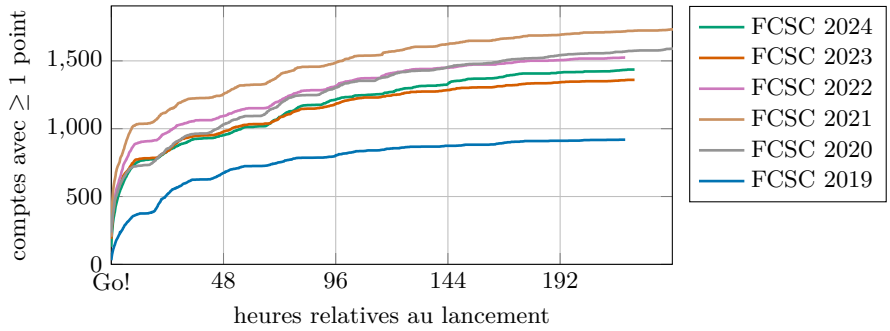

Fig. 4. Évolution du nombre de comptes ayant soumis au moins un *flag* pour chaque édition du FCSC.

2.4 L'infrastructure technique

L'infrastructure technique du FCSC est développée pour répondre aux contraintes propres d'hébergement d'un CTF. Les choix techniques adoptés dans ce cadre sont le fruit de plusieurs années d'apprentissage et d'itérations suite aux précédentes éditions de la compétition.

La construction d'une telle infrastructure est d'intérêt pour les équipes de l'ANSSI : les travaux menés dans le cadre du FCSC renforcent la cohésion d'un large panel de métiers. Ce projet est aussi l'occasion de relever des défis techniques variés en éprouvant de nouvelles technologies ou projets personnels directement en production.

La suite de l'article sera l'occasion de comprendre quelles spécificités revêt l'infrastructure du FCSC et comment elles ont conditionné les choix techniques d'architecture. Nous expliquerons également comment ces choix ont évolué d'année en année ainsi que les évolutions envisagées pour les prochaines éditions.

Défis techniques et particularités

Un challenge pour les administrateurs système. Du point de vue de l'équipe d'administrateurs système, le FCSC est un cas rare : c'est une infrastructure qui vise à héberger des applications volontairement vulnérables, exposées sur Internet. Ces applications vont d'ailleurs, à coup sûr, se faire attaquer. Les attaquants sont même connus et attendus. Ces caractéristiques rendent l'exercice stimulant pour tout administrateur système, d'autant qu'il se déroule sur un temps limité : environ six mois, de la préparation au décommissionnement.

Pour les membres de l'équipe d'infrastructure, le FCSC est aussi vu comme un challenge : les joueurs y participant sont d'un bon, voire très bon, niveau technique. Les failles qui pourraient leur permettre de se latéraliser hors des frontières prévues dans le cadre des épreuves seront scrutées de près. Heureusement, la très grande majorité des participants est bienveillante et ferait part des vulnérabilités (non volontairement introduites) identifiées.

Les contraintes techniques et organisationnelles. Pour mieux comprendre les choix d'architecture qui seront décrits plus tard, il est nécessaire de présenter les contraintes techniques et organisationnelles qui les sous-tendent.

La première contrainte est humaine : l'infrastructure est gérée depuis plusieurs années par une petite équipe, entre une et quatre personnes, majoritairement sur temps personnel. Cela va de pair avec la durée de vie des systèmes mis en place : ils sont éphémères et ne durent que le temps de la préparation et de la compétition. Moins de six mois se déroulent entre le premier serveur commandé et le dernier serveur rendu à l'hébergeur, dont environ trois semaines où les épreuves sont ouvertes au public : 10 jours de compétition, puis 10 jours où les épreuves restent ouvertes après la fin de la compétition. Cette caractéristique est aussi un avantage : le coût de maintenance est relativement faible, au détriment du coût de construction qui est fixe et se répète tous les ans.

Malgré une automatisation croissante des tâches d'installation des serveurs, ce coût initial se renouvelle chaque année. Des réflexions sont en cours pour garder une partie des serveurs toute l'année. Cela permettrait à la fois de servir de laboratoire de test pour les idées des administrateurs, même hors période de préparation, et également comme environnement de test en conditions réelles pour les concepteurs d'épreuves.

Un autre point central dans la réflexion sur les architectures mises en place est le coût financier de ce projet. Le but est de limiter autant que possible la dépense publique engendrée par le FCSC, tout en proposant un système fiable, redondé et robuste. Jusqu'ici, le coût total de l'infrastructure est toujours resté en dessous de 7500 euros (en moyenne 5000 euros environ), toutes taxes comprises.

Les solutions techniques utilisées doivent également être agnostiques de l'hébergeur. Le but est de limiter au maximum l'adhérence technique à un fournisseur. La disponibilité des épreuves et des services associés doit également être assurée. La plate-forme doit être redondée et facile

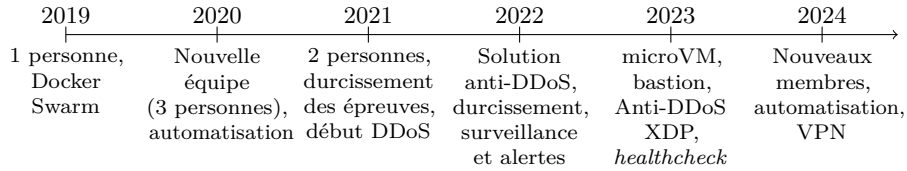

Fig. 5. Innovations dans l'infrastructure au fil des années.

à réparer en cas de problème technique (panne matérielle d'un serveur, coupure de courant dans un centre de données, etc.).

En prévision de l'évolution du FCSC (nombre de joueurs, ouverture à un public plus large, etc.), la plate-forme doit pouvoir passer à l'échelle de façon linéaire : tant pour les coûts financiers que pour la taille de l'équipe d'administrateurs.

Les risques. Au-delà des contraintes décrites plus haut, plusieurs risques doivent être adressés. Ils sont d'ailleurs listés dans une analyse de risque faite à chaque changement majeur d'architecture. Le premier risque est classique : la compromission de l'infrastructure au-delà des challenges à exploiter. Ce risque comprend la latéralisation des joueurs entre les épreuves afin d'exfiltrer les *flags* sans résoudre les épreuves.

Un autre risque important est l'indisponibilité de la plate-forme qui reviendrait, dans le pire des cas, à annuler une édition du FCSC. Ce risque s'est principalement concrétisé ces dernières années sous la forme d'attaques DDoS que nous abordons dans le reste de l'article.

L'ensemble de ces risques, associés aux contraintes détaillées plus haut, ont été pris en compte lors des réflexions sur l'architecture à déployer.

Choix d'architecture. Les contraintes et risques détaillés nous permettent désormais de mieux expliquer les choix faits pour l'infrastructure de FCSC. Nous commençons ces explications avec un rapide aperçu de l'évolution des choix techniques des dernières années.

Évolution au fil des éditions. La compétition est utilisée par les équipes d'infrastructure comme un laboratoire pour tester des idées en production. Cette infrastructure a beaucoup évolué au fil des années (voir figure 5).

Cet article est aussi l'occasion de remercier toutes les personnes ayant contribué à la conception et à la mise en place des briques d'infrastructure qui permettent au FCSC d'exister.

La première itération de l'architecture du FCSC consistait en trois serveurs physiques exécutant un cluster Docker Swarm [14] regroupant tous les services d'infrastructure :

— Le serveur Gitea [4] hébergeant les épreuves ;
— Le serveur Drone CI [15] hébergeant la chaîne d'intégration continue ;
— Un registre d'images Docker privé ;
— La plate-forme CTFd décrite précédemment ;
— Les conteneurs Docker hébergeant les épreuves ;
— Un *reverse proxy* Traefik [19] exposant les services sur Internet ;
— Un cluster GlusterFS [26] permettant de partager les fichiers entre les nœuds du cluster Docker Swarm.

Cette solution disposait de plusieurs fonctionnalités intéressantes de gestion automatique des ressources : la mise à l'échelle des épreuves, le redéploiement en cas de panne. Ces fonctionnalités avaient cependant un coût caché lié à leur complexité : les services étaient tous hébergés au même endroit, donc fortement interdépendants du fait de leur fonctionnement en cluster. De plus, les interconnexions fortes entre les différentes briques d'infrastructure facilitaient la latéralisation d'un attaquant qui aurait pris le contrôle d'une partie des ressources.

Ces coûts peuvent être assumés par une équipe d'administrateurs dédiés qui traitent ces sujets sur le temps long. Ce n'est pas le cas du FCSC : les administrateurs participent à ce projet sur du temps libre et sur une période restreinte. Il était donc nécessaire de passer à un autre paradigme.

Une infrastructure simple, robuste et résiliente. Les premières années du FCSC nous ont appris une leçon claire : il est préférable de troquer des fonctionnalités d'orchestration automatique au profit d'une robustesse accrue. La complexité alors supprimée renforce la sécurité et la résilience de la plate-forme. Cette approche va en réalité à rebours des tendances récentes : le FCSC n'utilise pas de cluster Kubernetes, aucun système d'orchestration mettant en œuvre des algorithmes de consensus complexes. Le but principal est de réduire au maximum les dépendances entre les services déployés.

Cette approche est également justifiée par le nombre limité d'administrateurs disponibles pour maintenir l'infrastructure. Tous les services qui étaient jusqu'alors hébergés sur un même cluster lors des premières éditions du FCSC ont été isolés et hébergés de façon statique sur des machines complètement indépendantes les unes des autres. Chaque serveur

doit être le plus ignorant possible du reste de l'infrastructure, pour une raison simple : faciliter la résilience et accroître la sécurité de l'ensemble. Un exemple très concret peut être utilisé : la compromission d'un des serveurs hébergeant les épreuves ne doit pas permettre de se latéraliser vers les autres.

Cette approche a été intégrée au déploiement des services au sein des serveurs. À titre d'exemple, les règles de pare-feu ne sont pas déléguées à Docker sur les serveurs. Elles sont générées à partir de gabarits développés à la main en fonction des services déployés et modifiés seulement quand nécessaire par les administrateurs via des *playbooks* Ansible [25]. Le but est simple : limiter les choix qui sont faits par les applicatifs à la place des administrateurs afin de mieux maîtriser les briques de sécurité essentielles de l'infrastructure.

Suppression du cluster Swarm et isolation des applicatifs. Les premiers changements majeurs ont donc consisté à supprimer le cluster Docker Swarm. Tous les applicatifs (chaîne de déploiement, serveur git, CTFd, etc.) ont été migrés vers des serveurs physiques ou virtualisés dédiés à chaque ressource. Les épreuves ne sont plus déployées sur un cluster : elles sont directement mises en production en contactant la socket Docker depuis la chaîne de déploiement continue sur chaque serveur hébergeant des épreuves (avec un accès restreint à l'IP du serveur de déploiement et une authentification TLS mutuelle).

Changement de proxy. Le proxy Traefik a été remplacé par des HAProxy pour les épreuves et des serveurs Nginx pour la partie CTFd. L'utilisation de deux technologies différentes vient de besoins spécifiques non supportés par HAProxy pour le CTFd : l'utilisation intensive du cache afin de soulager l'applicatif et le fonctionnement du CTFd impliquant des connexions longues pour les notifications asynchrones. Les serveurs HAProxy utilisés pour les épreuves nous permettent d'avoir un contrôle très fin de la configuration de chaque épreuve.

Protection des services pour les organisateurs. Tous les services à destination unique des organisateurs (chaîne de déploiement, serveur git, etc.) sont filtrés par IP. Les organisateurs souhaitant y accéder ont donc deux choix : utiliser le VPN mis à disposition ou donner une adresse IP qui sera autorisée à accéder à l'infrastructure.

Tous les services ouverts aux joueurs pendant la compétition sont soumis aux mêmes règles jusqu'aux dernières minutes avant l'ouverture

du FCSC. Cette mesure permet de réduire drastiquement la surface d'exposition de la plate-forme et de limiter les impacts de potentielles erreurs humaines.

Évolutions dans l'utilisation des conteneurs. Lors des premières années de la compétition, la sécurité des épreuves s'est largement basée sur la conteneurisation. Au-delà des services d'infrastructure qui sont tous déployés dans des conteneurs, l'isolation des épreuves était garantie par ces mêmes mécanismes (principalement les *namespaces*, *cgroups* et *capabilities*).

La préparation de l'infrastructure du FCSC peut être résumée en deux grandes tâches parallèles : la préparation des serveurs et le durcissement des épreuves.

Une épreuve en fin de conception correspond à un fichier `docker-compose.yml` associé à un ou plusieurs `Dockerfile`. L'étape de durcissement du challenge consiste à sécuriser l'environnement de l'épreuve. Cela permet à la fois de protéger l'infrastructure, mais aussi de garantir que les joueurs ne se gênent pas mutuellement dans la résolution des épreuves. En effet, les joueurs jouent tous dans les mêmes conteneurs au FCSC, principalement pour des raisons de coût d'infrastructure.

Dans les faits, le durcissement des conteneurs correspond aux grandes étapes suivantes :

- Suppression d'un maximum de *capabilities*, si besoin en modifiant légèrement le code de l'épreuve ;
- Suppression des droits d'élévation de privilège au sein du conteneur (option `no-new-privileges` de Docker) ;
- Passage du conteneur en lecture seule ;
- Ajout d'un profil AppArmor spécifique à l'épreuve (sur certaines épreuves seulement) ;
- Ajout d'un profile seccomp spécifique à l'épreuve (assez rarement utilisé).

Ces modifications sont généralement faites par l'équipe d'infrastructure. Elles sont cependant de plus en plus intégrées aux phases de conception des épreuves.

Ces deux dernières années, la conteneurisation des challenges a changé de statut. La fonction principale des conteneurs est passée de frontière de sécurité à un mécanisme facilitant le déploiement des épreuves. Cette frontière de sécurité permettait jusqu'ici de limiter les risques que le joueur sorte de l'environnement de l'épreuve, voire se latéralise entre celles-ci. Ce rôle est désormais assuré par la virtualisation. Chaque épreuve s'exécute

dans un environnement virtualisé minimaliste qui sera abordé plus en détail dans la suite de l'article.

Schéma d'infrastructure. Le schéma d'infrastructure figure 6 donne une vue d'ensemble de l'infrastructure.

Les étapes de déploiement d'un challenge. Au FCSC, les épreuves sont déployées par une chaîne de déploiement automatique. Ce service permet notamment de déployer les épreuves automatiquement lorsqu'une modification intervient sur les dépôts git hébergeant les épreuves. Les grandes étapes de déploiement d'une épreuve dans la chaîne de déploiement continu sont les suivantes :

— Clone du dépôt git ;
— Lint des fichiers `Dockerfile` via hadolint [10] ;
— *Build* des images Docker et ajout dans le *registry* privé Docker ;
— Build du disque de la microVM ;
— Déploiement sur les managers de microVM.

Les étapes de préparation du disque et de déploiement des VM sont réalisées par des plugins développés spécifiquement par les administrateurs pour les usages spécifiques du FCSC. Le plugin préparant les disques crée un disque au format `ext4` par sous-dossier du répertoire `vm/disks` du dépôt. Les disques sont ensuite mis à disposition des VM pour qu'ils puissent être montés au moment du démarrage de celles-ci (ces étapes sont détaillées par la suite).

Les plugins de déploiement des VM réalisent les étapes suivantes pour chaque serveur hébergeant des épreuves (`blackbuck-1`, `blackbuck-2`, `metal-1`, `metal-2`, voir figure 6) :

— Passage du *backend* HAProxy en mode *drain*. Cela signifie que les joueurs seront dirigés vers une autre instance du challenge. Si le déploiement se fait sur le serveur `metal-1`, les joueurs seront dirigés vers un des trois autres serveurs. Cette étape est réalisée par l'intermédiaire d'une interface de programmation applicative (API) développée spécifiquement pour le FCSC et basée sur les fonctionnalités intégrées à HAProxy ;
— Suppression de la VM correspondant au challenge si elle existe ;
— Suppression des interfaces réseau correspondant au challenge si elles existent ;
— Création des interfaces réseau du challenge ;
— Création de la VM ;
— Passage du *backend* HAProxy en mode *ready*. Les joueurs peuvent à nouveau être redirigés vers cette instance de l'épreuve.

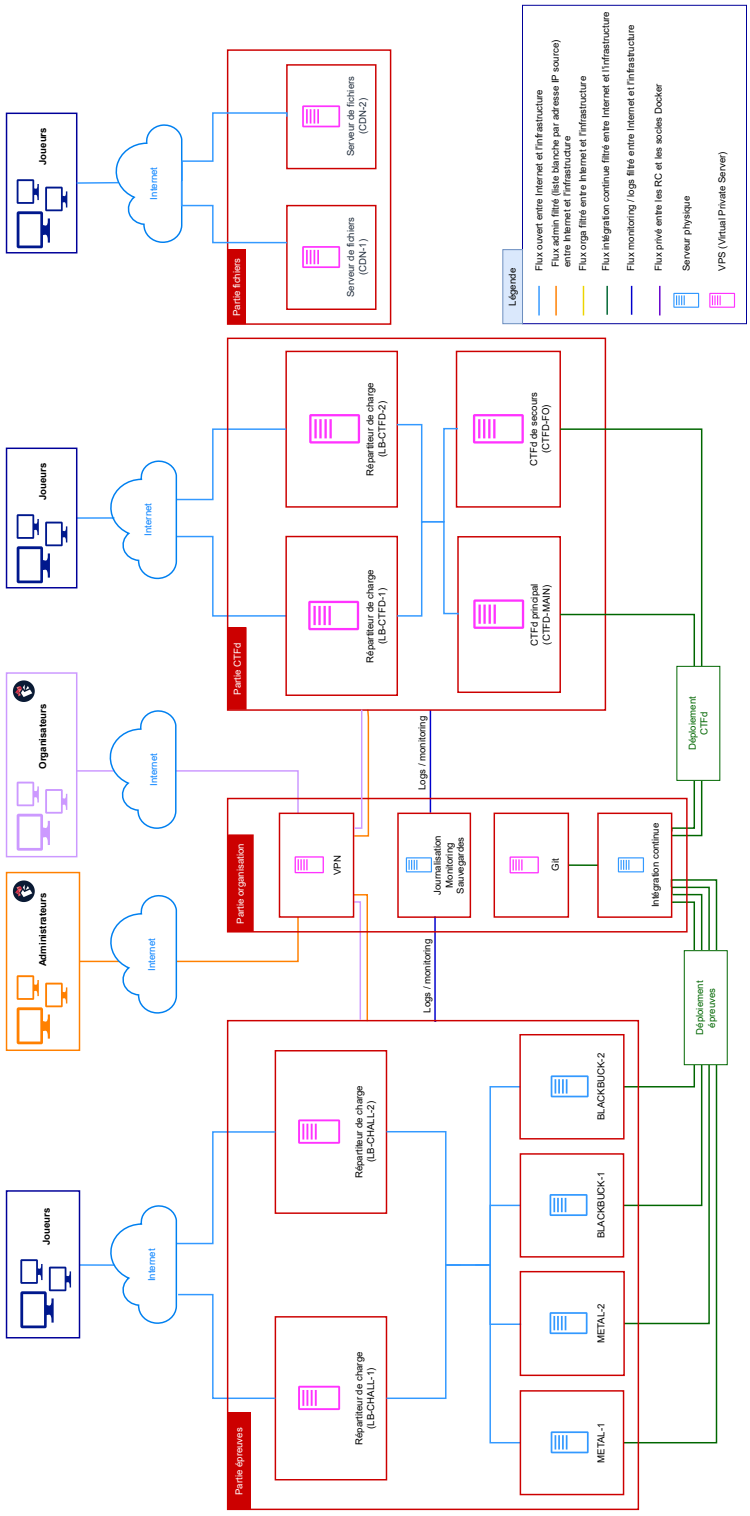

Fig. 6. Schéma de l'infrastructure du FCSC 2024.

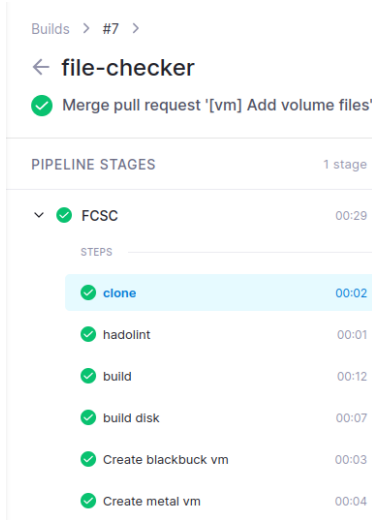

Fig. 7. Étapes de déploiement d'une épreuve en production.

Ces étapes sont réalisées de manière séquentielle pour chaque serveur hébergeant des épreuves. Cela permet de limiter la période d'indisponibilité des services. Ainsi, le redéploiement d'une épreuve est généralement transparent pour les joueurs. Visuellement, les étapes de déploiement dans la CI sont représentées sur la figure 7. La mise en production des épreuves se fait via des *merge request* sur une branche dédiée à la production et protégée.

Supervision et sauvegardes de l'infrastructure. Tous les journaux des serveurs sont envoyés à un serveur central. Cela inclut les journaux des services d'infrastructure, ceux des VM et ceux des conteneurs s'exécutant dans les VM. Ces données sont mises à disposition d'un nombre limité d'organisateurs afin de pouvoir investiguer d'éventuels problèmes sur les épreuves. Toutes les données les plus sensibles sont également sauvegardées. Cela concerne principalement le CTFd et le serveur git, qui sont sauvegardés toutes les 30 minutes.

De nombreuses métriques sont également envoyées au serveur de supervision. Des *dashboard* Grafana sont mis en place afin de surveiller les métriques et l'état des épreuves. Visuellement, cela permet de se rendre compte très rapidement si les vérifications faites par les proxy sont valides ou non (voir figure 8).

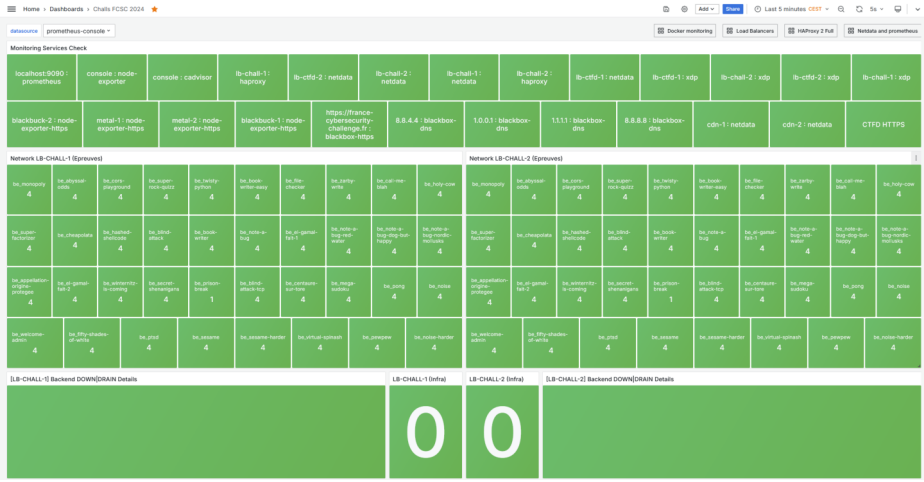

Fig. 8. Supervision des épreuves lors de la compétition.

Accès aux VM des épreuves. Une des problématiques rencontrées lors du FCSC est l'accès aux machines exécutant les épreuves par les organisateurs pendant la compétition. En effet, il est souvent utile d'accéder aux conteneurs des épreuves afin de régler certains problèmes mineurs lors de la compétition.

Lors des premières années du FCSC, ces actions étaient déléguées à l'équipe d'administrateurs, car très peu d'organisateurs avaient accès aux serveurs. Depuis 2023, l'arrivée des VM dans l'infrastructure a permis de mettre en place un nouveau système. Il permet aux organisateurs d'accéder aux machines virtuelles hébergeant les épreuves sans avoir accès aux serveurs physiques les hébergeant.

Pour ce faire, plusieurs composants sont utilisés :

— Une Infrastructure de Gestion de Clés (IGC) en ligne qui gère les clés SSH des organisateurs et celles des VM ;
— Une VM d'administration qui sert de bastion afin d'accéder aux machines virtuelles hébergeant les épreuves ;
— Un conteneur permettant aux organisateurs de générer automatiquement une clé SSH grâce à une authentification SSO avec leur compte Gitea. Le conteneur leur permet ensuite de se connecter à la VM hébergeant un challenge sur le serveur de leur choix (parmi les quatre serveurs de VM décrits sur le schéma d'infrastructure).

Cette nouvelle possibilité, couplée à la chaîne de déploiement continu, permet aux organisateurs d'être complètement autonomes pour la gestion des épreuves. Ils peuvent investiguer le fonctionnement de l'épreuve dans

les machines virtuelles, pousser des modifications s'ils le souhaitent, ce qui redéploiera l'épreuve automatiquement en production.

2.5 Projet *Hodor* : l'anti-DDoS du FCSC

Le FCSC est sujet à des attaques par DDoS depuis 2021. L'infrastructure a évolué depuis afin de contrer ces attaques. En 2021, après plusieurs éditions plutôt apaisées, la disponibilité de l'infrastructure du FCSC est fortement affectée par un DDoS sur la couche applicative. [4] Le service CTFd utilisé comme principale façade web par les joueurs est submergé de requêtes et rendu indisponible durant près de 24 heures. L'architecture monolithique alors en place ne permet pas la mise en œuvre de contre-mesures efficaces. L'organisation du FCSC décide alors d'une suspension temporaire des épreuves afin de permettre aux administrateurs d'isoler physiquement le CTFd, de renforcer sa configuration et d'y adjoindre des mesures de protection. L'ensemble de ces mesures reçoit pour nom de projet « *Hodor* » en référence au personnage du Trône de Fer qui parvient par sa seule force à retenir une porte stratégique face à l'afflux d'une horde de monstres. Ce projet ne vise pas à se substituer aux mesures de protection des hébergeurs qui sont indispensables notamment pour les attaques volumétriques sur les couches 3 et 4. Il constitue néanmoins un complément efficace en second rideau pour faire face à des attaques de moindre ampleur sur la couche applicative qu'elles proviennent des joueurs eux-mêmes dans leur empressement de résoudre les épreuves ou de plateforme d'attaque dédiée.

Première version en 2021. La première version d'*Hodor* permet de revoir complètement la configuration du serveur web. La pile réseau du noyau Linux est configurée pour accepter une très forte activité réseau. Les règles de filtrage `iptables` en charge de l'activité malveillante sont positionnées sur les tables les plus performantes. De nombreux paramètres du serveur d'hébergement web (Nginx) sont modifiés pour accroître sa performance, limiter la consommation des ressources et limiter l'impact des abus. Des mécanismes de *rate-limiting* sont ajoutés pour limiter le nombre de requêtes pouvant être effectuées par pas de temps, URL, adresse IP. Les clients dépassant trop régulièrement les seuils de *rate-limiting* sont dynamiquement ajoutés à `iptables` dans une instance de l'outil libre `fail2ban` [17]. Cette version de *Hodor* permet la remise en ligne des

[4] Plusieurs plates-formes de la communauté CTF ont été ciblées sur la même période (CTFTime, HeroCTF).

épreuves et la mitigation des attaques DDoS qui persisteront malgré tout jusqu'à la fin de l'événement.

L'objectif est atteint, mais plusieurs axes d'amélioration sont identifiés. Lors de la mitigation des attaques DDoS, `fail2ban` accumulait du délai sur la lecture des journaux applicatifs, entraînant une très forte charge CPU et un retard de l'insertion des règles au sein du pare-feu. Au-delà de la solution d'urgence mise en place en 2021, l'équipe souhaite pouvoir disposer d'une solution répondant au cahier des charges suivant :

- Disposer d'une solution d'analyse des journaux applicatifs fonctionnant à forte charge (plusieurs centaines de milliers d'événements par seconde) et permettant l'insertion rapide de règles de filtrage dans le pare-feu.
- Disposer d'une solution pouvant être distribuée sur plusieurs serveurs afin de protéger le CTFd ainsi que les épreuves du FCSC, aussi bien aussi bien sur des protocoles en couche 7 (HTTP, HTTPS) que sur des connexions TCP en couche 4.
- Disposer de mécanismes de notifications (email, Discord) permettant d'alerter les administrateurs de la plate-forme tout en enrichissant les alertes de contexte (utilisateur FCSC, géolocalisation IP, extrait de logs, etc.).
- Disposer de mécanismes de blocages automatisés permanents, temporaires et/ou exponentiels en fonction des comportements observés.
- Disposer d'un outil en ligne de commande permettant d'ajouter manuellement des règles de blocages.

Dès 2021, un prototype est écrit dans le langage Golang. Il intègre une partie de ces fonctionnalités et est mis en production durant les derniers jours de l'événement. Les éditions successives ont permis une amélioration permanente de cet outillage.

Évolutions de 2022 à 2024. En 2022, la configuration du serveur est améliorée et *Hodor* est en capacité d'ingérer 300k req/s avec une charge mémoire et CPU réduite (50Mio en moyenne) tout en répondant à l'intégralité du cahier des charges. Bien qu'il n'y ait pas eu d'activité de DDoS identifiée en 2022, *Hodor* effectue le blocage de 1217 DoS utilisateurs et permet de fluidifier grandement les accès au CTFd et aux épreuves. L'insertion des règles de blocage s'interface directement avec des ipsets gérés dans la table *prerouting raw* d'`iptables`. L'utilisation des ipsets permet l'insertion de plusieurs milliers de règles de filtrage par seconde sans pénaliser les performances de filtrage (accès aux règles en temps constant).

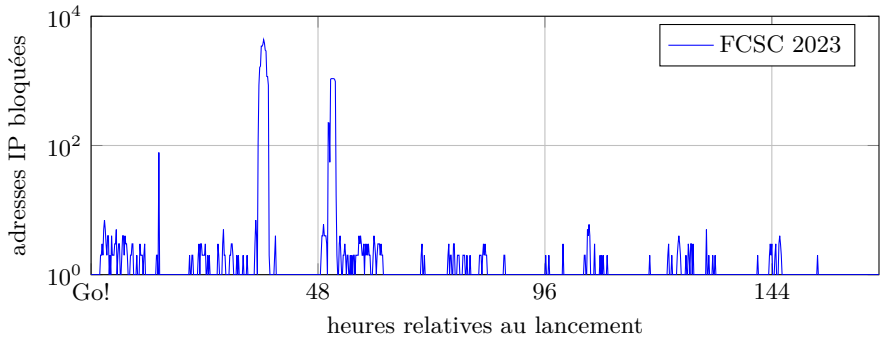

Fig. 9. Blocages réalisés par *Hodor* pendant le FCSC 2023.

En 2023, *Hodor* change de backend de stockage des règles de filtrage en s'interfaçant directement avec un filtre eBPF XDP (bibliothèque `Cilium`) [3]. Le recours à XDP permet à *Hodor* d'effectuer le filtrage des paquets en amont de pile réseau du noyau directement dans le pilote de la carte réseau via une map eBPF partagée entre le noyau et l'espace utilisateur. Cette implémentation confère à *Hodor* un très important gain de performance tout en permettant le découplage complet des règles de filtrage d'*Hodor* de celles gérées par les administrateurs via `iptables`. En 2023, en plus des abus des utilisateurs de la plateforme, trois DDoS mobilisant de quelques dizaines à plusieurs centaines d'adresses IP sont mitigés par Hodor. 7419 adresses IP ont été bloquées sur les dix jours de la compétition. La figure 9 détaille la chronologie de mitigation.

Architecture actuelle de Hodor. *Hodor* est actuellement composé de trois programmes : *Hodor*, *Hodormon* et *Hodorctl* (voir figure 10).

Hodor contrôle les maps eBPF [8] partagées avec le noyau et expose une API REST permettant d'interagir avec ces maps depuis l'espace utilisateur. Le programme eBPF/XDP écrit en langage C est compilé en même temps que le programme Golang et est intégré directement au binaire. Ce programme est attaché directement à l'interface réseau spécifiée dans la configuration de *Hodor* lorsque le programme en espace utilisateur démarre. Dans le cadre du FCSC, les machines virtuelles utilisées pour les répartiteurs de charge permettent le chargement du programme XDP en mode natif, directement au niveau du pilote VirtIO.

Hodor instancie plusieurs maps eBPF qui sont interrogées au passage de chaque paquet par la carte réseau. Deux maps de types `BPF_MAP_TYPE_LPM_TRIE` sont en charge de stocker des blocs réseau (CIDR) IPv4 et IPv6 devant faire l'objet d'un drop (`XDP_DROP`) et deux

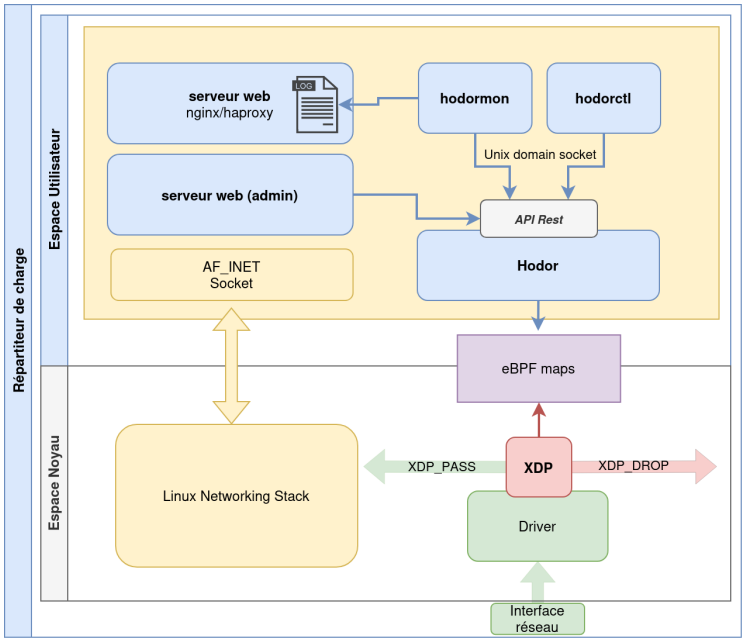

Fig. 10. Architecture logicielle simplifiée de *Hodor*.

autres maps de même type sont en charge de stocker les CIDR IPv4 et IPv6 ne devant faire l'objet d'aucun blocage (`XDP_PASS`). Ces structures définissent respectivement les "cibles" `DROP` et `IGNORE` de Hodor. La cible `IGNORE` est utilisée pour s'assurer qu'un certain nombre de blocs IPs ne feront jamais l'objet d'un blocage (adresses IP de l'infrastructure, des administrateurs...). Deux maps de types `BPF_MAP_TYPE_LRU_HASH` permettent de suivre en temps réel les statistiques réseau par IPv4 et IPv6 avec une remise à zéro chaque seconde par le programme en espace utilisateur. Enfin, une map de type `BPF_MAP_TYPE_PERCPU_ARRAY` permet d'agréger l'ensemble des statistiques du programme afin de pouvoir extraire des métriques d'exploitation en espace utilisateur.

Deux vecteurs permettent d'ajouter ou de supprimer des entrées dans les cibles de filtrage de Hodor : via les sets ou l'API REST (et ses clients : *Hodormon* et *Hodorctl*). Les sets sont des fichiers définissant des listes de CIDR pour les cibles `DROP` ou `IGNORE` de *Hodor*. Ces listes sont chargées au démarrage du programme et peuvent être rechargées de manière atomique via l'API. L'API REST est exposée sur l'hôte via un Unix Socket Domain ainsi qu'en HTTP/s. Un jeton défini dans la configuration de *Hodor*

permet d'autoriser les appels. Cette API permet d'accéder au ressources suivantes :

— Contrôler les entrées de la cible `IGNORE`.
— Contrôler les entrées de la cible `DROP`.
— Contrôler le rechargement des sets.
— Accéder aux statistiques de *Hodor* y compris au format Prometheus.

Toutes les entrées ajoutées peuvent avoir une date d'expiration ainsi qu'un tag permettant de suivre la provenance des enregistrements et de faciliter les traitements par lots (voir 1).

Listing 1: Insertion d'une entrée à la cible `DROP` de Hodor.

```
hodorctl drop add 195.154.171.95/32 -t SSTIC -e 1h
{
  "cidr": "195.154.171.95/32",
  "tag": "SSTIC",
  "expiration": 1711666092
}
```

La configuration permet également de spécifier la synchronisation des enregistrements entre plusieurs nœuds disposant de *Hodor*.

Hodormon est le principal client de *Hodor*. Son rôle est d'effectuer le traitement de tous les journaux applicatifs des serveurs web afin d'y détecter des abus. Il maintient des compteurs de requêtes, d'erreur et de dépassement de seuil et persiste éventuellement la décision de blocage en utilisant l'API REST de *Hodor*. Les heuristiques de blocage sont adaptées chaque année en fonction de nos différents retours d'expérience. Afin de ne pas bloquer les joueurs, le premier blocage est de courte durée. Cette durée devient néanmoins exponentielle en cas de récidive. *Hodormon* est également en charge de notifier les actions de blocage aux organisateurs du FCSC via un connecteur Discord. La notification contient un extrait de l'activité malveillante ainsi que la géolocalisation de l'adresse IP offensive. Ce canal de supervision permet aux modérateurs du FCSC de prévenir les compétiteurs concernés le cas échéant.

Hodorctl est l'interface en ligne de commande permettant de contrôler *Hodor* depuis l'hôte ou à distance.

Listing 2: Interface en ligne de commande de *Hodor*.

```
hodorctl - control hodor fiwerall

Use hodorctl to create XDP filtering rules with ease

Usage:
  hodorctl [flags]
  hodorctl [command]

Available Commands:
  drop        Manage drop entries
  help        Help about any command
  ignore      Manage ignore entries
  sets        Manage Hodor sets
  stats       Display Hodor statistics

Flags:
  -c, --config string   config file (default
  ↪  "/etc/hodor/hodorctl.toml")
  -h, --help            help for hodorctl
  -v, --version         version for hodorctl

Use "hodorctl [command] --help" for more information about a
↪  command.
```

Pour les prochaines éditions, les fonctionnalités suivantes sont en cours de développement :

— Création d'une brique centrale de notification en charge de la corrélation, de la déduplication et de la visualisation de l'activité de *Hodor*.

— Amélioration des mécanismes de synchronisation entre les nœuds *Hodor*.

— Amélioration des heuristiques de détection de comportements suspects.

L'API de *Hodor* en facilitant l'interfaçage avec des programmes existants ou de petits scripts devrait également permettre aux administrateurs système d'étendre simplement les fonctionnalités de filtrage tout en bénéficiant de mécanismes de filtrage performants.

2.6 Les microVM : un nouvel environnement d'exécution des épreuves

Une nouvelle frontière de sécurité. Comme expliqué précédemment dans l'article, l'édition 2023 a vu un changement important dans l'infrastructure :

la conteneurisation n'est plus envisagée comme la principale frontière de sécurité. C'est désormais la virtualisation qui remplit ce rôle. La raison principale vient du fait que le noyau ne peut pas être envisagé comme une frontière de sécurité pour une partie des épreuves du FCSC.

Ce changement est bénéfique pour toutes les épreuves, car il donne plus de contrôle aux concepteurs sur l'environnement. Un noyau spécifique peut désormais être utilisé pour une épreuve en particulier, ce qui n'était pas le cas avec les conteneurs. Cette nouveauté ouvre également un nouveau champ des possibles : certaines épreuves n'étaient avant pas proposées au FCSC car trop compliquées ou risquées à isoler dans un conteneur.

Choix de la technologie utilisée. Le passage des conteneurs aux machines virtuelles posait la question des ressources nécessaires. Le but était de faire cette transition sans grever le budget du FCSC. Les machines virtuelles devaient donc être minimalistes, ce qui contribuerait également à réduire leur surface d'attaque.

Les microVM semblaient idéales pour cet usage : leur usage permet la jonction entre la faible consommation d'un conteneur et les garanties de sécurité d'une machine virtuelle. Il existe plusieurs projets *open source* permettant l'utilisation de microVM. En effet, cette technologie est de plus en plus utilisée par les fournisseurs de solutions *cloud* dans leurs offres de services. Plusieurs acteurs majeurs du marché ont développé leur propre solution, puis l'ont publiée : Google (crosvm [9]), AWS (firecracker [28]). Certains acteurs se sont même regroupés au sein d'un même projet appelé *Cloud Hypervisor* [24]. Parmi eux, on retrouve notamment Microsoft, Intel, ARM et Alibaba. La grande majorité de ces solutions étant développées avec le même langage de programmation (Rust), plusieurs grandes entreprises (Amazon, Google, Intel et Red Hat) se sont accordées pour développer une base de code commune : rust-vmm [27].

Ces différentes solutions offrent des approches différentes du sujet. Certaines sont particulièrement minimalistes et prévues pour héberger des services de *back-end* : c'est par exemple le cas de Firecracker qui est utilisé pour le service *AWS Lambda*. D'autres intègrent plus de fonctionnalités et sont conçues pour héberger des systèmes d'exploitation entiers, potentiellement à destination d'utilisateurs : c'est le cas de CloudHypervisor.

Dans le cadre du FCSC, le besoin est relativement simple : héberger des services légers en minimisant les ressources consommées et en maximisant la sécurité. Le manager de microVM se rapprochant le plus de ce besoin étant Firecracker, nous avons décidé d'utiliser cette solution.

Développement de deux solutions ad hoc. Firecracker se présente sous la forme d'un binaire. Chaque nouvelle microVM est créée en exécutant le binaire et en lui donnant les détails de configuration de la machine par une API ou directement dans un fichier. Son usage est assez simple, mais peu adapté au déploiement de nombreuses machines virtuelles sur plusieurs serveurs.

Certains projets ont pris le parti d'intégrer Firecracker comme un *runtime* Docker. C'est par exemple ce que proposent les *Kata Containers* [5]. Cela permet de profiter de l'écosystème Docker (fichiers *compose*, gestion du réseau, etc.) tout en profitant des gains de la virtualisation. La solution des *Kata Containers* n'a pas été retenue, pour une raison principale : l'architecture utilisée par le projet. Les VM disposent d'un agent qui communique avec la machine hôte au travers d'un *vsock*. Nous préférions éviter de garder cette surface d'attaque dans la VM, qui a déjà été exploitée à plusieurs reprises et constitue un chemin naturel pour s'échapper des machines virtuelles. Le but est d'avoir une VM la plus déconnectée possible de la machine hôte, toujours dans l'objectif de réduire la surface d'attaque exposée depuis l'intérieur de la machine invitée.

Après avoir réalisé des recherches sur l'écosystème des microVM et les solutions proposées en sources ouvertes, nous sommes arrivés à la conclusion qu'il n'existe pas de solution qui couvre exactement nos besoins : permettre de massivement créer des microVM, sans agent dans la machine virtuelle et avec un logiciel simple à déployer. Nous aurions pu faire en sorte d'utiliser des briques existantes, cependant, le FCSC faisant office de laboratoire pour les envies les plus folles des administrateurs, nous avons décidé de développer notre propre solution. Il est important de préciser que le choix de développer une solution ad hoc n'est pas nécessairement raisonnable : nous l'avons fait car c'était possible dans le cadre du FCSC et que le défi technique dans un temps restreint nous motivait.

La finalité de ce développement était de pouvoir créer de multiples microVM par l'intermédiaire d'une API en gérant automatiquement les ressources nécessaires à l'environnement de la VM, sans recourir à un agent dans celle-ci. Une machine virtuelle devait pouvoir être mise en production par une chaîne de déploiement en quelques secondes. Deux versions ont été développées en parallèle par deux administrateurs, sur le modèle du Hackathon longue durée. Le développement a été fait sur du temps libre, sur une période de quelques mois. L'intérêt principal de développer deux solutions, au-delà de la motivation mutuelle, était de pouvoir mettre en production ces preuves de concept en limitant le risque d'avoir fait les mêmes erreurs. La préparation des machines en 2023 permettait de passer

d'un manager de microVM à l'autre en quelques minutes si nécessaire. Ces deux programmes sont nommés `metal` et `blackbuck`. Ils ont été développés dans des langages de programmation différents : Golang pour `metal` et Rust pour `blackbuck`.

Fonctionnalités des managers de microVM. Les programmes développés pour gérer les microVM réalisent deux grandes tâches : créer les ressources nécessaires à la VM sur la machine hôte, puis créer la microVM.

La partie réseau est assez simple. Les microVM disposent d'une interface TAP sur la machine hôte qui peut être ajoutée à un bridge lorsque les microVM doivent disposer d'un réseau commun. L'interface bridge est par exemple utilisée au FCSC pour créer un réseau d'administration auquel appartiennent toutes les VM. Ce réseau sert à exporter les logs et à se connecter en SSH à travers une microVM bastion. Les caractéristiques nécessaires à la configuration d'un réseau sont envoyées à l'API sous forme de fichier JSON. Pour référence, le fichier de configuration du réseau d'administration 3 est inclus dans le papier.

Listing 3: Fichier de configuration du réseau d'administration des microVM en 2024.

```
{
    "id": "chall-adm",
    "config": {
        "type": "bridge",
        "if_name": "br-admin",
        "net": "192.168.98.1/24"
    }
}
```

La partie création des machines virtuelles est celle qui concentre la plus-value du programme. Ses fonctionnalités peuvent être résumées en quelques points :

— Réception de la configuration de la VM par l'API, en JSON ;
— Validation de la configuration : cohérence de la configuration réseau, identifiant unique, référence à un disque système et un noyau valide, etc ;
— Ajout des informations de la machine dans une base de données en mémoire déchargée sur disque ;
— Création des interfaces réseau de la VM sur la machine hôte ;
— Création à la volée d'un initramfs contenant un binaire d'init personnalisé et la configuration JSON de la machine. L'utilisation d'un initramfs plutôt qu'un initrd vient du fait que la VM n'a pas

besoin de supporter un système de fichier spécifique pour charger un initramfs (ce qui n'est pas le cas de l'initrd) ;
— Démarrage de la VM.

Pour référence, le fichier de configuration de la VM de l'épreuve `PTSD` est donné dans le Listing 4.

Listing 4: Fichier de configuration de la VM de l'épreuve `PTSD` en 2024.

```
{
  "id": "ptsd",
  "vm_type": "firecracker",
  "hostname": "ptsd",
  "interfaces": [
    {
      "guest_dev": "eth0",
      "host_dev": "ptsd-p",
      "ip_configs": [
        {
          "host_net": "10.23.2.1/24",
          "guest_net": "10.23.2.2/24"
        }
      ]
    }, {
      "guest_dev": "eth1",
      "host_dev": "ptsd-a",
      "ip_configs": [
        {
          "host_net": "172.16.98.36/24",
          "guest_net": "192.168.98.36/24",
          "network": "chall-adm"
        }
      ]
    }
  ],
  "additional_drives": [{
        "id": "docker",
        "read_only": true,
        "device_path": "/data/disks/ptsd/docker.ext4",
        "mount_point": "/data/bootstrap"
  }],
  "kernel": "hardened/6.7/vmlinux.bin",
  "rootfs": "debian.ext4",
  "rootfs_overlay": false,
  "disk_size": "5 GiB",
  "resources": {
    "ram_size": "1 GiB",
    "vcpu": 1
  },
  "dns": [ "8.8.8.8", "1.1.1.1" ]
}
```

Tous les fichiers utilisés par la VM sont sur une partition dédiée, formatée en BTRFS [18]. Cela permet d'utiliser les fonctionnalités de *copy on write* de ce système de fichier afin d'économiser beaucoup de place

et de lancer les VM plus rapidement. En moyenne, les VM utilisent un disque système de 5 Gio, 1 Gio de RAM et 1 vCPU.

Les avantages de l'utilisation du *copy on write* sont conséquents au niveau des ressources utilisées. À titre d'exemple, environ 40 VM étaient déployées en ligne sur chaque hyperviseur lors de l'édition 2024. L'espace disque total utilisé par toutes les VM sur chaque hyperviseur était de 36 Gio. Sans *copy on write*, cela aurait représenté plus de 200 Gio. En tout, plus de 1000 microVM ont été déployées pour l'édition 2024 du FCSC. Cela inclut les périodes avant, pendant et après la compétition (préparation de l'infrastructure, test des épreuves, etc.)

*Binaire d'*init. L'absence d'agent dans la VM nécessite que celle-ci dispose de toutes les informations nécessaires pour se configurer en autonomie au démarrage. Cette étape est prise en charge par un binaire d'init créé spécifiquement pour cela et exécuté lors du chargement de l'initramfs. Il charge la configuration JSON injectée à la volée lors de la création de l'initramfs, puis configure les briques de base de la VM en fonction du contenu du fichier : configuration DNS, configuration des interfaces réseau, disques montés, routes par défaut, etc.

Déploiement de l'épreuve dans la machine virtuelle. Les épreuves sont toujours déployées sous forme de conteneur Docker dans les machines virtuelles. Sauf nécessité pour la résolution des challenges, les VM n'ont pas accès à Internet. Cette mesure pose une question d'infrastructure : comment charger les images Docker des épreuves sans accès au *registry* Docker privé du FCSC ?

Nous avons décidé d'utiliser les disques additionnels des microVM pour résoudre cette problématique. Une fois les images Docker construites par la chaîne de déploiement, le plugin développé pour créer les disques vient parcourir les fichiers `docker-compose.yml` afin d'avoir la liste des images nécessaires. Ces images sont alors téléchargées depuis le *registry*, compressées, injectées dans le disque `ext4` préparé par la chaîne de déploiement. Ce disque est envoyé aux hôtes des microVM et monté en lecture seule.

Le démarrage de la VM suit alors son cours en passant par plusieurs services `systemd` prévus spécifiquement pour cet usage. Un premier service s'occupe d'amorcer la VM et de placer tous les éléments du disque monté dans la VM au bon endroit :

- Les images Docker compressées présentes dans le disque sont chargées par le démon Docker de la VM ;
- Les certificats SSH servant à la connexion par le bastion sont copiés ;
- Les éventuels profils AppArmor sont chargés dans la VM ;

— Les autres actions spécifiques à chaque épreuve peuvent aussi être faites dans ce service.

Concrètement, ce service utilise un script bash injecté dans le disque monté en lecture seule dans la VM. Ce script est hébergé dans chaque dépôt d'épreuve sur le serveur git et peut donc être modifié facilement. Un deuxième service se charge ensuite de démarrer la *stack* Docker de l'épreuve dans la machine virtuelle.

Préparation du noyau Linux. La compilation des noyaux déployés dans les microVM fait l'objet d'une attention particulière des administrateurs afin de minimiser leur surface d'attaque, réduire leur taille et optimiser le temps de chargement des machines virtuelles.

Un dépôt dédié permet l'édition et la compilation automatisée de deux types de noyau :

— Les noyaux de référence de Firecracker et cloud-hypervisor qui sont utilisés pour les tests d'intégration. Chacun de ces projets fournit les `.config` correspondant à plusieurs versions de noyau. [5,6]

— Le noyau durci de production utilisant la dernière version stable de Linux. Ce noyau peut être décliné pour certaines épreuves nécessitant l'ajout de modules supplémentaires.

Le noyau de production est construit en appliquant plusieurs mesures de durcissement aux configurations de références précitées :

— Application des recommandations de sécurité relatives à un système GNU/Linux [2] de l'ANSSI.

— Application du patch noyau du projet `linux-hardened` [22].

— Application de l'ensemble des recommandations de l'utilitaire d'analyse de configuration `kernel-hardening-checker` [23]. Seules les options `CONFIG_USER_NS` et `CONFIG_BPF_SYSCALL` sont conservées pour permettre l'exécution des conteneurs Docker.

— Retrait de tous les drivers, fonctionnalités et modules non utilisés ou présentant des risques de sécurité importants.

La configuration fine du noyau ainsi que l'étape de déploiement du noyau compilé n'ont pas encore été automatisées. La feuille de route suivante est identifiée pour les prochaines éditions :

— Poursuite des travaux de suppression des fonctionnalités non utilisées du noyau.

[5] `https://github.com/firecracker-microvm/firecracker/blob/main/docs/kernel-policy.md`

[6] `https://github.com/cloud-hypervisor/cloud-hypervisor/tree/main/resources`

— Automatisation de bout en bout de la récupération des dernières versions de noyau, leur compilation et leur déploiement sur les dépôts d'infrastructure.
— Ajout du support de nouvelles architectures matérielles comme ARM.
— Ajout du support de nouveaux systèmes d'exploitation (BSD, Windows, etc.).

Préparation du disque système. La construction du disque système générique des machines virtuelles est réalisée avec l'outil d'automatisation `Packer` [11]. Une image de base Debian stable est chargée avec QEMU et configurée avec `Ansible` [12]. Les recettes suivantes sont appliquées :

— *durcissement système* : supprime les paquets non utilisés, place en liste noire les modules noyaux inutiles ou obsolètes, applique les sysctl de durcissement et plusieurs recommandations du guide Linux [2] de l'ANSSI.
— *ssh* : configure et applique des paramètres de durcissement du serveur OpenSSH.
— *users/pki* : ajoute les utilisateurs système et configure l'IGC du FCSC utilisée par les membres de l'organisation pour avoir accès aux épreuves.
— *chrony* : configure le serveur de temps de la machine virtuelle pour utiliser le module `KVM_PTP` et obtenir une source de temps depuis le système hôte.
— *docker* : configure et applique des paramètres de durcissement de Docker pour l'exécution des épreuves.
— *bootstrap* : installe le service `systemd` permettant le chargement des épreuves conteneurisées en mode déconnecté.
— *syslog* : installe et configure de la journalisation centralisée de la machine virtuelle.
— *network* : active `systemd-networkd` comme gestionnaire de configuration réseau.

Cette tâche d'automatisation se termine par l'extraction de la partition système de la machine virtuelle qui est ensuite manuellement distribuée sur les dépôts d'infrastructure du FCSC. À l'instar de la préparation des noyaux, l'équipe d'administration souhaite dans le futur pouvoir automatiser de bout en bout la création et la distribution des images systèmes et élargir le nombre de distributions Linux et de systèmes d'exploitation supportés.

Des tests sont également en cours pour évaluer la possibilité d'utiliser de Nix [6, 7] afin de simplifier le processus de génération des noyaux et des images système. Cette évolution faciliterait aussi la réalisation d'images reproductibles, ce qui est particulièrement intéressant pour garantir que l'environnement des épreuves reste le même.

3 European Cybersecurity Challenge (ECSC)

L'ECSC est un événement européen coordonné par l'ENISA depuis 2014 dans le but de promouvoir le domaine de la cybersécurité chez les jeunes. Cet événement a lieu une fois par an, entre septembre et novembre, et s'inscrivait historiquement dans le cadre du cybermois. Il est notamment géré par un *Steering Committee* constitué de membres des différents pays participants à l'ECSC. Ce comité se réunit plusieurs fois par an et décide et échange sur les modalités d'organisation, des évolutions, de la communication, etc. Le pays hôte en charge de l'organisation effective de l'événement (épreuves, infrastructure, mais aussi nourriture, hôtel, etc.) est désigné par l'ENISA et change tous les ans (tableau 2).

D'un point de vue technique, l'ECSC se joue en équipe et s'étale sur deux journées avec, depuis 2022, un CTF de type « Jeopardy » le premier jour, et un CTF de type « Attaque/Défense » le deuxième jour.

Nous revenons dans cette section sur la sélection de la *Team France*, l'équipe qui représente la France durant l'ECSC.

Tableau 2. Historique des événements ECSC passés.

Année	Pays hôte	Participants	Position France
2014	Autriche (Fürstenfeld)	3	—
2015	Suisse (Lucerne)	6	—
2016	Allemagne (Düsseldorf)	10	—
2017	Espagne (Malaga)	15	—
2018	Royaume Uni (Londres)	17	2ème
2019	Roumanie (Bucarest)	20	7ème
2020	*annulé*	—	—
2021	République Tchèque (Prague)	19	4ème
2022	Autriche (Vienne)	28	3ème
2023	Norvège (Hamar)	28	5ème
2024	Italie (Turin)	*à venir*	—

Fig. 11. ECSC 2023 : les équipes sont disposées sur des tables de 10 personnes dans le Vikingskipet de Hamar (Norvège). La *Team France* est celle située le plus en bas sur l'image.

3.1 Constitution de la *Team France*

Répartition d'âge de l'équipe de France. Étant à destination des jeunes, les règles de l'ECSC imposent d'avoir une équipe de 10 personnes de moins de 25 ans, avec au moins cinq personnes de moins de 20 ans. Les plus jeunes se faisant plus rares, la compétition est généralement plus rude pour les 21-25 ans. En pratique, l'équipe française est composée de cinq personnes de moins de 20 ans (« Junior »), cinq personnes de moins de 25 ans (« Senior »). Nous sélectionnons aussi deux remplaçants pour chaque catégorie d'âge : ces remplaçants font partie de la *Team France*, même s'ils ne vont normalement pas à l'ECSC.

Entretiens pour constituer l'équipe. À la fin du FCSC, nous proposons des entretiens :

— au top cinq du classement général « Junior » et « Senior »,
— et au top trois du classement « Junior » et « Senior » dans chaque catégorie qualifiante (généralement cryptographie, rétro-ingénierie, exploitation binaire, web).

Selon les années, les entretiens permettent également d'identifier des profils de capitaine d'équipe ou d'administrateur système.

3.2 Entraînement de l'équipe

L'ECSC met au défi les équipes des différents pays sur une compétition Jeopardy similaire au FCSC. Selon les années, l'ECSC peut également proposer une phase de compétition dans un format « Attaque-Défense ».

Entraînement pour les CTF Jeopardy. La *Team France* participe à une dizaine de CTF Jeopardy en ligne entre juin et septembre. Grâce à ces participations, les membres de l'équipe apprennent à travailler ensemble. C'est aussi l'occasion pour les coachs d'identifier un ou plusieurs potentiels capitaines d'équipe.

Entraînement pour les CTF Attaque-Défense. En revanche, le FCSC ne prépare pas aux CTF de type Attaque-Défense. Il est donc important pour l'équipe de France de s'entraîner sur ce sujet avant l'ECSC.

Depuis 2022, l'équipe a participé aux deux principaux CTFs qui suivent ce mode de jeu sur cette période : le `ENOWARS` et le `FAUSTCTF`. En 2023, l'ENISA a organisé un entraînement avec toutes les équipes nationales.

Contrairement au Jeopardy, il est important en Attaque-Défense d'être équipé d'outils d'analyse réseau efficaces. Les outils classiques tels que `Wireshark` ou `tcpdump` ne sont pas adaptés à une compétition d'attaque-défense en temps limité et stressante. En 2022, la Team France a utilisé `Tulip` [29], un outil développé par d'autres équipes participant à l'ECSC. En 2023, une interface graphique (figure 12) pour la sonde réseau Suricata a été développée puis publiée par l'équipe [16].

4 Hackropole : archives des épreuves du FCSC

Hackropole est une plate-forme web archivant les épreuves des précédents FCSC. Elle est accessible sur `https://hackropole.fr/` (voir figure 13).

4.1 Motivation

Chaque année, entre 50 et 100 épreuves sont publiées pour le FCSC. Afin de garder la compétition intéressante et équitable, chacune de ces épreuves est nouvelle et créée pour l'occasion. Ces épreuves sont mises en ligne durant le FCSC et seulement pendant une dizaine de jours. À la fin de l'événement, l'infrastructure est démontée et les épreuves deviennent inaccessibles. Certains participants conservent les fichiers pour les regarder

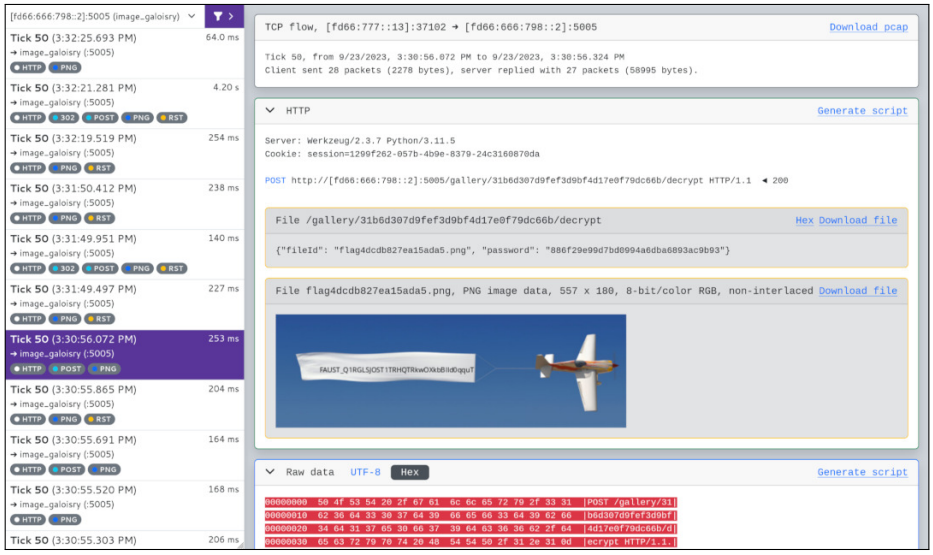

Fig. 12. Shovel : interface graphique pour visualiser les journaux de Suricata.

à nouveau dans les semaines qui suivent, mais il n'existait auparavant aucun archivage public.

Suite au retour de nombreux joueurs dès 2019, les épreuves écrites pour la sélection de l'ECSC 2019 ont été mise en ligne dans un dépôt Git sur le compte GitHub de l'ANSSI : `https://github.com/ANSSI-FR/ctf`. Néanmoins, faute de temps les années suivantes, les épreuves des FCSC n'ont pas été mises en ligne. Par ailleurs, ce format de publication sert uniquement à archiver les épreuves, sans réelle réutilisation.

Naissance de l'idée d'Hackropole. Dès 2020, l'idée de développer une plate-forme web d'archivage est née. Cette plate-forme a pour but de :

— créer des ressources pédagogiques pouvant être utilisées pour de l'autoformation, ou par l'enseignement supérieur français ;
— agréger des solutions contribuées par la communauté pour illustrer l'état de l'art des outils ou des méthodes de résolution innovantes ;
— mettre en avant les épreuves du FCSC à la communauté internationale (traduction anglaise et française).

Aspect non compétitif. Le FCSC est une compétition limitée dans le temps, ce qui peut être un format stressant et frustrant. Hackropole contraste avec le FCSC en n'offrant aucun classement sur la résolution des épreuves, mais en mettant en avant les utilisateurs qui contribuent au contenu de

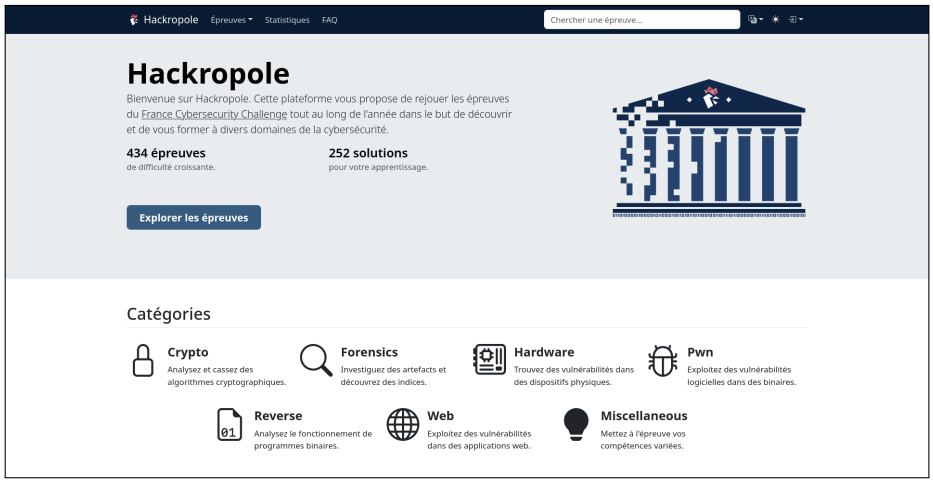

Fig. 13. Hackropole : Page d'accueil du site.

la plate-forme avec des solutions pédagogiques. Par ailleurs, toutes les épreuves d'Hackropole ont déjà été publiées dans le cadre du FCSC, donc les solutions à certaines de ces épreuves sont déjà publiques et en ligne. Pour cette raison, un classement basé sur la résolution comme d'autres sites le font aurait peu de sens.

4.2 Choix techniques

Objectifs. Au-delà des objectifs généraux du projet, plusieurs objectifs supplémentaires ont guidé les choix techniques pour la conception et la mise en ligne de la plate-forme :

- minimiser les besoins de maintenance et d'administration de la plate-forme ;
- ne pas nécessiter de compte pour accéder aux épreuves, aux solutions, et garder sa progression locale ;
- minimiser les informations personnelles traitées/stockées ;
- minimiser la surface d'attaque exposée et l'impact de la compromission de la plate-forme.

Images Docker pour les épreuves en ligne. Afin de ne pas avoir d'infrastructure d'épreuves à gérer, le choix a été fait de ne pas déployer les épreuves comme le fait le FCSC. À la place, des images Docker sont mises en ligne et peuvent être téléchargées et lancées par les participants. Cela permet de limiter l'infrastructure à un unique site Web.

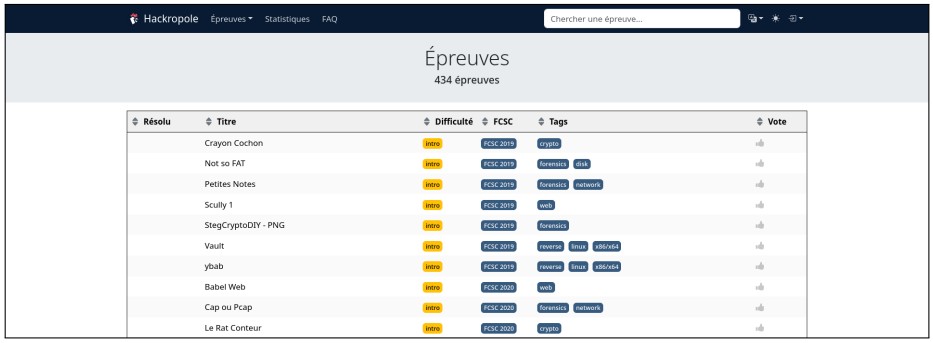

Fig. 14. Liste des épreuves sur Hackropole (capture d'écran, avril 2024).

Solutions accessibles. Contrairement à beaucoup de plate-formes de CTFs, les solutions des épreuves sont accessibles aux utilisateurs, même sans les avoir résolues. Ce choix a été fait pour renforcer l'aspect pédagogique d'Hackropole, pour que ses utilisateurs puissent apprendre, sans rester bloqués sur une épreuve.

Site statique avec API. Pour limiter encore l'infrastructure à maintenir et faciliter le déploiement et la résilience de la plate-forme, le site Hackropole est statique et généré automatiquement avec le moteur Hugo [1]. L'ensemble du contenu HTML, CSS et JavaScript est généré et servi directement aux utilisateurs. Cela a pour autre conséquence de limiter la surface d'attaque de la plate-forme et de garantir son bon fonctionnement futur avec une maintenance minimale.

Quelques interactions dynamiques sont tout de même implémentées à l'aide d'une API. Si un utilisateur souhaite sauvegarder sa progression, soumettre une solution ou voter pour une épreuve ou solution, cela passera par l'API. Pour simplifier l'hébergement, l'API a été développée en PHP et est accolée à une base de données MySQL.

Néanmoins, le site reste conçu pour être utilisé statiquement et ne nécessite pas cette API pour fonctionner. Il est toujours possible de parcourir les épreuves d'Hackropole et de les résoudre même en cas de problème technique de la base de données. Par exemple, le 26 décembre 2023, une coupure de courant a eu lieu dans le centre de données de notre hébergeur, ce qui a rendu complètement indisponible la base de données pendant plusieurs heures. Malgré cette indisponibilité, la plate-forme Hackropole est tout de même restée fonctionnelle et cela n'a causé que des désagréments mineurs (e.g., solution non soumise).

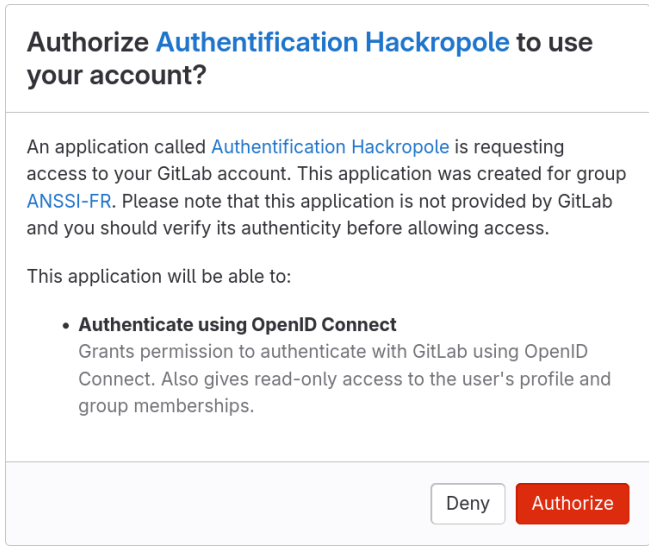

Fig. 15. Première connexion sur Hackropole via GitLab.

Délégation de l'authentification des utilisateurs. Une forme d'authentification est nécessaire pour permettre aux utilisateurs de sauvegarder leur progression d'un ordinateur à un autre. Cette authentification sert également pour comptabiliser des votes et soumettre des solutions. Implémenter cette authentification nécessiterait de stocker des noms d'utilisateurs et des condensats de mot de passe, et de les recevoir en clair. Cette fonctionnalité en impliquerait également d'autres, par exemple : stockage d'une adresse mail, gestion de la réinitialisation des mots de passe, assistance aux personnes n'arrivant plus à se connecter, etc.

Par conséquent, le choix est fait de déléguer l'authentification à des tiers et de ne stocker qu'un identifiant unique et un pseudonyme. Les fournisseurs d'authentification choisis sont, à date de rédaction, GitHub, GitLab et Discord. Nous avons choisi ces tiers, car il est habituel pour les joueurs de CTF d'être présent au moins sur l'une de ces trois plates-formes. Même si Hackropole ne les stocke pas, la plate-forme voit passer des jetons de compte GitHub, GitLab et Discord. Afin de limiter l'impact d'une possible compromission d'Hackropole, les jetons demandés ne permettent que la lecture des informations publiques des comptes GitHub, GitLab et Discord associés aux jetons.

Validation des flags. Comme toute plate-forme de CTF, Hackropole a besoin de pouvoir vérifier les *flags* soumis par les utilisateurs. Pour renforcer

(a) Avec JavaSript (b) Sans JavaScript

Fig. 16. Formulaire de soumission du flag sur une épreuve

la résilience du site, cette vérification se fait localement si l'utilisateur n'est pas connecté à la plate-forme. Ainsi, chaque page d'épreuve est générée avec un condensat SHA256 du flag attendu dans le code. Un script JavaScript vérifie sa valeur côté client, et si l'utilisateur a désactivé JavaScript, Hackropole affiche une commande Bash pour valider le flag sous Linux (figure 16).

Ouverture du code source. Dans l'ensemble des ressources servant à générer le site Hackropole, nous avons mis de côté les gabarits HTML, feuilles de style CSS et sources Javascript. Ce code a été publié sous forme d'un thème pour Hugo dans le but d'être réutilisable par une entité souhaitant publier des épreuves de CTF. Le code source est disponible sur `https://github.com/ANSSI-FR/hackropole-hugo` sous licence libre MIT.

4.3 Accueil de la plate-forme par le public

Hackropole a été officiellement lancée le 1er décembre 2023, après quelques semaines de tests par la *Team France*. Les épreuves ont été publiées progressivement jusqu'au 25 décembre (figure 17) afin de répartir la charge et de motiver les utilisateurs à revenir sur le site.

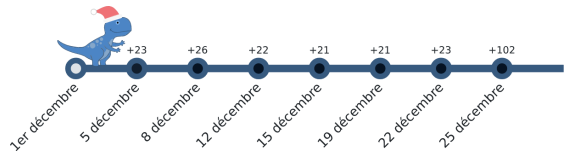

Fig. 17. Déblocage des épreuves pendant le mois de décembre 2023.

L'hébergeur choisi journalise les requêtes effectuées, nous avons donc pu observer les vagues de connexions début décembre (figure 18). Parce qu'il n'y a pas de traqueurs sur le site, le calcul du nombre de visiteurs

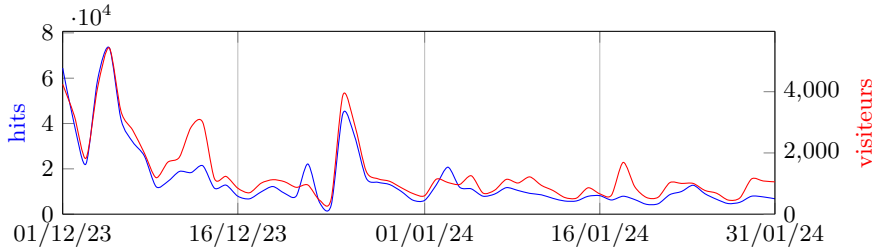

Fig. 18. Nombre de hits et de visiteurs (même adresse IP, même User-Agent, même jour) du 1er décembre 2023 au 31 janvier 2024.

par jour est nécessairement sous-estimé : il est évalué en comptant le nombre de couples d'adresses IP et d'User-Agent différents.[7] Les vagues de connexion sont corrélées à l'annonce d'Hackropole sur les réseaux sociaux. De janvier à mars 2024, il y a eu entre 500 et 1000 visiteurs par jour.

Au-delà de ces statistiques de connexion, Hackropole a été accueilli positivement sur les réseaux sociaux et par les médias [20]. Des échanges informels dans divers cercles ont également confirmé l'intérêt de la communauté pour ce projet. À titre d'exemple, certains professeurs de l'enseignement supérieur français et européen envisagent d'utiliser des épreuves tirées d'Hackropole pour illustrer des concepts de sécurité informatique.

Pendant décembre, 221 solutions ont été proposées par la communauté et acceptées par la modération. Ces solutions sont modérées manuellement et acceptées ou rejetées selon la pertinence de la solution proposée, de son côté pédagogique, sa qualité de rédaction, etc. À titre d'exemple, les solutions qui consistent à passer à côté des épreuves en allant chercher le flag directement à l'intérieur du conteneur Docker sont automatiquement rejetées.

La plate-forme étant encore jeune, nous restons attentifs au retour de la communauté et à son évolution.

5 Conclusion

Les pistes d'évolutions pour le FCSC ou la préparation à l'ECSC sont nombreuses. Tout d'abord, dans le cas du FCSC, les briques indépendantes d'infrastructure construites ces dernières années et détaillées dans cet article permettent d'envisager d'importantes évolutions. Une des pistes majeures que nous espérons explorer d'ici l'édition 2025 concerne la

[7] Plusieurs utilisateurs se connectant depuis des machines d'établissement scolaire partageant la même adresse IP seront comptés comme 1 unique visiteur.

possibilité pour les participants de démarrer une microVM individuelle à la demande. En effet, les VM et les conteneurs des épreuves sont pour l'instant partagés entre tous les participants, et les conséquences et limitations sont importantes. Cela demande par exemple un durcissement adapté des environnements afin que les utilisateurs soient suffisamment bien cloisonnés et qu'une mauvaise utilisation (malveillante ou non) n'impacte pas les autres utilisateurs. Ces problématiques deviendraient moins cruciales si les utilisateurs disposaient d'une VM individuelle qu'ils pourraient démarrer, éteindre, corrompre, etc. Du point de vue conception d'épreuves, cette possibilité offrirait également de nouvelles perspectives, avec par exemple des épreuves orientées réseau (e.g., *adversary-in-the-middle*), administration système, pentest, Windows, etc.

Toujours suite au FCSC, nous avons entamé la publication des améliorations et corrections que nous avons apportées au logiciel CTFd que nous utilisons comme interface utilisateur, et nous projetons de poursuivre dans cette direction. Il est prévu de proposer une contribution au code public de CTFd pour faciliter la mise en place de la vue matricielle du classement ainsi que les classements par catégories techniques que nous avons intégrés. La publication du code de certaines briques de l'infrastructure est également à l'étude.

Ensuite, dans le cadre de l'ECSC, les axes de développement sont plus limités étant donné que l'organisation de l'événement change tous les ans. En revanche, l'outillage pour les membres de la Team France pourrait être enrichi, en particulier pour le CTF de type « Attaque/Défense ». L'outil Shovel présenté précédemment a été développé pour cette occasion, mais des outils supplémentaires pourraient également être utiles. Dans ce mode de jeu, il est par exemple crucial d'être en mesure d'exécuter une même attaque en parallèle sur toutes les équipes adverses de manière simultanée. Ce type d'outil s'appelle un *launcher*, et bien qu'il en existe certains en source ouverte, par exemple `ataka`,[8] ils possèdent des limitations connues qui ne permettent pas d'être compétitif avec les meilleures équipes

Pour finir, de multiples pistes d'évolution de Hackropole sont envisageables, notamment pour toucher des personnes plus débutantes et leur faire découvrir des notions concrètes et techniques liées à la sécurité informatique, mais le temps des personnes impliquées est pour le moment le facteur le plus limitant.

Remerciements. Les auteurs de cet article souhaitent remercier l'ANSSI ainsi que toutes les personnes qui ont été impliquées de près ou de loin

[8] `https://github.com/OpenAttackDefenseTools/ataka`

dans l'organisation du FCSC ou la préparation à l'ECSC, et notamment Julien Raeis qui avait mis en place la première version de l'infrastructure utilisée pendant le FCSC en 2019. Les auteurs remercient également tous les participants du FCSC entre 2019 et 2024 ainsi que les alumnis de la Team France, sans qui le FCSC et la participation à l'ECSC ne seraient pas possibles.

Références

1. Hugo : The world's fastest framework for building websites. https://gohugo.io/.
2. ANSSI. Recommandations de sécurité relatives à un système GNU/Linux. https://cyber.gouv.fr/publications/recommandations-de-securite-relatives-un-systeme-gnulinux.
3. Cilium. Cilium bpf and xdp reference guide. https://docs.cilium.io/en/latest/bpf/.
4. Inc CommitGo. Gitea official website. https://about.gitea.com.
5. Kata Containers. Github - kata-containers. https://github.com/kata-containers.
6. NixOS contributors. Nix and nixos - declarative builds and deployments. https://nixos.org/.
7. Eelco Dolstra. The purely functional software deployment model. https://edolstra.github.io/pubs/phd-thesis.pdf.
8. The Linux Foundation. Linux kernel documentation for BPF maps. https://www.kernel.org/doc/html/v6.8/bpf/maps.html.
9. Google. Github - google/crosvm. https://github.com/google/crosvm.
10. Hadolitn. Github - hadolint/hadolint : Docker linter, validate inline bash, written in haskell. https://github.com/hadolint/hadolint.
11. HashiCorp. Packer : Create identical images for multiple platforms from a single source configuration. https://github.com/hashicorp/packer.
12. Red Hat. Ansible : simple it automation platform that makes your applications and systems easier to deploy and maintain. https://github.com/ansible/ansible.
13. Docker Inc. Docker : Docker helps developers bring their ideas to life by conquering the complexity of app development. https://github.com/docker.
14. Docker Inc. Swarm mode overview. https://docs.docker.com/engine/swarm.
15. Harness Inc. Drone ci – automate software testing and delivery. https://www.drone.io.
16. Ambre Iooss. Shovel : Web interface to explore suricata eve outputs. https://github.com/ANSSI-FR/shovel.
17. Cyril Jaquier. fail2ban : Daemon to ban hosts that cause multiple authentication errors. https://github.com/fail2ban/fail2ban.
18. The kernel development community. Btrfs - the linux kernel documentation. https://docs.kernel.org/filesystems/btrfs.html.
19. Traefik Labs. Traefik proxy documentation. https://doc.traefik.io/traefik.

20. David Latouche. Testez vos compétences en cybersécurité avec hackropole ! https://www.dane.ac-versailles.fr/spip.php?article717.
21. CTFd LLC. CTFd : CTFs as you need them. https://github.com/CTFd/CTFd.
22. Levente Polyak. linux-hardened : Minimal supplement to upstream kernel self protection project changes. https://github.com/anthraxx/linux-hardened.
23. Alexander Popov. kernel-hardening-checker : A tool for checking the security hardening options of the linux kernel. https://github.com/a13xp0p0v/kernel-hardening-checker.
24. Linux Fondation Projects. Cloud hypervisor - run cloud virtual machines securely and efficiently. https://www.cloudhypervisor.org/.
25. Inc. Red Hat. Ansible collaborative. https://www.ansible.com/.
26. Inc. Red Hat. Gluster. https://www.gluster.org.
27. Rust-vmm. Github - rust-vmm. https://github.com/rust-vmm.
28. Amazon Web Services. Firecracker. https://firecracker-microvm.github.io.
29. TeamEurope. Tulip is a flow analyzer meant for use during attack / defence ctf competitions. it allows players to easily find some traffic related to their service and automatically generates python snippets to replicate attacks. https://github.com/OpenAttackDefenseTools/tulip.