CTF : Capture The Flag

Intervenants :
Ahmad Adell - `ahmad.adell@insa-cvl.fr`
Jérémy Briffaut - `jeremy.briffaut@insa-cvl.fr`
Emile Ferrere - `emile.ferrere@insa-cvl.fr`

Challenges :
4A STI – Projet Sécurité

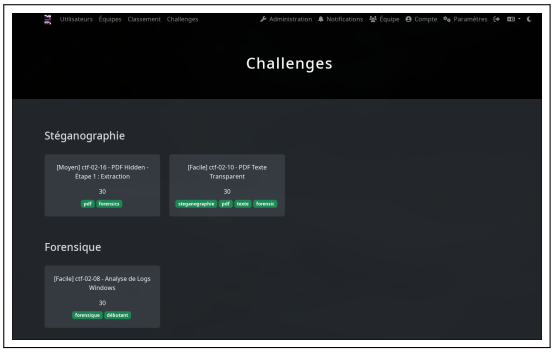


Table des matières

Chapitre 1

Introduction

1. Création des CTFs

Origine du projet CTF

Ces challenges CTF ont été conçus dans le cadre du module de Sécurité Informatique de 4e année du département STI (Sciences et Technologies de l'Information) à l'INSA Centre Val de Loire.

L'objectif pédagogique du projet est double :

- Mettre les étudiants en situation réelle d'analyse et d'exploitation de vulnérabilités.
- Concevoir des scénarios techniques cohérents intégrant à la fois attaque et remédiation.

Chaque groupe d'étudiants a ainsi conçu, testé et documenté un ou plusieurs challenges, couvrant différentes thématiques : authentification, cryptographie, web, gestion des secrets, configuration sécurisée, etc.

Ce projet s'inscrit dans une démarche pédagogique, visant à développer :

- La compréhension des vulnérabilités
- La capacité d'analyse offensive et défensive
- La rédaction technique

Étudiants contributeurs

TP	Nom	LinkedIn
TP1 - Groupe 1	GAYE Fama	https://www.linkedin.com/in/fama-gaye/
TP1 - Groupe 1	HABLOUZ Najib	https://www.linkedin.com/in/najib-hablouz-58b72b253/
TP1 - Groupe 1	ID KAROUM Mouad	https://www.linkedin.com/in/mouadidk/
TP1 - Groupe 1	KRILL Maxence	https://www.linkedin.com/in/maxkrl/
TP1 - Groupe 2	Erroussi Ayman	https://www.linkedin.com/in/ayman-erroussi/
TP1 - Groupe 2	El Mahjour Achraf	https://www.linkedin.com/in/achraf-el-mahjour/
TP1 - Groupe 2	El Idrissi Ismail	https://www.linkedin.com/in/ismail-el-idrissi-ing/
TP1 - Groupe 2	Terro Ali	https://www.linkedin.com/in/ali-terro/
TP1 - Groupe 3	Ahmad Saad	https://www.linkedin.com/in/ahmad-msaad/
TP1 - Groupe 3	Amer El Jibbawe	https://www.linkedin.com/in/amer-el-jibbawe/
TP1 - Groupe 3	Yasser Aamoud	https://www.linkedin.com/in/yasser-aamoud-68155b286/
TP1 - Groupe 3	Marwa Boumour	https://www.linkedin.com/in/marwa-boumour-854399306/
TP3 - Groupe 1	Yassine Aherrass	https://www.linkedin.com/in/yassine-aherrass
TP3 - Groupe 1	El Khlifi Ilias	https://www.linkedin.com/in/ilias-el-khlifi-1a96702b2/
TP3 - Groupe 1	Errais Adam	https://www.linkedin.com/in/adam-errais-3408b3334/
TP3 - Groupe 1	Yassine Ben Belgacem	https://www.linkedin.com/in/yassine-belgacem-77bab1387/
TP3 - Groupe 2	BARBIER Ted	https://www.linkedin.com/in/ted-barbier/
TP3 - Groupe 2	DIACK Amadou	https://www.linkedin.com/in/amadou-diack-mad/
TP3 - Groupe 2	DJEMAI Yacine	https://www.linkedin.com/in/yacine-d-877b112a0/
TP3 - Groupe 2	ESSERGHINI Hafsa	https://www.linkedin.com/in/hafsa-esserghini/
TP3 - Groupe 2	HABBACHICH Ilyass	—
TP3 - Groupe 3	Mikaël Ferreira	http://www.linkedin.com/in/mikaÃńl-ferreira-25b081343
TP3 - Groupe 3	Khalid El Abbari	https://www.linkedin.com/in/khalid-elabbari-9b665025b
TP3 - Groupe 3	Oumayma Ennamous	https://www.linkedin.com/in/oumayma-ennamous-ba3842294/

2. Objectifs

Cette formation a pour vocation d'initier les apprenants à la cybersécurité offensive et défensive au travers d'une approche ludique et progressive par la pratique des *Capture The Flag* (CTF). Les objectifs sont organisés en deux volets : la découverte de la démarche CTF elle-même, puis l'acquisition de compétences techniques dans les grandes familles de vulnérabilités rencontrées.

- **Comprendre et pratiquer la démarche CTF**
  - Comprendre le principe d'un CTF : trouver un *flag* caché en exploitant une vulnérabilité ou en résolvant un puzzle technique.
  - Découvrir la plateforme **CTFd** : création de compte, soumission de flags, consultation du classement et des catégories de challenges.
  - Appréhender les niveaux de difficulté (*Facile*, *Moyen*, *Difficile*) et adopter une progression méthodique.
  - Identifier les grandes catégories de challenges : Web, Système, Réseau, Cryptographie, Stéganographie, Forensique, Reverse, OSINT, Programmation.
  - Développer une méthodologie de résolution : reconnaissance, hypothèse, exploitation, validation du flag.
  - Apprendre à utiliser les indices (*hints*) et à documenter sa démarche pour restituer une solution (*write-up*).
  - Travailler en équipe : répartition des challenges, partage des découvertes, communication des avancées.
- **Maîtriser les fondamentaux de la sécurité web**
  - Comprendre et exploiter les failles d'authentification par jeton : secret JWT faible, attaque par dictionnaire avec `hashcat`, forge de token et élévation de rôle.

  - Identifier et exploiter les vulnérabilités d'upload de fichiers arbitraires (webshell PHP, plugin WordPress vulnérable).
  - Réaliser des injections LDAP et des traversées de répertoires.
  - Exploiter des ressources exposées par mauvaise configuration (`robots.txt`, répertoires cachés, métadonnées).
  - Manipuler des cookies et mécanismes de session côté client.
- **Appréhender la sécurité des systèmes**
  - Analyser et exploiter des droits fichiers mal configurés (permissions UNIX).
  - Comprendre et mettre en œuvre une attaque de type *PATH Hijacking*.
  - Identifier des services systemd obsolètes ou mal sécurisés laissés en production.
  - Utiliser **Metasploit** pour exploiter un service vulnérable et obtenir un accès système.
  - Réaliser une élévation de privilèges (*privesc*) via `sudo`, bit SUID ou `sqlite3`.
- **Comprendre la sécurité réseau et les protocoles**
  - Capturer et analyser du trafic réseau (fichiers PCAP, Wireshark).
  - Identifier et exploiter des services mal exposés : NFS, FTP anonyme, SMB, service TCP non authentifié.
  - Effectuer des pivots SSH entre segments réseau isolés (DMZ, DMZINT, ADM).
  - Compromettre un partage SMB et exploiter une tâche `cron` pour créer une persistance.
  - Analyser une poignée de main TLS/TCP incomplète (*Missing Handshake*).
- **Pratiquer le forensique numérique**
  - Analyser des journaux système (logs SSH, logs Windows) pour détecter une intrusion.
  - Extraire des informations sensibles depuis des métadonnées de fichiers (PDF, images EXIF).
  - Identifier les traces d'une compromission dans un système de fichiers.
- **Découvrir la stéganographie**
  - Détecter et extraire des données dissimulées dans des images ou du bruit visuel.
  - Décoder un message transmis par signal SSTV et le croiser avec de l'OSINT.
  - Reconstituer un fichier fragmenté dissimulé dans un PDF en plusieurs étapes.
- **S'initier à la cryptographie appliquée**
  - Décoder et identifier des encodages classiques (Morse, Base64, chiffrements par substitution).
  - Comprendre le chiffrement AES et réaliser un déchiffrement à partir d'une clé extraite lors d'une phase d'exploitation.
  - Mener une attaque par dictionnaire sur un secret cryptographique (`hashcat`, wordlist `rockyou.txt`).
- **S'initier au reverse engineering et à la programmation**
  - Analyser un binaire ELF et identifier ses conditions de validation sans accès au code source.
  - Analyser une application Android (APK) pour en extraire des secrets ou des credentials codés en dur.
  - Écrire de petits scripts Python pour automatiser une exploitation ou valider une hypothèse.
- **Développer une démarche OSINT**
  - Rechercher des informations publiques sur des personnes ou des organisations à partir de sources ouvertes.
  - Croiser des sources hétérogènes (LinkedIn, réseaux sociaux, images, métadonnées) pour identifier une cible ou un contexte.
- **Réaliser une chaîne d'attaque complète (scénario TechCorp)**
  - Identifier un point d'entrée web et obtenir une exécution de code à distance (RCE via upload PHP).
  - Escalader les privilèges sur une machine en zone DMZ.
  - Pivoter en SSH vers un réseau interne cloisonné (DMZINT).

- Compromettre un partage SMB et créer un SUID persistant via une tâche cron.
- Accéder à une machine de production (zone ADM) via une escalade `sudo sqlite3`.
- Exfiltrer et déchiffrer des données sensibles (AES) pour obtenir le flag final : comprendre ainsi l'enchaînement complet d'une attaque réelle sur une infrastructure segmentée.

3. Architecture

4. Infrastructure mise en place

L'infrastructure déployée pour cette formation repose sur une plateforme de virtualisation **Proxmox VE** hébergée sur une machine virtuelle dédié (`proxmox-ctf-bacpro.cyberfarm.insa-cvl.fr`) sur la CyberFarm de CyberINSA. Elle met en œuvre 45 machines virtuelles réparties sur 7 segments réseau isolés, interconnectés par un pare-feu central. L'ensemble reproduit fidèlement une infrastructure d'entreprise segmentée, conçue pour exposer progressivement les apprenants à des vulnérabilités réalistes.

Hyperviseur et stockage

La couche de virtualisation s'appuie sur **Proxmox VE**, solution open source basée sur **KVM/QEMU**. Chaque VM dispose de ressources allouées indépendamment (vCPU, RAM, disque). Le stockage est géré par **ZFS**, qui garantit l'intégrité des données et facilite la création de snapshots pour la réinitialisation rapide des machines après exploitation. Les sauvegardes sont assurées par **Proxmox Backup Server (PBS)**.

Segmentation réseau et pare-feu

Le réseau est découpé en **7 segments** distincts, chacun porté par un bridge virtuel Proxmox (`vmbr0` à `vmbr6`) :

- **WAN** (`vmbr0` – `172.16.0.0/16`) : interface de sortie vers le réseau INSA-CVL, point d'entrée des participants depuis les salles.
- **LAN** (`vmbr1` – `10.0.0.0/24`) : réseau principal hébergeant les 33 VM de challenges des séries 01 à 04.
- **INFRA** (`vmbr2` – `10.1.0.0/24`) : réseau isolé dédié à la plateforme CTFd (`10.1.0.10`).
- **DMZ** (`vmbr3` – `192.168.100.0/24`) : zone démilitarisée exposant les services web accessibles depuis le LAN.
- **DMZINT** (`vmbr4` – `10.0.1.0/24`) : réseau interne de rebond, inaccessible directement depuis l'extérieur.
- **ADM** (`vmbr5` – `10.0.2.0/24`) : zone d'administration, cible finale du scénario de chaîne d'attaque.
- **KALI** (`vmbr6` – `10.0.3.0/24`) : réseau dédié aux machines Kali Linux des participants, accessible via un proxy noVNC.

Le routage inter-segments et le filtrage sont entièrement assurés par **OPNsense** (VM 1000), dérivé de FreeBSD, qui concentre les 7 interfaces réseau (`vtnet0`–`vtnet6`). La politique de filtrage applique le principe du *moindre privilège* : les réseaux INFRA, KALI et WAN sont soumis à des règles `BLOCK any` → `any` par défaut, tandis que DMZ, DMZINT et ADM sont volontairement ouverts pour permettre les enchaînements d'exploitation propres aux scénarios CTF. L'interface d'administration OPNsense est exposée sur le port `:444` et accessible en SSH depuis le réseau INSA uniquement.

En complément, **nftables** assure le filtrage au niveau de l'hyperviseur Proxmox VE via les groupes de sécurité (*firewall groups*), offrant une deuxième couche de protection indépendante d'OPNsense avec une politique de refus par défaut sur chaque bridge.

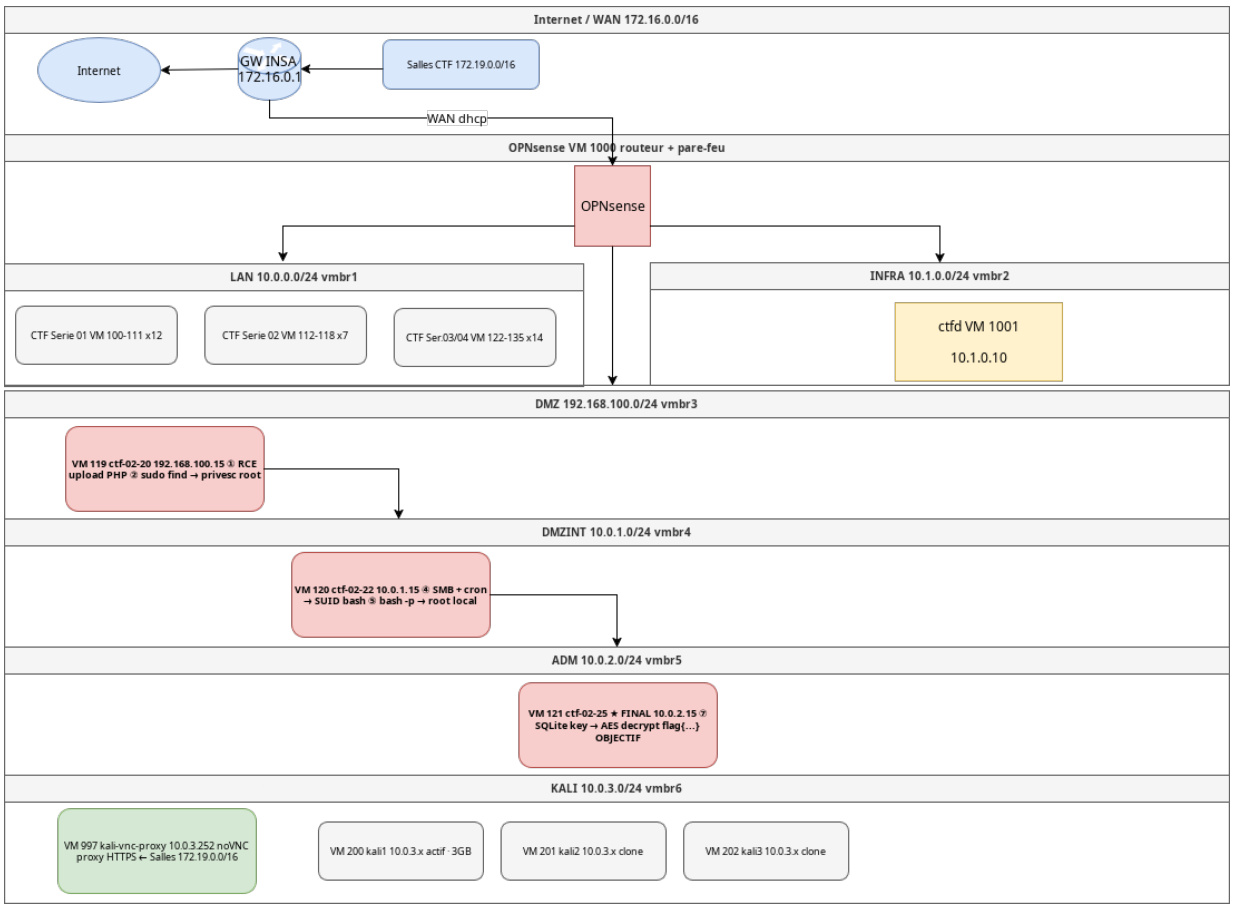


FIGURE 1.1 – L'infrastructure mise en place

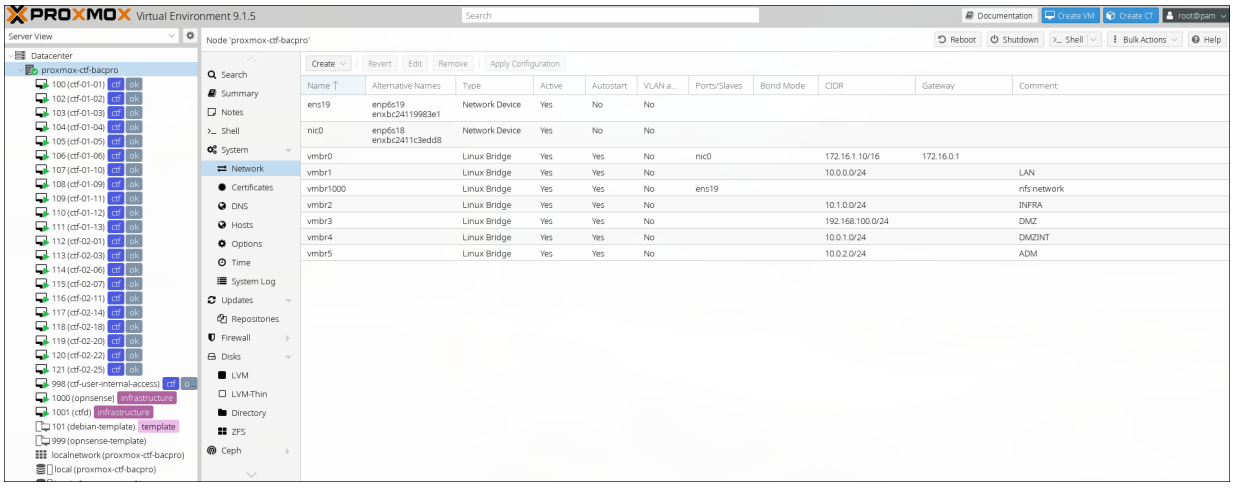


FIGURE 1.2 – L'infrastructure mise en place

Accès des participants

La plateforme **CTFd** est accessible sur `http://opnsense-ctf-bacpro.cyberfarm.insa-cvl.fr:8080/` depuis le réseau LAN ; elle expose le tableau de bord, les catégories de challenges, le système de soumission de flags et le classement en temps réel.

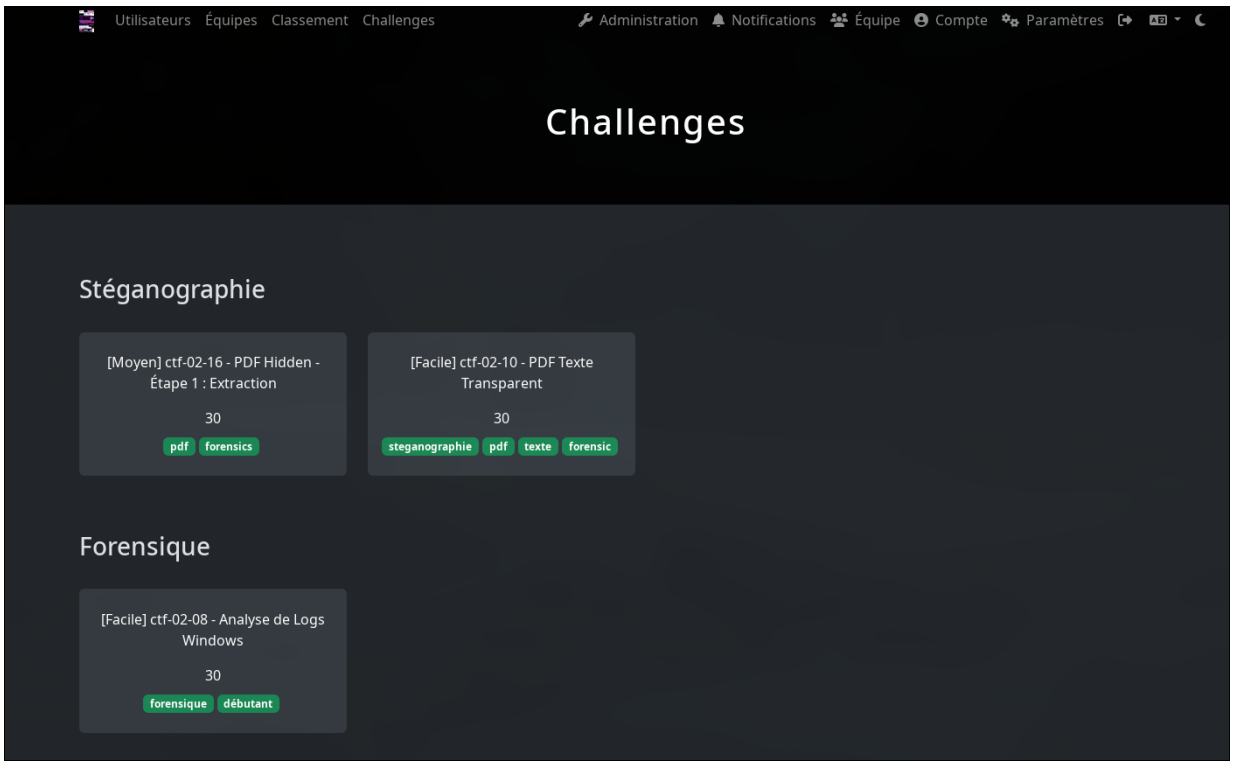


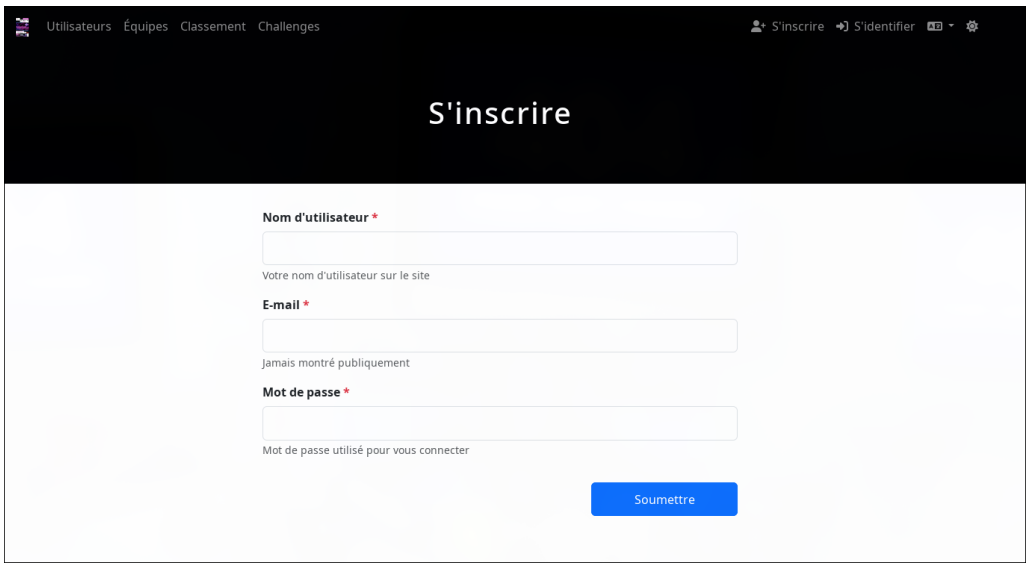
FIGURE 1.3 – Le site principal : CTFd

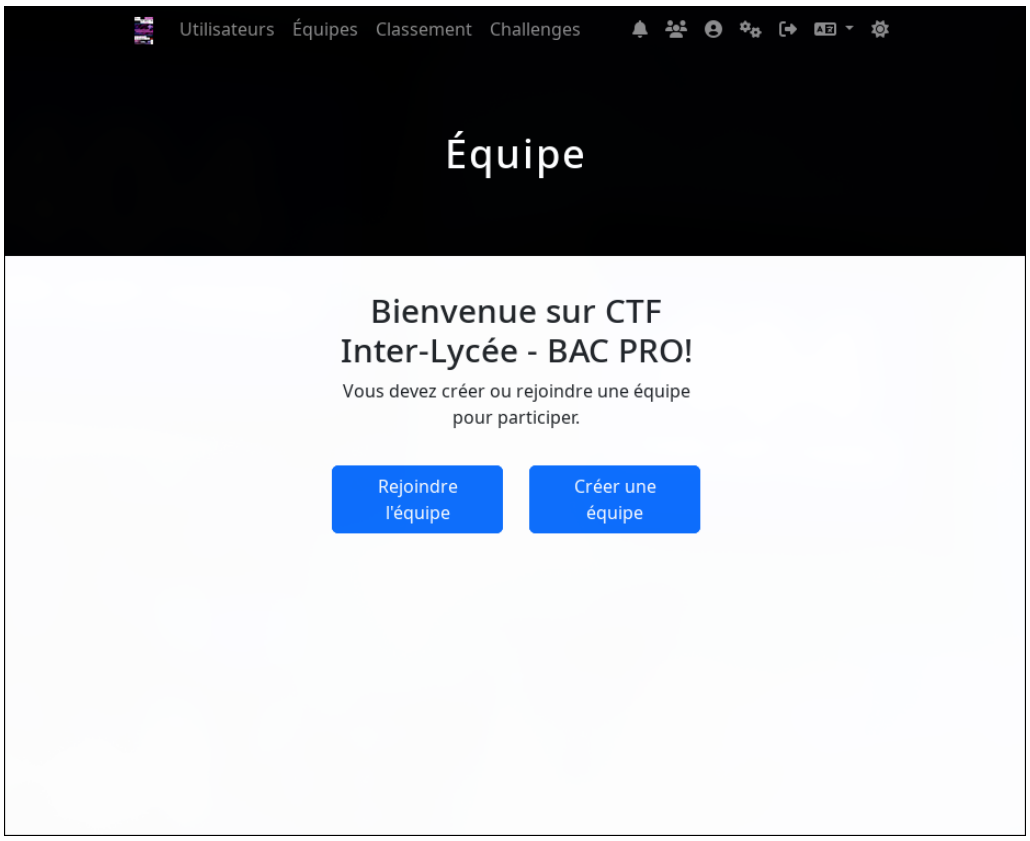
Chapitre 2

Accès aux challennges

Accéder au site : `http://opnsense-ctf-bacpro.cyberfarm.insa-cvl.fr:8080/challenges`

Créer un compte (lien inscription) :





Créer ou rejoindre une équipe :

Chapitre 3

CTF

JWT – Secret faible

Nom :	`JWT - Secret faible`		
Catégorie :	`Web`	**Niveau :**	`Facile`
Code :	`ctf-01-01`	**Tags :**	`auth` `jwt` `http` `recon`

1 - Vulnérabilité Exploitée

Mauvaise gestion d'un secret cryptographique utilisé pour signer un JSON Web Token (JWT) avec l'algorithme HS256.

Le mécanisme cryptographique est correct, mais la clé secrète utilisée est faible et peut être retrouvée par attaque par dictionnaire.

2 - Scénario

Le développeur de cette application affirme que son système d'authentification par jeton est totalement sécurisé, car la clé secrète est… secrète.

Cependant, lors des phases de test, il utilise souvent des mots de passe simples et oublie de les modifier en production.

L'application repose sur un mécanisme d'authentification basé sur des JSON Web Tokens (JWT) signés en HS256.

Objectif

1. Se connecter avec le compte invité fourni.
2. Récupérer le JWT stocké côté client.
3. Identifier l'algorithme utilisé.
4. Retrouver la clé secrète par attaque par dictionnaire.
5. Forger un nouveau jeton avec le rôle administrateur.
6. Récupérer le flag.

Accès

- URL : `http://opnsense-ctf-bacpro.cyberfarm.insa-cvl.fr:1001/`
- Compte invité : user / user123
- Format du flag : {CTF_INSA_STI_FACILE_01_<...>}

3 - Outils recommandés

- Navigateur Web (Inspecteur → LocalStorage)
- jwt.io (décodeur / encodeur JWT)
- Kali Linux
- hashcat

- Wordlist rockyou.txt

4 - Indices

- Le jeton est stocké côté client.
- Le header contient un champ `alg`.
- L'algorithme utilisé est symétrique.
- Le payload contient un champ `role`.
- Une clé faible peut être retrouvée par dictionnaire.

La Galerie

Nom :	La Galerie		
Catégorie :	Web	**Niveau :**	Facile
Code :	ctf-01-03	**Tags :**	auth http recon

1 - Vulnérabilité Exploitée

Mauvaise configuration du serveur web permettant l'affichage du contenu d'un répertoire.

Le listing de dossier expose des fichiers internes qui n'auraient jamais dû être accessibles publiquement.

2 - Scénario

Bienvenue sur la Galerie d'Art Numérique.

Le site présente différentes œuvres hébergées sur un serveur récemment migré. L'administrateur affirme que tout fonctionne correctement depuis la transition, mais certains éléments semblent avoir été oubliés pendant l'opération.

Des ressources internes pourraient être encore accessibles publiquement.

Objectif

1. Explorer le site web.
2. Identifier l'emplacement des fichiers médias.
3. Accéder directement au répertoire.
4. Trouver un fichier confidentiel exposé.
5. Lire son contenu et récupérer le flag.

Accès

- URL : `http://opnsense-ctf-bacpro.cyberfarm.insa-cvl.fr:1003/`
- Format du flag : `{CTF_INSA_STI_FACILE_03_<...>}`

3 - Outils recommandés

- Navigateur web
- Inspection d'URL
- Observation du code source

4 - Indices

- Les images d'un site sont stockées quelque part.
- L'URL d'une ressource révèle souvent son emplacement.
- Certains serveurs affichent la liste des fichiers d'un dossier.
- Les fichiers texte oubliés peuvent contenir des informations sensibles.

Bip bip boup boup

Nom : Bip bip boup boup

Catégorie : Programmation

Niveau : Facile

Code : ctf-01-04

Tags : netcat, sockets, tcp, python, scripting, automation

1 - Vulnérabilité Exploitée

Absence de protection contre l'automatisation côté serveur.

Le service repose uniquement sur une contrainte de rapidité humaine, supposant qu'un utilisateur répond manuellement. Il ne prévoit aucun mécanisme anti-bot ni limitation de requêtes.

2 - Scénario

Dans un futur dystopique où les machines dominent le monde numérique, l'accès aux données est réservé aux intelligences artificielles.

Vous tentez d'infiltrer un mainframe sécurisé gardé par un programme arrogant : le Système de Vérification v1.0.

Son unique règle :

Seuls les esprits capables de répondre instantanément peuvent passer.

Il vous défie de résoudre une série de calculs à une vitesse impossible pour un humain.

Objectif

1. Se connecter au service TCP.
2. Comprendre le fonctionnement du challenge.
3. Automatiser les réponses.
4. Résoudre toutes les équations.
5. Obtenir le flag.

Accès

- Connexion : `nc opnsense-ctf-bacpro.cyberfarm.insa-cvl.fr 1004`
- Format du flag : {CTF_INSA_STI_FACILE_04_<...>}

3 - Outils recommandés

- Python
- Bibliothèque socket
- Netcat

4 - Indices

- Un humain ne peut pas répondre assez vite.
- Les machines peuvent.
- Les services réseau se manipulent facilement avec des scripts.
- Python permet d'évaluer des expressions mathématiques dynamiquement.

Analyse d'Activité SSH Suspecte

Nom : Analyse d'Activité SSH Suspecte

Catégorie : Système

Niveau : Facile

Code : ctf-01-05

Tags : ssh, bruteforce, privesc, filesystem, linux, credentials

1 - Vulnérabilité Exploitée

Il ne s'agit pas d'une vulnérabilité technique classique, mais d'une **mauvaise surveillance et détection des connexions SSH**.

Un attaquant a tenté d'accéder au serveur via SSH avec des identifiants inconnus, exploitant un potentiel manque de contrôle d'accès ou de surveillance en temps réel.

2 - Scénario

Une activité suspecte a été détectée sur les serveurs de l'entreprise : plusieurs tentatives de connexion SSH depuis une adresse IP inconnue ont été enregistrées.

Les logs du serveur contiennent toutes les informations nécessaires pour identifier l'intrus et générer le flag.

Votre mission consiste à :

- Accéder au serveur en tant que participant.
- Analyser le fichier de logs SSH.
- Identifier l'adresse IP suspecte et l'utilisateur ciblé.
- Soumettre le flag associé via le programme $verify\_incident$.

Objectif

1. Se connecter au serveur cible via SSH.
2. Localiser le fichier de logs SSH (‘logs/auth.log‘).
3. Rechercher les tentatives échouées ou utilisateurs invalides.
4. Identifier l'IP de l'attaquant, l'utilisateur ciblé, l'heure et le port.
5. Vérifier le flag avec $./verify\_incident$.

Accès

- SSH : `ssh participant@opnsense-ctf-bacpro.cyberfarm.insa-cvl.fr -o PubkeyAuthentication=no -p 1005`
- Identifiants : `participant` / `participant`
- Format du flag : `{CTF_INSA_STI_FACILE_05_<...>}`

3 - Outils recommandés

- Terminal Linux / SSH
- Commandes : `ls`, `cat`, `grep`, `less`, `awk`, `tail`
- Analyse du fichier : `logs/auth.log` dans le répertoire utilisateur

4 - Indices

- Chercher les lignes contenant `Failed password` ou `Invalid user`.
- Identifier les adresses IP non reconnues.
- Vérifier le port source associé à chaque tentative.
- Le programme *verify_incident* permet de confirmer la bonne identification.

Double verrou – Fichier masqué

Nom :	Double verrou - Fichier masqué		
Catégorie :	Reverse	**Niveau :**	Facile
Code :	ctf-01-07	**Tags :**	reverse strings obfuscation

1 - Vulnérabilité Exploitée

Obfuscation faible reposant sur deux techniques simples :

- extension de fichier trompeuse
- encodage ROT13 facilement réversible

Ces protections reposent uniquement sur l'obscurcissement et non sur un mécanisme de sécurité réel.

2 - Scénario

Un agent de terrain a intercepté un fichier suspect transmis entre deux machines compromises.

Le fichier semble contenir des informations importantes, mais son ouverture échoue avec les outils habituels. Les analystes pensent qu'il a été volontairement dissimulé à l'aide de techniques simples destinées à tromper une inspection rapide.

Votre mission consiste à analyser ce fichier et révéler ce qu'il cache réellement.

Objectif

1. Identifier le véritable type du fichier.
2. Restaurer son format exploitable.
3. Lire son contenu.
4. Décoder le message caché.
5. Récupérer le flag.

Accès

- Fichier fourni : `Double_verrou.bin`
- Format du flag : {CTF_INSA_STI_FACILE_07_<...>}

3 - Outils recommandés

- Commandes Linux
- file
- mv
- echo / tr

4 - Indices

- Une extension n'indique pas toujours le vrai type d'un fichier.
- Linux peut identifier le format réel.
- Les images peuvent contenir du texte.
- Certains encodages simples transforment uniquement les lettres.

Bruit de fond – Analyse réseau

Nom :	Bruit de fond - Analyse réseau		
Catégorie :	Réseau	**Niveau :**	Facile
Code :	ctf-01-08	**Tags :**	pcap sniffing enumeration services

1 - Vulnérabilité Exploitée

Transmission d'informations sensibles en clair sur le réseau.

Le fichier de configuration a été téléchargé via HTTP sans chiffrement, permettant son interception et son analyse hors ligne.

2 - Scénario

Une capture réseau provenant d'un poste suspect a été interceptée lors d'une surveillance.

L'utilisateur semblait naviguer normalement, générant un trafic important composé d'images, scripts, publicités et requêtes diverses. Pourtant, au milieu de ce bruit, une action sensible aurait eu lieu : le téléchargement d'un fichier de configuration contenant une information secrète.

Votre mission consiste à analyser cette capture réseau afin d'identifier et d'extraire ce fichier.

Objectif

1. Ouvrir la capture réseau.
2. Filtrer le trafic pertinent.
3. Identifier les fichiers transférés.
4. Trouver le bon fichier.
5. Décoder le secret.
6. Récupérer le flag.

Accès

- Fichier fourni : `Bruit_de_fond.pcap`
- Format du flag : `{CTF_INSA_STI_FACILE_08_<...>}`

3 - Outils recommandés

- Wireshark
- Tshark
- CyberChef
- base64

4 - Indices

- Le flag n'est pas visible en clair.
- Le trafic chiffré n'est pas utile ici.
- Les objets HTTP peuvent être extraits automatiquement.
- Certains encodages sont reconnaissables visuellement.

Nothing to see here – Service exposé

Nom :	Nothing to see here - Service exposé		
Catégorie :	Système	**Niveau :**	Facile
Code :	ctf-01-09	**Tags :**	privesc systemd Linux

1 - Vulnérabilité Exploitée

Exposition d'un secret dans la ligne de commande d'un service.

Le service contient un mot de passe en argument visible par tout utilisateur pouvant inspecter les processus ou les unités systemd.

2 - Scénario

Un administrateur affirme avoir déployé un service propriétaire de synchronisation pour sécuriser son serveur.

Cependant, il remarque :

- des ralentissements
- des processus suspects
- une activité inhabituelle

Il vous demande d'auditer la machine afin de vérifier si ce service est réellement sécurisé.

Objectif

1. Se connecter au serveur.
2. Identifier le service de synchronisation.
3. Analyser sa configuration.
4. Trouver un secret exposé.
5. Utiliser ce secret pour obtenir le flag.

Accès

- SSH : `ssh participant@opnsense-ctf-bacpro.cyberfarm.insa-cvl.fr -o PubkeyAuthentication=no -p 1009`
- Identifiants : `participant` / `participant`
- Format du flag : `{CTF_INSA_STI_FACILE_09_<...>}`

3 - Outils recommandés

- Commandes Linux
- systemctl
- ps
- grep

4 - Indices

- Les services tournent en arrière-plan.
- La ligne de commande d'un processus peut révéler des informations.
- Les fichiers .service décrivent comment un programme est lancé.
- Certains binaires attendent une clé d'accès.

Portes ouvertes

Nom : `Portes ouvertes`

Catégorie : `Réseau`

Niveau : `Facile`

Code : `ctf-01-10`

Tags : `sniffing` `ftp` `enumeration` `services`

1 - Vulnérabilité Exploitée

- Serveur FTP exposé publiquement.
- Compte `anonymous` activé.
- Présence d'informations sensibles accessibles sans authentification.
- Encodage Base64 utilisé pour masquer des identifiants.

2 - Scénario

Une organisation stocke ses archives sur un ancien serveur FTP.

L'administrateur considère que les données sont protégées car aucun mot de passe n'a été communiqué. Cependant, une configuration par défaut permet un accès non authentifié.

Objectif

- Accéder au serveur FTP.
- Identifier les fichiers accessibles.
- Exploiter les informations trouvées pour obtenir le flag final.

Accès

- FTP : `ftp opnsense-ctf-bacpro.cyberfarm.insa-cvl.fr 1010`
- Identifiant : `anonymous`
- Format du flag : `{CTF_INSA_STI_FACILE_10_<...>}`

3 - Outils recommandés

- Client FTP
- Commande `base64`
- Commandes Linux standards (`cat`, `echo`)

4 - Indices

- Tester l'utilisateur `anonymous`.
- Lire tous les fichiers accessibles.
- Un encodage n'est pas un chiffrement.

Hit Parade

Nom : Hit Parade
Catégorie : OSINT
Code : ctf-01-12
Niveau : Moyen
Tags : recon ssh enumeration social-engineering

1 - Vulnérabilité Exploitée

Mot de passe SSH faible présent dans une wordlist publique, combiné à un nom d'utilisateur facilement identifiable par OSINT sur le site web de la cible.

Le service SSH est exposé sans restriction d'accès, et le mot de passe choisi par l'utilisateur est trivial et directement lié au thème de son blog.

2 - Scénario

Thomas, un étudiant en informatique nostalgique des années 2000, tient un blog personnel hébergé sur son propre serveur. Il y partage ses coups de cœur musicaux, parlant de toute la scène rock/punk de cette décennie.

Son serveur expose un accès SSH sans restriction. Son nom d'utilisateur est visible partout sur le site, et son mot de passe, bien que non affiché, peut être deviné grâce à une attaque par dictionnaire.

Objectif

1. Visiter le blog et identifier le nom d'utilisateur.
2. Repérer le service SSH exposé (port 22).
3. Lancer une attaque par dictionnaire avec `rockyou.txt`.
4. Se connecter en SSH avec les identifiants trouvés.
5. Lire le flag dans `/home/thomas/flag.txt`.

Accès

- URL : `http://opnsense-ctf-bacpro.cyberfarm.insa-cvl.fr:1012/`
- SSH : `ssh thomas@opnsense-ctf-bacpro.cyberfarm.insa-cvl.fr -o PubkeyAuthentication=no -p 1112`
- Fichier fourni : rockyou.txt
- Format du flag : `{CTF_INSA_STI_MOYEN_02_<...>}`

3 - Outils recommandés

- Navigateur Web (reconnaissance OSINT)
- Hydra ou Medusa (brute-force SSH)
- Wordlist `rockyou.txt`
- Client SSH (`ssh`)

4 - Indices

- Le nom d'utilisateur est omniprésent sur le blog (titre, footer, auteur des articles).
- Il n'y a pas de formulaire de connexion web ; le serveur est administré à distance via SSH.
- Le thème du blog (rock/punk des années 2000) oriente vers le contenu de la wordlist.
- La chanson *We Will Rock You* de Queen est un indice subtil pointant vers `rockyou.txt`.
- Le mot de passe est probablement en lien avec un groupe cité sur le blog.

Manipulation de Cookies

Nom : Manipulation de Cookies

Catégorie : Web

Niveau : Facile

Code : ctf-02-03

Tags : auth cookies session http recon

1 - Vulnérabilité Exploitée

Le contrôle d'accès repose sur un cookie manipulable côté client.

Le rôle de l'utilisateur est stocké en Base64 dans le navigateur sans vérification côté serveur, permettant une élévation de privilèges.

Références : - OWASP Top 10 2021 – A01 :2021 Broken Access Control - MITRE ATT&CK – T1556.003 : Web Session Cookie

2 - Scénario

L'interface d'administration de *TechCorp* est protégée par un mécanisme d'authentification classique.

Une inspection des cookies révèle que le rôle est stocké côté client.

Votre mission consiste à tester si cette implémentation permet d'obtenir un accès administrateur sans connaître le mot de passe correspondant.

Objectif

1. Se connecter avec un compte utilisateur standard.
2. Identifier le cookie contenant le rôle.
3. Décoder et comprendre sa structure.
4. Modifier le cookie pour attribuer le rôle `admin`.
5. Vérifier si l'accès administrateur est accordé.

Accès

- URL : `http://opnsense-ctf-bacpro.cyberfarm.insa-cvl.fr:1032/`
- Login : `user1`
- Mot de passe : `welcome123`

3 - Outils recommandés

- Navigateur Web avec outils de développement (F12)
- Terminal Linux
- CyberChef (facultatif)

4 - Indices

- Observer les cookies après connexion.
- Vérifier si des informations sensibles sont stockées côté client.
- Chercher le cookie contenant le rôle.
- Tester une modification locale de ce cookie.

Application Android Compromise

Nom : Application Android Compromise

Catégorie : Reverse **Niveau :** Facile

Code : ctf-02-04 **Tags :** android apk strings obfuscation

1 - Vulnérabilité Exploitée

Stockage de secrets en dur (*hardcoded credentials*) dans le code source d'une application Android, combiné à un encodage Base64 utilisé à tort comme mécanisme de dissimulation.

Un APK est une archive décompilable : tout secret embarqué dans le code est potentiellement accessible à quiconque disposant du fichier.

2 - Scénario

TechCorp utilise une application Android interne pour ses employés. L'application intègre une logique d'authentification qui vérifie les identifiants directement côté client, en les comparant à des valeurs stockées en dur dans le code.

Le mot de passe est encodé en Base64 pour tromper un observateur non averti, mais cet encodage n'offre aucune protection réelle.

Objectif

1. Décompiler l'APK fourni avec JADX.
2. Localiser les identifiants cachés dans le code source.
3. Décoder le mot de passe encodé en Base64.
4. Se connecter au portail web d'administration et récupérer le flag.

Accès

- Fichier fourni : `techcorp.apk`
- URL du portail Web : découverte dans l'APK ou fournie avec le challenge.
- Format du flag : CTF_INSA_STI_FACILE_4_/login%motdepasse

3 - Outils recommandés

- JADX / JADX-GUI (décompilateur APK)
- CyberChef (décodage Base64 en ligne)
- Commande `base64` sous Linux
- Navigateur Web

4 - Indices

- Un fichier `.apk` est une archive : son contenu peut être exploré directement.
- Des outils comme JADX permettent de retrouver le code source Java à partir d'un APK.
- Cherchez des chaînes de caractères suspectes dans la classe `MainActivity`.
- Le mot de passe est encodé en Base64 – ce n'est pas du chiffrement.

Trouver le Compte LinkedIn

Nom :	Trouver le Compte LinkedIn		
Catégorie :	OSINT	**Niveau :**	Facile
Code :	ctf-02-05	**Tags :**	recon, enumeration, social-engineering, linkedin

1 - Vulnérabilité Exploitée

Divulgation involontaire d'informations sensibles via une photo publiée sur un réseau social professionnel. Un post-it contenant un secret est visible en arrière-plan d'une photo de bureau partagée publiquement.

Cette faille relève de l'*information disclosure* par négligence : l'auteur n'a pas vérifié le contenu visible derrière son sujet principal avant de publier.

2 - Scénario

Alexandre Mercier, RSSI chez TechCorp Industries à Lyon, est actif sur LinkedIn où il partage régulièrement son expérience en cybersécurité. Dans une publication récente célébrant l'achèvement d'une politique de sécurité interne, il a joint une photo de son bureau.

En arrière-plan, parmi plusieurs post-its leurres, un post-it jaune contient des informations confidentielles — visibles par n'importe quel observateur attentif qui zoome sur la photo.

Objectif

1. Retrouver le profil LinkedIn d'Alexandre Mercier.
2. Identifier la publication contenant une photo de bureau.
3. Analyser attentivement la photo et zoomer sur les post-its.
4. Lire le flag dissimulé dans l'image.

Informations connues

- Nom : Alexandre Mercier
- Entreprise : TechCorp Industries
- Poste : RSSI (Responsable de la Sécurité des Systèmes d'Information)
- Localisation : Lyon, France
- Format du flag : CTF_INSA_STI_FACILE_05_1/1_<...>

3 - Outils recommandés

- LinkedIn (recherche de profil)
- Google (Google Dorks)
- Navigateur Web avec zoom image
- Google Images (recherche inversée)

4 - Indices

- Utilisez LinkedIn pour chercher le profil. Essayez différentes combinaisons de recherche.
- Regardez attentivement les photos de ses publications récentes.
- Zoomez sur les post-its visibles dans la photo de bureau.
- Tous les post-its ne sont pas des leurres.

Automatisation de Challenge-Response

Nom : Automatisation de Challenge-Response

Catégorie : Scripting / Programmation **Niveau :** Moyen

Code : ctf-02-07 **Tags :** python socket tcp automation regex

1 - Vulnérabilité Exploitée

La vulnérabilité réside dans une **conception de sécurité reposant uniquement sur la vitesse d'exécution humaine**. Le serveur suppose qu'un utilisateur ne peut pas résoudre 5 calculs complexes en moins de 30 secondes. Cependant, comme le protocole est basé sur du texte clair (TCP Cleartext) et que les questions sont prévisibles, le système est vulnérable à une attaque par **automatisation (scripting)**.

2 - Scénario

Le système d'authentification de TechCorp impose une preuve de rapidité. Le serveur envoie successivement 5 expressions mathématiques (incluant des parenthèses et des priorités opératoires) que l'utilisateur doit résoudre.

Objectif

Développer un script capable d'interagir avec le serveur, d'extraire l'opération, de la calculer et de renvoyer la réponse avant l'expiration du délai de 30 secondes.

Accès

- Connexion : `nc opnsense-ctf-bacpro.cyberfarm.insa-cvl.fr 1036`
- Format du flag : {CTF_INSA_STI_FACILE_07_<...>}

3 - Outils recommandés

- **Python 3** : Idéal pour le prototypage rapide de scripts réseau.
- **Module `socket`** : Pour la gestion de la connexion TCP bidirectionnelle.
- **Module `re`** : Pour utiliser les expressions régulières afin d'isoler l'opération mathématique dans le flux de texte.
- **Fonction `eval()`** : Pour interpréter la chaîne de caractères extraite comme une opération mathématique réelle.

4 - Indices

1. Les expressions varient en complexité (ex : `((11 + 19) * 17) - 11`). Une simple extraction de deux chiffres ne suffit pas.
2. Le serveur attend une réponse suivie d'un caractère de fin de ligne (`\n`).

3. La rapidité est la clé : seule une machine peut garantir le succès sur les 5 étapes consécutives.

Analyse de Logs Windows

Nom :	Analyse de Logs Windows		
Catégorie :	Forensique	**Niveau :**	Facile
Code :	ctf-02-08	**Tags :**	forensique, windows, evtx, logs, password, debutant

1 - Vulnérabilité Exploitée

Exposition de credentials en clair dans les journaux d'événements Windows (CWE-532 – Insertion of Sensitive Information into Log File).

Lorsqu'un utilisateur tape son mot de passe dans le champ *nom d'utilisateur* d'une interface d'authentification, le système enregistre cette saisie erronée dans les journaux d'événements. Si l'application de journalisation enregistre le nom d'utilisateur soumis sans filtrage, le mot de passe se retrouve stocké en clair dans les logs – accessible à tout analyste ayant accès aux fichiers `.evtx`.

2 - Scénario

Un employé jure ne pas avoir divulgué son mot de passe. Pourtant son compte a été compromis. L'analyse des journaux d'événements Windows de sa machine révèle qu'il a accidentellement tapé son mot de passe dans le champ username lors d'une tentative d'authentification. L'application `TechCorpAuth` a enregistré fidèlement la saisie – exposant le mot de passe en clair dans le journal applicatif.

Objectif

1. Analyser les fichiers de journaux Windows (`.evtx`) fournis.
2. Identifier l'événement d'échec d'authentification contenant le mot de passe.
3. Extraire les informations clés : EventID, username, mot de passe exposé.

Accès

- **Fichier fourni :** `windows_logs.zip`
- **Contenu :** 4 fichiers `.evtx` (`Log_Erreur.evtx`, `Log_Erreur_2.evtx`, `Log_Erreur_3.evtx`, `Log_Erreur_4.evtx`
- **Machine source :** `DESKTOP-EMTF8D7`
- **Application :** `TechCorpAuth` (journal applicatif)
- Format du flag : `CTF_INSA_STI_FACILE_8_1/1_<...>`

3 - Outils recommandés

- **Windows Event Viewer** – outil natif Windows pour lire les fichiers `.evtx` avec interface graphique
- **FullEventLogView** (NirSoft) – alternative légère pour lire les `.evtx` sous Windows
- `strings -e l` – extraction de chaînes UTF-16LE depuis un fichier binaire (Linux)
- `python-evtx` – bibliothèque Python pour parser les fichiers `.evtx` et extraire le XML complet
- `unzip` – extraction de l'archive fournie

4 - Indices

- L'énoncé dit *"taper son mot de passe au mauvais endroit"* – erreur classique où l'utilisateur saisit son mot de passe dans le champ *nom d'utilisateur*. Le système enregistre alors le mot de passe comme tentative de connexion avec un nom d'utilisateur invalide.
- Les fichiers `.evtx` sont des journaux d'événements Windows au format binaire – ils nécessitent soit **Windows Event Viewer**, soit des outils d'extraction (strings, python-evtx) pour être lus sous Linux.
- Le format EVTX utilise l'encodage **UTF-16LE** pour les chaînes de caractères – la commande `strings` doit être utilisée avec l'option `-e l` pour extraire correctement le texte.
- Parmi les 4 fichiers de logs, seul `Log_Erreur_2.evtx` provient de l'application `TechCorpAuth` – c'est là que se trouve l'événement d'authentification.
- L'**EventID 1337** est un clin d'œil classique dans les CTF (*leet speak* pour *leet/elite*) – sa présence dans un événement de sécurité est un indicateur fort d'un log intentionnellement créé pour le challenge.

PDF Texte Transparent

Nom :	PDF Texte Transparent		
Catégorie :	Stéganographie	**Niveau :**	Facile
Code :	ctf-02-10	**Tags :**	steganographie, pdf, texte, forensic, transparent

1 - Vulnérabilité Exploitée

Dissimulation d'informations par texte transparent dans un PDF (CWE-312 – Cleartext Storage of Sensitive Information).

Le format PDF permet de définir la couleur du texte indépendamment de la couleur du fond de page. Un texte blanc sur fond blanc est visuellement invisible à l'écran, mais reste présent dans le flux de données du fichier et est extrait par tout outil d'analyse. Cette technique de stéganographie triviale est souvent utilisée pour dissimuler des informations sensibles dans des documents d'apparence anodine.

2 - Scénario

Un rapport annuel de *TechCorp Industries* semble ne contenir que des informations financières banales. L'indice *"peut-être est-il sous vos yeux depuis le début"* suggère que le flag est présent dans le document mais rendu invisible par sa couleur. En utilisant des outils d'extraction de texte PDF, le contenu caché est immédiatement révélé.

Objectif

1. Analyser les métadonnées du fichier PDF.
2. Extraire le texte brut du PDF pour révéler le contenu invisible.
3. Récupérer le flag caché.

Accès

- **Fichier fourni :** `rapport_annuel.pdf`
- **Taille :** 2.5 kB
- **Producteur :** ReportLab PDF Library (opensource)
- **Auteur :** anonymous
- Format du flag : CTF_INSA_STI_FACILE_10_<...>

3 - Outils recommandés

- `exiftool` – extraction des métadonnées du fichier PDF
- `pdftotext` – extraction du texte brut depuis un PDF, indépendamment de la couleur du texte
- `strings` – extraction de toutes les chaînes de caractères lisibles dans un fichier binaire
- **Lecteur PDF** – la sélection `Ctrl+A` sélectionne tout le texte y compris le texte transparent

4 - Indices

- L'énoncé dit *"peut-être est-il sous vos yeux depuis le début"* – le flag est littéralement dans le document, mais invisible visuellement.
- La technique classique de stéganographie PDF consiste à écrire du texte en **couleur blanche sur fond blanc** – invisible à l'écran mais présent dans les données du fichier.
- `pdftotext` extrait le texte brut d'un PDF sans tenir compte des couleurs – tout texte présent dans le fichier, quelle que soit sa couleur, est affiché.
- Dans un lecteur PDF, **Ctrl+A** sélectionne tout le texte de la page, y compris le texte transparent – il apparaît alors en surbrillance.
- `exiftool` permet de vérifier les métadonnées (auteur, date, producteur) – informations utiles pour l'analyse forensique mais ne contenant pas le flag ici.

Metasploit - Étape 1 : Service Vulnérable

Nom : Metasploit - Étape 1 : Service Vulnérable

Catégorie : Système

Niveau : Moyen

Code : ctf-02-11

Tags : samba, exploit, rce, metasploit

1 - Vulnérabilité Exploitée

Exploitation de la vulnérabilité CVE-2017-7494, connue sous le nom de SambaCry.

Cette faille affecte certaines versions de Samba (dont la 4.5.9) et permet le chargement distant d'un module malveillant via un partage SMB inscriptible, conduisant à une exécution de code à distance (RCE).

2 - Scénario

Un audit interne de l'entreprise *TechCorp* a révélé la présence d'un service vulnérable exposé sur le réseau interne.

Votre mission consiste à :

- Identifier le service exposé.
- Déterminer sa version exacte.
- Rechercher une vulnérabilité connue.
- Exploiter la machine.
- Récupérer les flags.

Objectif

1. Scanner la machine cible.
2. Identifier le service Samba.
3. Vérifier la version installée.
4. Énumérer les partages accessibles.
5. Exploiter la vulnérabilité SambaCry.
6. Obtenir un accès distant.

Accès

- Connexion SSH :

```
ssh ctf@opnsense-ctf-bacpro.cyberfarm.insa-cvl.fr -o PubkeyAuthentication=no -p 1000
```

- Mot de passe :

```
46292ca6-b2e8-48cc-bcaf-b185b5fb7cc0
```

- Cible : **10.0.0.40**
- Identifiants SMB :
  - Utilisateur : `sambacry`
  - Mot de passe : `nosambanocry`

3 - Outils recommandés

- Nmap
- smbclient
- Metasploit Framework
- Terminal Linux

4 - Indices

- Scanner tous les ports ouverts.
- Identifier la version précise du service Samba.
- Vérifier si un partage est inscriptible.
- Rechercher les CVE associées à la version détectée.

Metasploit – Étape 2 : Accès

Nom :	Metasploit - Étape 2 : Accès		
Catégorie :	Système	**Niveau :**	Moyen
Code :	ctf-02-12	**Tags :**	metasploit, exploit, rce, samba

1 - Vulnérabilité Exploitée

Exploitation de la vulnérabilité CVE-2017-7494 (SambaCry) via le module Metasploit approprié.

Cette faille permet une exécution de code à distance (Remote Code Execution) à travers un partage SMB inscriptible.

2 - Scénario

Lors du challenge précédent (ctf-02-11), vous avez :

- Identifié le service Samba vulnérable.
- Détecté la version 4.5.9.
- Confirmé la présence de la vulnérabilité CVE-2017-7494.

Votre mission consiste maintenant à exploiter cette vulnérabilité afin d'obtenir un accès distant à la machine cible.

Objectif

1. Lancer Metasploit Framework.
2. Charger le module d'exploitation Samba.
3. Configurer correctement les paramètres.
4. Obtenir une session Meterpreter.

Accès

Connexion à l'environnement :

```
ssh ctf@opnsense-ctf-bacpro.cyberfarm.insa-cvl.fr -o PubkeyAuthentication=no -p 1000
```

Mot de passe :

```
46292ca6-b2e8-48cc-bcaf-b185b5fb7cc0
```

Cible :

10.0.0.40

Identifiants SMB :

- user : `sambacry`
- pass : `nosambanocry`

3 - Outils recommandés

- Metasploit Framework
- Nmap (vérification préalable)
- Terminal Linux

4 - Indices

- Le module Metasploit cible la vulnérabilité CVE-2017-7494.
- Vérifier le nom exact du partage SMB.
- Configurer correctement LHOST et LPORT.
- Sélectionner la bonne architecture cible.

JWT – Étape 1 : Secret

Nom :	`JWT - Étape 1 : Secret`		
Catégorie :	`Web`	**Niveau :**	`Moyen`
Code :	`ctf-02-18`	**Tags :**	`jwt` `cryptographie` `brute-force` `authentification`

1 - Vulnérabilité Exploitée

Faible clé secrète HMAC dans un token JWT (CWE-326 – Inadequate Encryption Strength).

Les tokens JWT signés avec l'algorithme `HS256` reposent sur une clé secrète partagée. Si cette clé est trop courte ou prévisible, un attaquant peut la retrouver par force brute et forger des tokens arbitraires, compromettant ainsi l'intégralité du système d'authentification.

2 - Scénario

L'API REST de *TechCorp* utilise des tokens JWT pour authentifier ses utilisateurs. Un compte invité est accessible publiquement (`guest/guest123`).

En analysant le token émis par l'API, on constate qu'il est signé avec `HS256` et une clé secrète. Cette clé, choisie par le développeur, est suffisamment faible pour être retrouvée par dictionnaire.

Votre mission consiste à :

- S'authentifier avec le compte `guest` pour obtenir un token JWT valide.
- Analyser la structure du token.
- Bruteforcer la clé secrète à l'aide d'une wordlist fournie.
- Soumettre la clé retrouvée comme flag.

Objectif

1. Obtenir un token JWT via l'endpoint de login.
2. Identifier l'algorithme de signature utilisé.
3. Lancer une attaque par dictionnaire sur la clé secrète HMAC.
4. Valider la clé trouvée.

Accès

- **URL cible :** `http://opnsense-ctf-bacpro.cyberfarm.insa-cvl.fr:1047/api`
- **Identifiants :** `guest` / `guest123`
- **Wordlist :** fournie avec le challenge

3 - Outils recommandés

- `curl` – envoi de requêtes HTTP vers l'API

- `john` (John the Ripper) – bruteforce de clé HMAC-SHA256
- `hashcat` – alternative GPU pour le bruteforce JWT
- `jwt.io` – décodage et visualisation de la structure d'un token JWT
- `python3 + pyjwt` – forge et vérification de tokens JWT

4 - Indices

- Un token JWT est composé de trois parties séparées par des points : `header.payload.signature`. Chaque partie est encodée en Base64URL.
- L'algorithme `HS256` signe le token avec `HMAC-SHA256` en utilisant une clé secrète symétrique. Contrairement à `RS256`, cette clé doit rester secrète côté serveur.
- Si la clé est un mot courant ou une combinaison simple, une attaque par dictionnaire suffit.
- John the Ripper supporte nativement le format JWT avec le mode `HMAC-SHA256`.

JWT – Étape 2 : Admin

Nom :	`JWT - Étape 2 : Admin`		
Catégorie :	`Web`	**Niveau :**	`Moyen`
Code :	`ctf-02-19`	**Tags :**	`jwt` `token-forging` `privilege-escalation`

1 - Vulnérabilité Exploitée

Forgery de token JWT pour élévation de privilège (CWE-287 – Improper Authentication).

Le serveur se repose uniquement sur le contenu du token JWT pour déterminer les rôles. La connaissance de la clé secrète HMAC utilisée pour signer les tokens permet de générer un token arbitraire et de s'octroyer un rôle `admin`, contournant ainsi tout contrôle d'accès.

2 - Scénario

Après avoir découvert la clé secrète JWT (`techcorp2024`) à l'étape précédente, il est possible de :

- Modifier le payload du token pour changer le rôle utilisateur en `admin`
- Signer le nouveau token avec la clé secrète
- Accéder à l'endpoint protégé `/api/admin` pour récupérer le flag

Aucune vérification supplémentaire côté serveur n'empêche cette élévation de privilège.

Objectif

1. Utiliser la clé secrète JWT pour forger un token valide pour `admin`
2. Inclure dans le payload le rôle : `role: admin`
3. Accéder à l'endpoint `/api/admin` et récupérer le flag

Accès

- **URL cible :** `http://opnsense-ctf-bacpro.cyberfarm.insa-cvl.fr:1047/api/admin`
- **Token JWT :** généré avec la clé secrète `techcorp2024`
- **Payload requis :** `"role": "admin"`

3 - Outils recommandés

- `python3 + PyJWT` – génération et signature de tokens JWT
- `jwt.io` – visualisation et débogage du token
- $\texttt{jwt}_t ool - -alternative pour l'analyse et la forge de JWT$
- `curl - validation de l'accès aux endpoints protégés`

4 - Indices

- La signature HMAC du token repose sur une clé symétrique connue
- Le champ `role` dans le payload détermine les privilèges
- Le serveur ne valide pas la correspondance entre le rôle et l'utilisateur réel
- L'attaque consiste à modifier le payload puis signer le token avec la clé secrète

TechCorp - Étape 1 : Point d'Entrée

Nom : TechCorp - Étape 1 : Point d'Entrée

Catégorie : Web

Niveau : Difficile

Code : ctf-02-20

Tags : auth, http, recon, ssh, bruteforce, credentials

1 - Vulnérabilité Exploitée

Absence de restriction sur l'upload de fichiers côté serveur. L'application web accepte des fichiers PHP sans validation d'extension ni de contenu, permettant le dépôt d'un webshell et l'exécution de commandes système arbitraires (*Remote Code Execution*).

2 - Scénario

TechCorp Industries héberge un serveur web en DMZ exposé sur Internet. Ce serveur propose une fonctionnalité d'upload de fichiers sans contrôle adéquat. En tant que pentester, votre première mission est d'établir un accès initial sur ce serveur en exploitant cette faille.

Objectif

1. Scanner la cible pour identifier les services exposés.
2. Identifier et exploiter la fonctionnalité d'upload vulnérable.
3. Déposer un webshell PHP et établir un reverse shell stable.
4. Récupérer le premier flag.

Accès

- Machine Bastion SSH :
  - `ssh ctf@opnsense-ctf-bacpro.cyberfarm.insa-cvl.fr -o PubkeyAuthentication=no -p 1000`
  - Mot de passe : `46292ca6-b2e8-48cc-bcaf-b185b5fb7cc0`
- Pour accéder au réseau DMZ (192.168.100.15) depuis votre machine :
  - `sshuttle -disable-ipv6 -r ctf@opnsense-ctf-bacpro.cyberfarm.insa-cvl.fr:1000 "192.168.100.0/24" -ssh-cmd "ssh -o PubkeyAuthentication=no"`
  - Mot de passe : `46292ca6-b2e8-48cc-bcaf-b185b5fb7cc0`
- Cible DMZ : `192.168.100.15`
- Format du flag : CTF_INSA_STI_DIFFICILE_01_1/7_<...>

3 - Outils recommandés

- `nmap` (scan de ports et services)
- `curl` / Navigateur Web

- `nc` (Netcat, écoute reverse shell)
- Webshell PHP

4 - Indices

- Scannez tous les ports du serveur DMZ, pas seulement les ports standard.
- Le serveur web accepte des uploads de fichiers — regardez les restrictions appliquées.
- Un shell PHP minimal : `<?php system($_GET['c'] ?? 'id'); ?>`
- Cherchez le flag dans `/var/www/html/` après avoir obtenu un shell.

TechCorp - Étape 2 : Root DMZ

Nom :	TechCorp - Étape 2 : Root DMZ		
Catégorie :	Système	**Niveau :**	Difficile
Code :	ctf-02-21	**Tags :**	sudo privesc filesystem linux

1 - Vulnérabilité Exploitée

Permission `sudo` mal configurée sur le binaire `/usr/bin/find`. Ce binaire, autorisé sans restriction de commande via `sudo`, peut être utilisé pour exécuter un shell avec les privilèges root grâce à son option `-exec` — technique documentée sur GTFOBins.

2 - Scénario

Vous disposez d'un accès initial sur le serveur DMZ en tant que `www-data` via le webshell obtenu à l'étape précédente. Pour progresser dans l'infrastructure, vous devez escalader vos privilèges et devenir **root** afin d'accéder aux ressources protégées du système.

Objectif

1. Stabiliser le shell obtenu via le webshell.
2. Énumérer les permissions sudo disponibles.
3. Exploiter le binaire `find` pour obtenir un shell root.
4. Récupérer le deuxième flag.

Accès

- Shell initial via webshell sur `192.168.100.15:8015`
- Utilisateur courant : `www-data`
- Format du flag : CTF_INSA_STI_DIFFICILE_01_2/7_<...>

3 - Outils recommandés

- `python3` (stabilisation du shell)
- `sudo -l` (énumération des permissions)
- GTFOBins (`https://gtfobins.github.io`) (référence d'exploitation de binaires)

4 - Indices

- Stabilisez votre shell avant tout : un shell instable rend l'exploitation difficile.
- Vérifiez les permissions sudo avec `sudo -l` — certains binaires sont autorisés par erreur.
- Le binaire `/usr/bin/find` dispose de permissions sudo intéressantes. Consultez GTFOBins.
- L'option `-exec` de `find` permet d'exécuter une commande arbitraire.

TechCorp - Étape 3 : Pivot SSH

Nom : TechCorp - Étape 3 : Pivot SSH

Catégorie : Réseau

Niveau : Difficile

Code : ctf-02-22

Tags : sniffing, ssh, bruteforce, enumeration, services, credentials

1 - Vulnérabilité Exploitée

Stockage de credentials SSH en clair dans un fichier texte accessible depuis le compte root de la DMZ. Ces identifiants permettent de pivoter vers le réseau interne depuis le serveur compromis, franchissant ainsi le second périmètre de sécurité.

2 - Scénario

Vous êtes root sur le serveur DMZ. Pour progresser vers le réseau interne de TechCorp, vous devez trouver des credentials permettant une connexion SSH vers une machine interne. L'administrateur a malencontreusement laissé ces identifiants en clair dans le répertoire root.

Objectif

1. Explorer le répertoire `/root/` à la recherche de fichiers de credentials.
2. Identifier les identifiants SSH pour le réseau interne.
3. Se connecter en SSH à la machine pivot interne.
4. Récupérer le troisième flag.

Accès

- Accès root sur serveur DMZ `192.168.100.15`
- Réseau interne cible : `10.0.1.0/24`
- Format du flag : CTF_INSA_STI_DIFFICILE_01_3/7_<...>

3 - Outils recommandés

- Commandes Linux : `ls`, `cat`, `find`, `grep`
- Client `ssh`

4 - Indices

- Cherchez dans `/root/` des fichiers contenant les mots `creds`, `password` ou `internal`.
- Le fichier de credentials s'appelle `creds_internal.txt`.
- Les credentials permettent un accès SSH vers `10.0.1.15`.
- Le flag se trouve dans `/opt/techcorp/pivot/` sur la machine interne.

TechCorp - Étape 4 : Compromission SMB

Nom :	TechCorp - Étape 4 : Compromission SMB		
Catégorie :	Réseau	**Niveau :**	Difficile
Code :	ctf-02-23	**Tags :**	sniffing, smb, permissions, suid, privesc, enumeration

1 - Vulnérabilité Exploitée

Partage SMB local avec permissions d'écriture couplé à un cronjob root exécutant automatiquement les scripts déposés dans le répertoire surveillé. Cette combinaison permet à un attaquant de faire exécuter du code arbitraire avec les privilèges root en déposant un script malveillant dans le partage.

2 - Scénario

Sur la machine pivot interne, vous découvrez un partage SMB local dont le répertoire `inbox` est surveillé par un processus automatique tournant en root. En exploitant les droits d'écriture sur ce partage, vous pouvez faire créer un binaire SUID root et ainsi obtenir une élévation de privilèges complète.

Objectif

1. Énumérer les partages SMB locaux de la machine pivot.
2. Identifier le répertoire surveillé par le cronjob root.
3. Créer et déposer un script d'élévation de privilèges (SUID bash).
4. Attendre l'exécution automatique et exploiter le SUID créé.
5. Récupérer le quatrième flag.

Accès

- Accès SSH sur machine pivot interne (`10.0.1.15`, utilisateur `pivot`)
- Format du flag : CTF_INSA_STI_DIFFICILE_01_4/7_<...>

3 - Outils recommandés

- `smbclient` (énumération et accès aux partages SMB)
- `nc` / `bash` (création du script)
- Commandes Linux : `ls -l`, `id`

4 - Indices

- Listez les partages SMB locaux : `smbclient -L //127.0.0.1 -U pivot`
- Le partage `devshare/inbox` est surveillé par un cronjob root qui exécute les scripts déposés.
- Créez un script qui copie `/bin/bash` avec le bit SUID : `cp /bin/bash /tmp/rootbash && chmod u+s /tmp/rootbash`

- Attendez environ 20 secondes puis vérifiez : `ls -l /tmp/rootbash`

TechCorp - Étape 5 : Privesc Root Interne

Nom :	TechCorp - Étape 5 : Privesc Root Interne		
Catégorie :	Système	**Niveau :**	Difficile
Code :	ctf-02-24	**Tags :**	sqli, injection, database, permissions, sudo, privesc

1 - Vulnérabilité Exploitée

Exploitation du bit SUID positionné sur une copie de `/bin/bash` créée lors de l'étape précédente. L'option `-p` de bash préserve l'identité effective (`euid`) du propriétaire du binaire (root), accordant un accès root complet sans authentification.

2 - Scénario

À l'étape précédente, le cronjob root a exécuté votre script et créé `/tmp/rootbash` avec le bit SUID. Il ne reste plus qu'à exploiter ce binaire pour obtenir un shell root complet et récupérer le flag dans `/root/`.

Objectif

1. Vérifier la présence et les permissions du binaire SUID créé.
2. Exécuter `/tmp/rootbash -p` pour obtenir un shell root.
3. Récupérer le cinquième flag dans `/root/`.

Accès

- Binaire SUID disponible : `/tmp/rootbash`
- Machine pivot interne `10.0.1.15`
- Format du flag : CTF_INSA_STI_DIFFICILE_01_5/7_<...>

3 - Outils recommandés

- Commandes Linux : `ls -l`, `id`, `cat`
- Optionnel : `sudo python3` (méthode alternative)

4 - Indices

- Vérifiez que le bit SUID est bien positionné : `ls -l /tmp/rootbash` doit afficher `-rwsr-xr-x`.
- Utilisez l'option `-p` : elle indique à bash de ne pas réinitialiser l'euid au lancement.
- Une fois root, le flag se trouve dans `/root/flag5.txt`.

TechCorp - Étape 6 : Accès Ops via SQLite

Nom :	TechCorp - Étape 6 : Accès Ops via SQLite		
Catégorie :	Système	**Niveau :**	Difficile
Code :	ctf-02-25	**Tags :**	suid privesc filesystem linux

1 - Vulnérabilité Exploitée

Deux failles enchaînées menant à une élévation de privilèges sur une nouvelle machine :

- **Credentials en clair dans l'historique bash** : le fichier `.bash_history` de l'utilisateur `dev` contient une commande SSH avec mot de passe, permettant un mouvement latéral vers la machine `ops`.
- **sudo sqlite3 sans restriction** : le binaire `sqlite3` est autorisé via sudo avec la directive `.shell`, permettant l'exécution de commandes arbitraires avec les privilèges root.

2 - Scénario

Vous êtes root sur la machine pivot interne. Pour progresser vers les systèmes de gestion, vous devez effectuer un mouvement latéral vers une nouvelle machine. L'historique bash d'un utilisateur trahit des credentials SSH. Sur cette nouvelle machine, `sqlite3` est accessible via sudo — un vecteur d'élévation documenté sur GTFOBins.

Objectif

1. Explorer l'historique bash des utilisateurs pour trouver des credentials.
2. Se connecter en SSH à la machine `ops` avec les identifiants trouvés.
3. Identifier les permissions sudo disponibles.
4. Exploiter `sqlite3` via la directive `.shell` pour lire le flag root.

Accès

- Accès root sur machine pivot interne `10.0.1.15`
- Machine cible : `10.0.2.15` (ops)
- Format du flag : CTF_INSA_STI_DIFFICILE_01_6/7_<...>

3 - Outils recommandés

- Commandes Linux : `cat`, `ls -la`, `sudo -l`
- Client `ssh`
- `sqlite3` (interactif)
- GTFOBins (`https://gtfobins.github.io/gtfobins/sqlite3/`)

4 - Indices

- Vérifiez l'historique bash du compte `dev` : `cat /home/dev/.bash_history`
- Les credentials SSH pour `ops` se trouvent dans cet historique.
- Sur la machine `ops`, `sqlite3` a des permissions sudo.
- La directive `.shell` de sqlite3 permet d'exécuter des commandes système directement.

TechCorp - Étape 7 : Exfiltration et Crypto

Nom :	TechCorp - Étape 7 : Exfiltration et Crypto		
Catégorie :	Cryptographie	**Niveau :**	Difficile
Code :	ctf-02-26	**Tags :**	sqli, injection, database, cracking, decoding, crypto

1 - Vulnérabilité Exploitée

Passphrase de chiffrement stockée en clair dans une base SQLite accessible via `sudo sqlite3`. Un fichier chiffré en AES-256-CBC est stocké sur le système et déchiffrable avec cette passphrase, exposant le flag final de la compromission.

2 - Scénario

Vous avez accès à la base SQLite `techcorp_vault.db` avec les privilèges root via `sudo sqlite3`. Cette base contient une table `secrets` regroupant des clés sensibles, dont la passphrase utilisée pour chiffrer un fichier d'exfiltration stocké sur le système. Extrayez la passphrase et déchiffrez le fichier pour obtenir le flag final.

Objectif

1. Accéder à la base SQLite et lister ses tables.
2. Extraire la passphrase de chiffrement depuis la table `secrets`.
3. Localiser le fichier chiffré sur le système.
4. Déchiffrer le fichier avec OpenSSL et récupérer le flag final.

Accès

- Accès `sudo sqlite3` sur machine `ops` (`10.0.2.15`)
- Base de données : `/var/lib/techcorp_vault.db`
- Format du flag : {CTF_INSA_STI_DIFFICILE_01_7/7_<...>}

3 - Outils recommandés

- `sqlite3` (requêtes SQL, directive `.shell`)
- `openssl enc` (déchiffrement AES-256-CBC)
- Commandes Linux : `ls`, `cat`

4 - Indices

- Dans sqlite3, listez les tables avec `.tables`.
- La table `secrets` contient une colonne `k` (clé) et une colonne `v` (valeur).

- Cherchez la clé `final_passphrase` : `SELECT v FROM secrets WHERE k='final_passphrase';`
- Le fichier chiffré se trouve dans `/opt/techcorp/exfil/` — utilisez `.shell` pour le localiser.
- Déchiffrez avec : `openssl enc -d -aes-256-cbc -pbkdf2 -iter 200000 -pass pass:'PASSPHRASE' -in data.enc`

Portail d'Administration

Nom : `Portail d'Administration`

Catégorie : `Web` **Niveau :** `Facile`

Code : `ctf-03-05` **Tags :** `sqli` `blind` `authentication` `bypass`

1 - Vulnérabilité Exploitée

Blind SQL Injection sur formulaire d'authentification (CWE-89 – Improper Neutralization of Special Elements used in an SQL Command).

Le formulaire de connexion construit dynamiquement une requête SQL en concaténant directement les entrées utilisateur sans les filtrer ni les paramétrer. Cette faille permet à un attaquant d'injecter du code SQL arbitraire dans la requête, et ainsi d'extraire des données de la base sans jamais recevoir de message d'erreur explicite (*blind injection*).

2 - Scénario

L'entreprise *SecureAdmin* expose un portail d'administration interne. Les développeurs, convaincus que l'obscurité du code source constitue une protection suffisante, n'ont pas sécurisé les entrées du formulaire de connexion.

Lors d'un audit, on observe que le formulaire réagit de manière anormale à certains caractères spéciaux (erreur 500, délai de réponse), ce qui trahit une vulnérabilité SQLi. La base de données contient les credentials d'administration, récupérables par injection aveugle.

- Le serveur ne renvoie jamais les données SQL directement – c'est une **blind injection**.
- Deux techniques d'injection aveugle sont exploitables : **boolean-based** et **time-based**.
- L'extraction des données se fait caractère par caractère, de manière automatisée.

Objectif

1. Identifier la vulnérabilité SQLi sur le champ `username`.
2. Exploiter la blind injection pour extraire les tables et leurs contenus.
3. Récupérer les credentials administrateur et se connecter au portail.
4. Obtenir le flag affiché sur la page d'administration.

Accès

- **URL cible :** `http://opnsense-ctf-bacpro.cyberfarm.insa-cvl.fr:1055/`
- **Accès initial :** aucun – portail public
- **SGBD :** MySQL $\geq$ 5.0.12 (MariaDB)
- Format du flag : {CTF_INSA_STI_FACILE_5_<...>}

3 - Outils recommandés

- `curl` – test manuel des réponses HTTP et identification du comportement anormal
- `sqlmap` – exploitation automatisée de la blind SQL injection
- Un navigateur web – soumission du formulaire avec les credentials extraits

4 - Indices

- Le formulaire réagit bizarrement aux caractères spéciaux : une apostrophe `'` dans le champ `username` provoque une erreur 500 – signe classique d'une injection SQL non protégée.
- Tester une payload manuelle simple dans le champ `username` : `admin' OR '1'='1` – si le portail donne accès, l'injection est confirmée.
- La structure de la payload `OR '1'='1` exploite une condition toujours vraie pour court-circuiter la vérification du mot de passe.
- Le serveur redirige vers `admin.php` en cas de succès (HTTP 302) – sqlmap exploite cette différence de comportement pour ses tests booléens.
- Aucun message d'erreur SQL n'est affiché – c'est une **blind injection** ; pour extraire les données de la base (credentials, tables), un outil automatisé comme sqlmap est nécessaire.
- Le paramètre vulnérable est `username` dans la requête POST.

Vim Sudo

Nom :	`Vim Sudo`		
Catégorie :	`Système`	**Niveau :**	`Facile`
Code :	`ctf-03-07`	**Tags :**	`ssh` `bruteforce` `sudo` `privesc` `filesystem` `Linux`

1 - Vulnérabilité Exploitée

Mauvaise configuration de sudo permettant à un utilisateur d'exécuter `vim` avec des privilèges élevés.

Vim permet d'exécuter des commandes système via :

- : !command
- :shell

Si vim est autorisé via sudo sans restriction, cela permet une élévation de privilèges.

2 - Scénario

Vous disposez d'un accès SSH légitime avec l'utilisateur :

user1

Un autre utilisateur, user2, possède un fichier confidentiel (le flag) dans son répertoire personnel.

L'administrateur a accordé certains droits sudo à user1, probablement de manière trop permissive.

Objectif

- Identifier les droits sudo de user1
- Exploiter une mauvaise configuration
- Lire le fichier confidentiel de user2
- Récupérer le flag

Accès

- Connexion SSH : `ssh user1@opnsense-ctf-bacpro.cyberfarm.insa-cvl.fr -o PubkeyAuthentication=n -p 1057`
- Identifiants : `user1` / `user1`
- Format du flag : {CTF_INSA_STI_FACILE_7_<...>}

3 - Outils recommandés

- sudo
- vim
- GTFOBins (référence en ligne)

4 - Indices

- Vérifiez les permissions sudo.
- Certains éditeurs permettent d'exécuter des commandes système.
- Consultez GTFOBins.

Project Nebula

Nom : Project Nebula
Catégorie : Réseau
Code : ctf-03-08
Niveau : Moyen
Tags : recon, sniffing, ssh, bruteforce, dns, enumeration

1 - Vulnérabilité Exploitée

Deux vulnérabilités enchaînées menant à une exécution de commande à distance (RCE) :

- **Transfert de zone DNS non restreint (AXFR)** : le serveur DNS répond aux requêtes de transfert de zone depuis n'importe quelle source, exposant l'intégralité de l'infrastructure interne.
- **Injection de commande (Command Injection)** : l'outil de diagnostic web transmet l'entrée utilisateur sans validation à une commande système, permettant l'exécution de commandes arbitraires sur le serveur.

2 - Scénario

GlobalTech développe une API interne secrète hébergée sur un serveur de développement faisant également office de serveur DNS local. Le serveur est mal configuré et répond aux requêtes DNS depuis n'importe quelle source.

En tant qu'auditeur externe, l'objectif est d'explorer l'infrastructure DNS pour découvrir des sous-domaines cachés, accéder à l'outil de diagnostic interne, et exploiter une injection de commande pour lire le flag sur le serveur.

Objectif

1. Scanner les ports ouverts et identifier les services exposés.
2. Effectuer un transfert de zone DNS pour découvrir l'infrastructure interne.
3. Configurer l'accès au Virtual Host découvert.
4. Exploiter l'injection de commande dans l'outil de diagnostic.
5. Lire le flag sur le serveur.

Accès

- Cible : `10.0.0.58` accessible via :
  - `ssh ctf@opnsense-ctf-bacpro.cyberfarm.insa-cvl.fr -o PubkeyAuthentication=no -p 1000`
  - MDP : `46292ca6-b2e8-48cc-bcaf-b185b5fb7cc0`
- Web `http://opnsense-ctf-bacpro.cyberfarm.insa-cvl.fr:1058`
- Domaine connu : `globaltech.local`
- Format du flag : {CTF_INSA_STI_FACILE_8_<...>}

3 - Outils recommandés

- `nmap` (scan de ports)
- `dig` (requêtes DNS et transfert de zone)
- `/etc/hosts` (résolution DNS locale)
- Navigateur Web

4 - Indices

- Le site accessible via l'IP affiche une page de maintenance : un Virtual Host est probablement en place. Le vrai site n'est accessible que par son nom de domaine.
- Le port 53 est ouvert. Essayez de demander au serveur la liste complète de ses enregistrements DNS (*Zone Transfer* / AXFR).
- Une fois le sous-domaine trouvé, modifiez `/etc/hosts` pour forcer la résolution locale.
- L'outil de diagnostic exécute une commande système sur votre saisie. Que se passe-t-il si vous ajoutez `;` suivi d'une commande Linux ?

L'Infiltration Arkhos

Nom : L'Infiltration Arkhos

Catégorie : Réseau

Niveau : Facile

Code : ctf-03-09

Tags : sniffing, ftp, ssh, bruteforce, enumeration, services

1 - Vulnérabilité Exploitée

Mauvaise configuration d'un service FTP exposant le dossier `.ssh` d'un utilisateur comme répertoire racine. Un attaquant disposant des identifiants FTP peut exfiltrer la clé privée SSH de l'utilisateur et obtenir un accès shell complet sans connaître son mot de passe SSH.

2 - Scénario

Arkhos Tech vient de déployer un nouveau serveur de fichiers pour ses développeurs. L'administrateur a laissé des services actifs sur des ports non conventionnels et a mal configuré les droits d'accès du service FTP. En particulier, le répertoire racine du FTP de l'utilisateur `arkhos` correspond à son dossier `.ssh`, exposant sa clé privée à quiconque se connecte au service.

Objectif

1. Scanner l'intégralité des ports TCP et identifier les services exposés.
2. Se connecter au service FTP sur le port non standard.
3. Exfiltrer la clé privée SSH de l'utilisateur.
4. Se connecter en SSH et récupérer le flag.

Accès

- Cible : `10.0.0.59`, via :
  - `ssh ctf@opnsense-ctf-bacpro.cyberfarm.insa-cvl.fr -o PubkeyAuthentication=no -p 1000`
  - MDP : `46292ca6-b2e8-48cc-bcaf-b185b5fb7cc0`
- Utilisateur FTP : `arkhos`
- Mot de passe FTP : `p4ssw0rd123`
- Format du flag : CTF_INSA_STI_FACILE_9_<...>

3 - Outils recommandés

- `nmap` (scan de ports)
- Client `ftp` (connexion et transfert de fichiers)
- Client `ssh` avec option `-i` (authentification par clé)
- `chmod` (correction des permissions de la clé)

4 - Indices

- Un scan standard ne suffit pas : scannez l'intégralité des ports TCP (`-p-`).
- Le port 21 est une fausse piste. Un autre service FTP écoute sur un port inhabituel.
- Une fois connecté, listez les fichiers cachés (`ls -la`). Les fichiers `id_rsa` et `authorized_keys` évoquent un dossier bien particulier sous Linux.
- Si vous récupérez une clé privée, SSH exige des permissions strictes (`chmod 600`) avant de l'accepter.

Message étrange

Nom : Message étrange
Catégorie : Cryptographie
Niveau : Facile
Code : ctf-04-01
Tags : cryptographie, vigenere, bruteforce, dechiffrement

1 - Vulnérabilité Exploitée

Chiffrement symétrique faible – Chiffre de Vigenère (CWE-327 – Use of a Broken or Risky Cryptographic Algorithm).

Le chiffre de Vigenère est un algorithme de chiffrement par substitution polyalphabétique datant du XVIe siècle. Bien qu'il soit plus robuste que le chiffre de César, il reste vulnérable à l'analyse cryptanalytique dès lors que la clé est courte ou prévisible. Dans ce challenge, la clé `CRYPTO` est retrouvée par analyse des positions connues du flag.

2 - Scénario

Un message archivé est intercepté. Il contient des métadonnées en clair (`REPORT_ID`, `STATUS`) et plusieurs lignes de texte chiffré, dont une ligne au format caractéristique d'un flag CTF. Le chiffrement utilisé est le chiffre de Vigenère avec une clé symétrique répétée. En exploitant la connaissance du format du flag (`CTF_INSA_STI_...`), on retrouve la clé par analyse différentielle, puis on déchiffre l'intégralité du message.

Objectif

1. Identifier le type de chiffrement utilisé.
2. Retrouver la clé de chiffrement par analyse du texte chiffré connu.
3. Déchiffrer le message et extraire le flag.

Accès

- **Fichier fourni :** `challenge.txt`
- **Contenu du fichier :**

```
--- BEGIN ARCHIVED MESSAGE ---
YVJRHAG KM IAS EIWEMCIIYEAM EYYAESPXC
NHIT DGHLWQE GH MC FVAGRDV KFXL AGJQPZS
REPORT_ID: 4927
VYC UEOI ZQ RMT_KEQP_LHK_Vyhr_4.1_1/1_MHNEMJ
STATUS: ARCHIVED
--- END MESSAGE ---
```

- Format du flag : CTF_INSA_STI_Easy_4.1_<...>

3 - Outils recommandés

- `python3` – analyse et déchiffrement automatisé
- **dcode.fr** (`https://www.dcode.fr/chiffre-vigenere`) – outil en ligne de déchiffrement Vigenère avec recherche automatique de clé
- **CyberChef** (`https://gchq.github.io/CyberChef`) – outil polyvalent de manipulation cryptographique

4 - Indices

- Les métadonnées `REPORT_ID: 4927` et `STATUS: ARCHIVED` sont en clair – le chiffrement n'est pas appliqué uniformément, ce qui confirme que les caractères non alphabétiques (`:`, chiffres, espaces) sont conservés tels quels.
- La ligne `VYC UEOI ZQ RMT_KEQP_LHK_Vyhr_4.1_1/1_MHNEMJ` présente le format caractéristique d'un flag CTF chiffré. Le sous-texte `RMT_KEQP_LHK_` correspond à `CTF_INSA_STI_`.
- Si plusieurs décalages César sont testés sans trouver `CTF`, c'est que le chiffrement est **polyalphabétique** – chaque lettre est décalée d'une valeur différente selon sa position, ce qui caractérise le chiffre de Vigenère.
- Comparer lettre à lettre le texte chiffré connu (`RMT...`) et le texte clair attendu (`CTF...`) révèle les valeurs successives de la clé, qui se répète périodiquement.

Oups, trop tard

Nom :	`Oups, trop tard`		
Catégorie :	`Forensique`	**Niveau :**	`Facile`
Code :	`ctf-04-02`	**Tags :**	`logs` `commit` `persistance`

1 - Vulnérabilité Exploitée

La vulnérabilité exploitée est la **persistance des données sensibles dans l'historique Git (Git Secret Leak)**. Git enregistre l'intégralité des changements sous forme de snapshots. Une action corrective (comme supprimer un mot de passe ou un flag dans un commit récent) ne l'efface pas des objets Git créés lors des commits précédents. Sans une réécriture complète de l'historique, les données restent consultables par quiconque accède au dépôt.

2 - Scénario

L'utilisateur Toto (sous le pseudonyme Sailiek) a partagé son code source sur GitHub. Réalisant qu'il avait exposé des informations confidentielles, il a tenté de les masquer en modifiant son code. Cependant, le flag est resté "piégé" dans l'historique des commits, attendant d'être exhumé par une analyse forensique du dépôt.

Objectif

Identifier une fuite de secret en explorant les versions antérieures d'un projet hébergé sur GitHub.

Accès

- Dépôt GitHub public : `https://github.com/Sailiek/Challenge-11/`
- Format du flag : {CTF_INSA_STI_11_<...>}

3 - Outils recommandés

- **GitHub Web Interface** : Pour naviguer dans les commits et inspecter les "diffs".
- **Git CLI** : Utile pour cloner le dépôt et effectuer des recherches globales de chaînes de caractères (`git grep`).

4 - Indices

1. Le titre "Oups, trop tard" suggère qu'une erreur a été commise et que la correction a été tentée après coup.
2. Le tag "commit" oriente l'investigation vers les journaux de modifications de Git.
3. Les fichiers commençant par un point (comme `.env`) sont souvent ignorés par erreur ou contiennent des secrets.

Cadenas

Nom : `Cadenas`
Catégorie : `Réseau`
Code : `ctf-04-03`
Niveau : `Facile`
Tags : `tls` `certificate` `openssl` `forensic` `recon`

1 - Vulnérabilité Exploitée

Fuite d'informations sensibles dans les champs d'un certificat TLS (CWE-200 – Exposure of Sensitive Information).

Un certificat TLS/SSL contient des champs textuels libres (Organisation, Common Name, Subject Alternative Names) dont le contenu est publiquement accessible à quiconque établit une connexion TLS avec le serveur. Stocker des informations sensibles dans ces champs les expose à toute personne capable d'inspecter le certificat – ce qui ne requiert aucune authentification ni accès particulier.

2 - Scénario

Un serveur HTTPS expose un certificat TLS auto-signé invalide. L'indice *"même un cadenas peut divulguer des informations, surtout quand il n'est pas valide"* pointe vers les métadonnées du certificat TLS lui-même. En inspectant les champs du certificat avec `openssl`, le flag est découvert dans le champ `Organization (O)` des champs `Issuer` et `Subject`.

Objectif

1. Se connecter au serveur HTTPS et extraire le certificat TLS.
2. Inspecter les champs du certificat pour trouver le flag.

Accès

- **URL cible :** `https://opnsense-ctf-bacpro.cyberfarm.insa-cvl.fr:1083`
- **Certificat :** auto-signé (CA :TRUE, non approuvé par une autorité de confiance)
- **Clé :** RSA 4096 bits
- Format du flag : `{CTF_INSA_STI_9_<...>}`

3 - Outils recommandés

- `openssl s_client` – établit une connexion TLS et récupère le certificat du serveur
- `openssl x509` – parse et affiche les champs d'un certificat X.509
- **Navigateur web** – l'icône de cadenas permet d'inspecter le certificat visuellement
- `curl -v` – affiche les informations TLS lors d'une requête HTTPS

4 - Indices

- Le titre *"Cadenas"* fait référence à l'icône de cadenas des navigateurs web, symbole du protocole HTTPS et du certificat TLS associé.
- *"Même un cadenas peut divulguer des informations"* – un certificat TLS est public et accessible sans authentification. Ses champs textuels peuvent contenir n'importe quelle information, y compris des données sensibles.
- *"Surtout quand il n'est pas valide"* – un certificat auto-signé ou invalide a souvent été créé manuellement avec des valeurs personnalisées dans ses champs, contrairement aux certificats émis par une autorité de certification officielle qui valide les champs.
- Les champs `Issuer` et `Subject` d'un certificat contiennent plusieurs sous-champs libres : `C` (Country), `ST` (State), `L` (Locality), `O` (Organization), `OU` (Organizational Unit), `CN` (Common Name) – autant de zones potentielles pour cacher des informations.

Jean Dupont vs Sherlock

Nom :	Jean Dupont vs Sherlock		
Catégorie :	OSINT	**Niveau :**	Moyen
Code :	ctf-04-04	**Tags :**	recon enumeration social-engineering

1 - Vulnérabilité Exploitée

Exposition de l'empreinte numérique par corrélation de comptes (CWE-200 – Exposure of Sensitive Information).

L'utilisation de pseudonymes dérivés d'informations personnelles (nom, département géographique, année de création) permet à un attaquant de reconstituer l'identité numérique complète d'une cible par corrélation. Chaque profil public contient des indices qui pointent vers d'autres comptes, formant une chaîne d'informations exploitable avec des techniques OSINT.

2 - Scénario

Jean Dupont est un étudiant en informatique qui a créé plusieurs comptes en ligne au fil des années sous différents pseudonymes. Certains sont actifs, d'autres archivés et oubliés. Chaque profil contient des informations personnelles et des références croisées vers d'autres comptes. En partant d'un seul nom, il est possible de reconstituer toute son empreinte numérique et d'accéder à des fichiers sensibles qu'il pensait avoir oubliés.

Objectif

1. Identifier le premier pseudonyme de Jean Dupont.
2. Corréler les informations entre les profils pour découvrir les comptes suivants.
3. Récupérer les deux flags cachés dans ses profils.

Accès

- Connexion SSH : `ssh sherlock@opnsense-ctf-bacpro.cyberfarm.insa-cvl.fr -p 1084 -o PubkeyAuther`
- Mot de passe : `c9cd88e5-9812-4723-9346-9b85c9f06b1b`
- Outil OSINT : `./sherlock.sh`
- Port forwarding : `-L 8080:10.0.0.84:8080` pour accéder aux profils via navigateur
- Format des flags : CTF_INSA_STI_Easy_5.1_<...>

3 - Outils recommandés

- `./sherlock.sh` – outil OSINT simulé (recherche par username, email, profil, metadata)
- `curl` – accès direct aux profils web internes
- **SSH port forwarding** – accès aux sites internes via le navigateur local
- **Sherlock** (`https://github.com/sherlock-project/sherlock`) – équivalent réel pour la recherche de usernames sur 300+ plateformes

4 - Indices

- L'outil suggère les patterns `jd_dev`, `jdupont+<number>`, `jdup_+<word>` – le préfixe `jd` correspond aux initiales de Jean Dupont.
- Sur DevCommunity, Jean mentionne avoir grandi dans les **Hauts-de-Seine** – département numéro **92**, détail géographique qu'il réutilise dans ses pseudonymes.
- Sur TechForum, Jean mentionne son compte **jd__backup__22** qu'il a oublié et qui contient des fichiers migrés vers "un espace de notes".
- La note de migration sur StorageHub précise : *"logique de nommage plus simple"* et *"distinct de mes autres profils"* – indices pour deviner le nouveau pseudonyme.
- Le pattern de ses pseudonymes suit toujours la structure `jd_<mot>` ou `jdup_<mot>` – la variation `jdup_` combinée avec `notes` correspond au nouveau compte.

Manipulation de Token JWT

Nom :	Manipulation de Token JWT		
Catégorie :	Web	**Niveau :**	Facile
Code :	ctf-04-08	**Tags :**	auth jwt

1 - Vulnérabilité Exploitée

Mauvaise validation des tokens JWT côté serveur : l'application fait confiance au champ `alg` du header du token fourni par le client. En passant `alg: none`, un attaquant peut soumettre un token sans signature, modifier librement le payload (élévation de rôle de `guest` à `admin`), et obtenir un accès non autorisé.

2 - Scénario

L'application web utilise des tokens JWT pour la gestion de session. Les utilisateurs sont connectés par défaut en tant que `guest`. Le serveur émet un token signé en HS256, mais accepte également les tokens déclarant `alg: none` sans vérification de signature — une misconfiguration critique permettant à n'importe qui de se forger un token avec le rôle de son choix.

Objectif

1. Inspecter le cookie de session et décoder le token JWT pour obtenir le premier flag.
2. Modifier le header (`alg: none`) et le payload (`role: admin`).
3. Injecter le token forgé et accéder au panneau d'administration pour obtenir le second flag.

Accès

- URL : `http://opnsense-ctf-bacpro.cyberfarm.insa-cvl.fr:1088/`
- Format des flags : CTF_INSA_STI_Easy_2.3_<...>

3 - Outils recommandés

- Navigateur Web (DevTools : onglet Stockage / Application)
- `https://jwt.io` (décodage et forge de tokens JWT)
- Burp Suite (optionnel, interception de requêtes)

4 - Indices

- Un token JWT est composé de trois parties séparées par des points : `header.payload.signature`. Chaque partie est encodée en Base64.
- Le payload du token d'origine contient une information intéressante — décode-le sur `jwt.io`.
- Le serveur utilise le champ `alg` du header pour choisir l'algorithme de vérification. Que se passe-t-il si tu lui indiques qu'il n'y en a pas ?
- Un token `alg: none` doit se terminer par un point (`.`) indiquant une signature vide.

robots.txt et Répertoires Cachés

Nom :	robots.txt et Répertoires Cachés		
Catégorie :	Web	**Niveau :**	Facile
Code :	ctf-04-09	**Tags :**	http enumeration

1 - Vulnérabilité Exploitée

Exposition de chemins sensibles via le fichier `robots.txt`, combinée à l'absence de protection réelle sur les ressources référencées. Le fichier, destiné à guider les moteurs de recherche, devient une carte des répertoires cachés pour tout attaquant qui le consulte.

2 - Scénario

Un serveur web expose un fichier `robots.txt` listant des chemins "déconseillés" aux moteurs de recherche. Parmi eux se trouve un répertoire caché contenant un fichier confidentiel. Rien n'empêche un visiteur curieux d'y accéder directement malgré l'interdiction formulée dans le fichier.

Objectif

1. Identifier le port d'écoute du serveur web.
2. Consulter `robots.txt` pour y découvrir le premier flag et les chemins cachés.
3. Accéder au répertoire caché et lire le fichier contenant le second flag.

Accès

- URL : `http://opnsense-ctf-bacpro.cyberfarm.insa-cvl.fr:1089/`
- Format des flags : CTF_INSA_STI_Easy_2.1_<...>

3 - Outils recommandés

- Navigateur Web
- `curl` ou `gobuster` (optionnel, énumération de fichiers)

4 - Indices

- Les moteurs de recherche ne sont pas les seuls à lire les règles de `robots.txt` — les attaquants aussi.
- Certains chemins sont "déconseillés", mais rien n'empêche vraiment un visiteur curieux d'y jeter un œil.
- Si l'index du répertoire est désactivé, tente d'accéder directement à des fichiers courants : `flag.txt`, `index.html`.
- Le premier flag se trouve dans `robots.txt` lui-même.

Analyse de Trafic HTTP (PCAP)

Nom :	Analyse de Trafic HTTP (PCAP)		
Catégorie :	Réseau	**Niveau :**	Facile
Code :	ctf-04-10	**Tags :**	pcap http

1 - Vulnérabilité Exploitée

Transmission de données sensibles (tokens, cookies de session) en clair via le protocole HTTP non chiffré. Toute personne capable d'intercepter le trafic réseau peut lire ces informations directement dans une capture de paquets, sans aucun déchiffrement.

2 - Scénario

Une capture de trafic réseau (`traffic.pcap`) a été interceptée sur le réseau d'une organisation utilisant HTTP en clair. Parmi les échanges capturés se trouvent des requêtes contenant des tokens et des cookies de session transmis sans protection.

L'objectif est d'analyser ce trafic pour extraire les deux flags dissimulés dans les paramètres GET et les headers de cookies.

Objectif

1. Ouvrir la capture dans Wireshark et filtrer le trafic HTTP.
2. Identifier la requête GET contenant le premier flag dans un paramètre d'URL.
3. Identifier le header `Cookie` contenant le second flag.

Accès

- Fichier fourni : `traffic.pcap`
- Format des flags : CTF_INSA_STI_Easy_3.1_<...>

3 - Outils recommandés

- Wireshark (analyse de capture réseau)
- `tshark` (optionnel, analyse en ligne de commande)
- `strings` (optionnel, extraction rapide de texte)

4 - Indices

- Ouvre le fichier PCAP dans Wireshark et applique le filtre `http`.
- Inspecte les URL des requêtes GET : un paramètre peut contenir un token.
- Examine les headers des requêtes HTTP, notamment le champ `Cookie`.
- La fonctionnalité *Follow TCP Stream* permet de lire un échange complet d'un seul coup.