

SÉCURISATION DES PROTOCOLES RÉSEAUX

ACTIVITÉ 1 : ÉVALUER LA SÉCURITÉ DES PROTOCOLES POP ET SSH

Description du thème

Propriétés	Description
Intitulé long	Sécurisation des protocoles réseaux - Évaluation de la sécurité des protocoles POP et SSH
Formation(s) concernée(s)	<input type="checkbox"/> Classes de première Sciences et technologies du management et de la gestion (STMG) <input type="checkbox"/> Terminale STMG Système d'information de gestion (SIG) <input checked="" type="checkbox"/> BTS Services Informatiques aux Organisations
Matière(s)	<input type="checkbox"/> Sciences de gestion <input type="checkbox"/> SIG <input type="checkbox"/> Bloc 1 – Support et mise à disposition de services informatiques <input type="checkbox"/> Bloc 2 SISR – Administration des systèmes et des réseaux <input type="checkbox"/> Bloc 2 SLAM – Conception et développement d'applications <input checked="" type="checkbox"/> Bloc 3 SISR – Cybersécurité des services informatiques <input type="checkbox"/> Bloc 3 SLAM – Cybersécurité des services informatiques
Présentation	<p>Ce labo a pour objectifs de maîtriser les techniques de base de reconnaissance réseau, tout en comprenant les vulnérabilités liées aux services mal sécurisés. Les participants apprendront à appréhender les risques associés aux mots de passe faibles et mettront en pratique des outils de sécurité offensive. Enfin, il soulignera l'importance des bonnes pratiques de sécurité pour renforcer la protection des systèmes informatiques.</p> <p>Cette première activité aborde les enjeux associés aux protocoles POP (pour la messagerie) et SSH (pour l'accès distant sécurisé).</p>
Savoirs	<p>Principes de la sécurité : disponibilité, intégrité, confidentialité, preuve.</p> <p>Sécurité des communications numériques : rôle des protocoles, segmentation, administration, restriction.</p>

Compétences	<ul style="list-style-type: none"> • Sécuriser les équipements et les usages des utilisateurs ; <ul style="list-style-type: none"> ◦ Identifier les menaces et mettre en œuvre les défenses appropriées ; ◦ Gérer les accès et les privilèges appropriés. • Garantir la disponibilité, l'intégrité et la confidentialité des services informatiques et des données de l'organisation face à des cyberattaques. <ul style="list-style-type: none"> ◦ Caractériser les risques liés à l'utilisation malveillante d'un service informatique. • Recenser les conséquences d'une perte de disponibilité, d'intégrité ou de confidentialité.
Transversalité	
Prérequis	Commandes de base d'administration d'un système <i>Linux</i> .
Outils	<p>Le logiciel Proxmox (version 8 ou 9), une machine Kali Linux et une machine Debian 12/13 disposant d'un accès à internet.</p> <p>Les scripts sont disponibles sur la forge du réseau Certa : https://forge.apps.education.fr/reseau-certa/bts-sio/activites-ctf</p>
Mots-clés	PROXMOX, containers LXC, vulnérabilités, POP, SSH.
Durée	Deux heures
Auteur·e·s	Damien SCONTRINO, avec la relecture, les tests et les suggestions de Patrice DIGNAN et Apollonie RAFFALLI
Version	V1.0
Date de publication	XX Novembre 2025

Dernières révisions

Ce tableau contient les modifications apportées au document après sa publication uniquement.

Date	Auteur·e	Description

SOMMAIRE

A. Le protocole POP3.....	5
A.1. Présentation du protocole POP3.....	5
A.2. Problématiques de sécurités associées à POP3.....	5
A.3. Bonnes pratiques.....	5
B. Le protocole SSH.....	5
B.1. Présentation du protocole SSH.....	5
B.2. Problématiques de sécurités associées à SSH.....	5
B.3. Bonnes pratiques.....	6
C. Présentation des défis.....	6
D. Défi n°1 : brèche de confidentialité sur un service POP3.....	6
D.1. Travaux préparatoires.....	6
D.2. Reconnaissance initiale.....	7
D.3. Exploitation du service POP3.....	7
E. Défi n°2 : force brute d'un compte SSH.....	8
E.1. Travaux préparatoires.....	8
E.2. Attaque du compte SSH.....	8
E.3. Connexion au serveur SSH et capture du flag.....	8
E.4. Conclusions et bonnes pratiques.....	8
F. Dossier documentaire.....	9
F.1. Notice de déploiement de l'activité 1.....	9
F.2. Reconnaissance initiale.....	10
F.3. Défi n°1 : Brèche de confidentialité sur un service POP3.....	10
F.4. Défi n°2 : Force brute d'un compte SSH.....	11
F.5. Script CTF1 à placer dans le template LXC avant le clonage.....	11
F.6. Script clonage LXC à lancer depuis le shell du nœud Proxmox.....	15

A. LE PROTOCOLE POP3

A.1. PRÉSENTATION DU PROTOCOLE POP3

Le protocole POP3, ou Post Office Protocol version 3, est un protocole de communication utilisé pour récupérer des courriers électroniques à partir d'un serveur. Conçu pour permettre aux utilisateurs de télécharger leurs messages sur leur appareil local, il fonctionne principalement en mode déconnecté, ce qui signifie que les utilisateurs peuvent lire leurs e-mails sans être connectés à Internet une fois les messages téléchargés. POP3 est simple à mettre en œuvre et largement compatible avec divers clients de messagerie. Cependant, il présente des limitations, notamment en ce qui concerne la gestion des messages sur le serveur, car par défaut, les e-mails sont souvent supprimés du serveur après téléchargement.

A.2. PROBLÉMATIQUES DE SÉCURITÉS ASSOCIÉES À POP3

Le protocole POP3 présente plusieurs problématiques de cybersécurité, notamment la vulnérabilité aux attaques de type "man-in-the-middle" (homme du milieu), où des cybercriminels peuvent intercepter les communications entre le client et le serveur. De plus, l'absence de chiffrement par défaut expose les informations sensibles, comme les identifiants de connexion, à des risques de vol. Les mots de passe faibles sont également une préoccupation, car ils peuvent être facilement compromis, permettant ainsi un accès non autorisé aux comptes de messagerie. Enfin, la gestion des e-mails sur le serveur, qui supprime souvent les messages après téléchargement, rend difficile la récupération des données en cas de perte ou d'attaque.

A.3. BONNES PRATIQUES

Pour sécuriser le protocole POP3, plusieurs bonnes pratiques et contre-mesures peuvent être mises en œuvre. Tout d'abord, il est essentiel d'utiliser des connexions chiffrées via SSL/TLS pour protéger les données échangées, y compris les identifiants de connexion. L'adoption de mots de passe forts et uniques contribue également à réduire le risque de compromission des comptes. De plus, il est recommandé de configurer le serveur pour conserver une copie des e-mails, permettant ainsi une meilleure gestion et une récupération potentielle en cas d'incident. Enfin, la mise en œuvre de l'authentification à deux facteurs (2FA) renforce la sécurité en ajoutant une couche supplémentaire de protection contre les accès non autorisés.

B. LE PROTOCOLE SSH

B.1. PRÉSENTATION DU PROTOCOLE SSH

Le protocole SSH, ou Secure Shell, est un protocole de communication sécurisé utilisé principalement pour accéder à distance à des systèmes informatiques. Il permet aux utilisateurs de se connecter à un serveur de manière sécurisée, en chiffrant toutes les données échangées, y compris les identifiants de connexion. SSH est largement utilisé pour l'administration des serveurs, le transfert de fichiers et l'exécution de commandes à distance. Grâce à ses mécanismes d'authentification robustes, il offre une protection contre les attaques de type "man-in-the-middle" et assure l'intégrité et la confidentialité des communications. SSH est devenu un standard incontournable pour la gestion sécurisée des infrastructures informatiques.

B.2. PROBLÉMATIQUES DE SÉCURITÉS ASSOCIÉES À SSH

Le protocole SSH, bien qu'étant sécurisé, présente plusieurs problématiques de cybersécurité. L'une des principales préoccupations est le risque d'attaques par force brute, où des attaquants tentent de deviner les mots de passe d'accès. De plus, si les clés d'authentification ne sont pas gérées correctement, elles peuvent être compromises, permettant ainsi des accès non autorisés.

Les utilisateurs négligent souvent de désactiver les comptes obsolètes ou inutilisés, augmentant ainsi les vecteurs d'attaque. Enfin, une mauvaise configuration des serveurs SSH, comme l'autorisation d'authentification par mot de passe au lieu de clés, peut exposer des failles supplémentaires. Ces risques soulignent l'importance de pratiques de sécurité rigoureuses pour protéger les connexions SSH.

B.3. BONNES PRATIQUES

Pour garantir la sécurité des connexions SSH, plusieurs bonnes pratiques et contre-mesures doivent être adoptées. Tout d'abord, il est recommandé d'utiliser l'authentification par clé publique plutôt que par mot de passe, car elle offre une protection renforcée contre les attaques par force brute. Les clés privées doivent être conservées en lieu sûr et protégées par un mot de passe fort. De plus, il est essentiel de désactiver l'accès root direct et de créer des comptes utilisateurs spécifiques avec des privilèges limités. La mise en place d'un pare-feu pour filtrer les adresses IP autorisées à se connecter au serveur SSH renforce également la sécurité. Enfin, il est conseillé de surveiller régulièrement les journaux d'accès pour détecter toute activité suspecte et de mettre à jour régulièrement le logiciel SSH pour corriger les vulnérabilités connues.

C. PRÉSENTATION DES DÉFIS

Les deux défis suivants requièrent la création de conteneurs Debian 12 LXC pour les étudiants à l'aide d'un script de génération automatique disponible dans le dossier documentaire.

Avant de commencer cette activité, il est nécessaire de préparer l'environnement de travail en générant les containers Debian 12 pour les étudiants.

Défi n°1 : brèche de confidentialité sur un serveur POP3

Dans ce premier défi, l'objectif est de découvrir le mot de passe d'un compte POP3, puis d'explorer les e-mails de la victime pour identifier un nom d'utilisateur qui sera utilisé pour une attaque par force brute sur un compte SSH.

Défi n°2 : force brute d'un compte SSH

Dans ce deuxième défi, l'objectif est d'effectuer une attaque par force brute sur un compte SSH en utilisant le nom d'utilisateur précédemment découvert. Pour relever ce défi, il est nécessaire de capturer un *flag*. Dans le cadre des exercices de sécurité, un *flag* est une chaîne de caractères unique qui atteste de la réussite du défi, agissant comme un véritable trophée numérique.

D. DÉFI N°1 : BRÈCHE DE CONFIDENTIALITÉ SUR UN SERVICE POP3

D.1. TRAVAUX PRÉPARATOIRES

Pour le professeur

Suivre les étapes de la notice de déploiement de cette première activité. Cela inclut le script de configuration et le clonage pour les étudiants. Le script s'ajuste au nombre d'étudiants.

- Q1.** Préparez votre environnement de travail en lançant une machine Kali dédiée à l'attaque. Modifiez ensuite l'adresse IP de cette machine Kali pour qu'elle appartienne au même réseau que les conteneurs étudiants créés (par exemple : 192.36.253.1X, où X représente votre numéro d'étudiant). Enfin, ouvrez un terminal sur cette machine Kali.

D.2. RECONNAISSANCE INITIALE

Q2. Depuis le terminal de la machine Kali, utilisez la commande suivante (à adapter) :

```
nmap [Options] ${TARGET_IP}
```

Remplacer `${TARGET_IP}` par l'adresse IP de votre cible. Ce sera par exemple 192.36.253.100 + votre numéro d'étudiant. **Exemple pour l'étudiant12, l'adresse IP de la machine à hacker sera 192.36.253.112.**

Vous devez trouver l'option ou les options qui permettront de répondre précisément à la question.

Q3. Concluez sur l'existence d'un service POP3 et SSH sur le serveur cible et indiquez les versions de ces deux services sur le serveur cible.

D.3. EXPLOITATION DU SERVICE POP3

Q4. Depuis le terminal de la machine Kali, exécutez la commande suivante avec le logiciel Hydra. Notez ensuite le mot de passe découvert, qui permettra de se connecter au serveur POP3. Dans cette attaque, nous supposons que le login 'admin' existe. Veuillez noter que l'attaque peut prendre un certain temps.

```
hydra -l admin -P /usr/share/wordlists/rockyou.txt ${TARGET_IP} pop3 -V
```

Paramètres utilisés :

-l admin : Spécifie le nom d'utilisateur à tester

-P /usr/share/wordlists/rockyou.txt.gz : Utilise le dictionnaire rockyou

pop3 : Spécifie le protocole ciblé

-V : Mode verbeux pour voir la progression

Q5. Utilisez la commande netcat afin de vous connecter au serveur POP3 compromis.

```
nc ${TARGET_IP} 110
```

```
user admin
```

```
pass <mot_de_passe_trouvé>
```

Q6. Explorez les e-mails du serveur compromis en utilisant les commandes list et retr <numéro> pour lire chaque message. Votre objectif est de trouver un nom d'utilisateur qui servira pour le prochain défi.



Lire attentivement chaque e-mail car ils peuvent contenir des informations importantes pour la suite de l'attaque.

E. DÉFI N°2 : FORCE BRUTE D'UN COMPTE SSH

E.1. TRAVAUX PRÉPARATOIRES

- Q1. Vérifiez que le défi n°1 concernant la compromission du serveur POP3 a été réalisé avec succès, puis identifiez un compte SSH valide en examinant attentivement les courriels récupérés.

E.2. ATTAQUE DU COMPTE SSH

- Q2. Depuis le terminal de la machine Kali, utilisez la commande suivante pour découvrir le mot de passe associé au nom d'utilisateur précédemment identifié sur le serveur POP3.

```
hydra -l ${IDENTIFIANT} -P /usr/share/wordlists/rockyou.txt.gz ${TARGET_IP} ssh -V
```

- ⚠ Si la commande ne répond pas après 30 secondes, la relancer.

E.3. CONNEXION AU SERVEUR SSH ET CAPTURE DU FLAG

- Q3. Connectez-vous au serveur SSH sur le serveur cible en utilisant le compte et le mot de passe trouvé.

```
ssh user@${TARGET_IP}
```

- Q4. Lisez le contenu du fichier 'flag.txt' et reportez-le sur votre dossier documentaire.

E.4. CONCLUSIONS ET BONNES PRATIQUES

Points clés à retenir

1. La reconnaissance est une étape essentielle en sécurité offensive.
2. Les services mal sécurisés peuvent constituer une vulnérabilité.
3. Les mots de passe faibles représentent un risque majeur.

Limitations en environnement réel

- Les attaques par dictionnaire peuvent être longues.
- L'efficacité dépend de la qualité du dictionnaire utilisé.

Protection contre le brute force

- Configurez fail2ban pour bloquer les IP après plusieurs échecs de connexion à SSH.
- Limitez les connexions simultanées.
- Mettez en place des alertes en cas de multiples échecs de connexion.
- Privilégiez les clés SSH aux mots de passe lorsque cela est possible.

Pour aller plus loin

- Explorez d'autres dictionnaires de mots de passe.
- Étudiez les mécanismes de protection contre le brute-force.
- Familiarisez-vous avec les bonnes pratiques de sécurisation des services.

F. DOSSIER DOCUMENTAIRE

F.1. NOTICE DE DÉPLOIEMENT DE L'ACTIVITÉ 1

1. Création du conteneur modèle

- Se connecter à l'interface Proxmox.
- Cliquer sur Créer CT.
- Choisir un ID libre (exemple : 5040) et donner un nom explicite (ex : CTF1-MODELE).
- Attribuer un mot de passe.
- Sélectionner le template Debian standard (ex : debian-12-standard *.tar.zst ou `debian-13-standard-*.tar.zst`).
- Laisser tous les paramètres par défaut (pas besoin de configurer le réseau ni les ressources).
- Finaliser la création et démarrer le conteneur.

2. Préparation du script de configuration

- Ouvrir un shell dans le conteneur modèle (via "Console" ou `pct enter 5040`).
- En tant que root, créer le fichier `/root/script_ctf1.sh` :

```
nano /root/script_ctf1.sh
```

- Coller le contenu du script de configuration CTF1.
- Rendre le script exécutable :

```
chmod +x /root/script_ctf1.sh
```

- Quitter le conteneur et l'arrêter proprement :

```
exit  
pct shutdown 5040
```



Le fichier `script.sh` sera automatiquement présent dans chaque clone.

3. Lancement du clonage pour les étudiants

- Se connecter en root sur le nœud Proxmox.
- Créer un fichier de script de clonage :

```
nano /root/clonage_ctf1.sh
```

- Coller le script de clonage fourni. Il demandera :
- l'ID du conteneur modèle (ex : 5040),
- le nombre d'étudiants,
- et clonera un conteneur par étudiant, en exécutant `/root/script_ctf1.sh <num>` à l'intérieur.
- Rendre ce script exécutable :

```
chmod +x /root/clonage_ctf1.sh
```

- Lancer le script :

```
./clonage_ctf1.sh
```

Suivez les questions affichées pour fournir les informations requises par le script :

1. **Template source** : Indiquez le numéro du conteneur modèle.
2. **ID de base** : Précisez le numéro de départ pour le premier conteneur étudiant à créer.
3. **Stockage** : Choisissez le type de stockage à utiliser sur la ferme Proxmox pour la création des conteneurs.
4. **Bridge réseau** : Sélectionnez la carte réseau à utiliser pour la connexion des conteneurs étudiants.
5. **Configuration réseau** : Indiquez le réseau et la passerelle à utiliser pour configurer les conteneurs étudiants.
6. **Numéros de machine** : Définissez le numéro de départ et le numéro de fin pour les conteneurs étudiants.

i Après validation des paramètres, le script procède à la création des conteneurs pour les étudiants avec leur login, leur mot de passe et leur flag.

Exemples :

- **Étudiant 1 :**
 - CT 800001 (CTF1-Etudiant01)
 - VM TP1 configurée pour étudiant 1 (A)
 - SSH : Aavid / booboo - POP3 : admin / september - FLAG : FLAG{TP1_A_de76f6c9}
- **Étudiant 2 :**
 - CT 800002 (CTF1-Etudiant02)
 - VM TP1 configurée pour étudiant 2 (B)
 - SSH : Bavid / kissme - POP3 : admin / december - FLAG : FLAG{TP1_B_e9a2da23}
- **Étudiant 3 :**
 - CT 800003 (CTF1-Etudiant03)
 - VM TP1 configurée pour étudiant 3 (C)
 - SSH : Cavid / harley - POP3 : admin / morgan - FLAG : FLAG{TP1_C_67192230}

F.2. RECONNAISSANCE INITIALE

Préparation de la machine kali servant pour les défis

Chaque étudiant devra disposer d'une machine Kali sur un réseau IP permettant la communication avec la machine automatiquement créée. L'activité se déroulera depuis cette machine Kali. Les étudiants n'ont pas besoin de connaître les identifiants ni le *flag* des conteneurs créés. Cependant, ils doivent être en mesure de lancer un ping et d'effectuer des reconnaissances sur leur machine étudiante respective depuis la machine Kali.

F.3. DÉFI N°1 : BRÈCHE DE CONFIDENTIALITÉ SUR UN SERVICE POP3

Hydra est un outil de cracking de mots de passe utilisé sur Kali Linux. Il permet d'effectuer des attaques par force brute ou par dictionnaire sur divers protocoles de connexion, tels que SSH, FTP, et POP3. Grâce à sa polyvalence et sa rapidité, Hydra aide les utilisateurs à tester la sécurité des mots de passe de manière efficace. Voici un exemple à adapter selon le réseau IP utilisé.

```
hydra -l admin -P /usr/share/wordlists/rockyou.txt.gz 192.36.253.101 ssh -V
```

Une fois le mot de passe trouvé, il est possible de tester une connexion directe sur le serveur cible POP3 via la commande netcat.

Netcat, souvent appelé "le couteau suisse des réseaux", est un outil puissant disponible sur Kali Linux. Il permet d'établir des connexions TCP ou UDP, de transférer des fichiers, et de créer des shells interactifs. Utilisé pour le dépannage réseau et les tests de sécurité, Netcat facilite la communication entre machines et l'exécution de commandes à distance. Voici un exemple à adapter selon le réseau IP utilisé.

```
nc 192.36.253.101
```

La commande demande de saisir le nom de l'utilisateur (admin dans notre cas) puis le mot de passe trouvé précédemment.

Une fois connecté au serveur, il est possible de consulter les emails et de relever un identifiant valide de compte SSH.

F.4. DÉFI N°2 : FORCE BRUTE D'UN COMPTE SSH

Le logiciel Hydra est un outil puissant de test de pénétration qui permet non seulement de réaliser des attaques par force brute sur divers protocoles, mais il est particulièrement efficace pour brute forcer des comptes SSH. Grâce à sa capacité à gérer plusieurs threads simultanément, Hydra peut tenter rapidement de multiples combinaisons d'identifiants, rendant ainsi l'accès non autorisé à des systèmes distants plus accessible pour les testeurs de sécurité.

```
hydra -l <login_trouvé> -P /usr/share/wordlists/rockyou.txt <ip-  
container_cible> ssh -V
```

Le script peut mettre un certain temps à découvrir le bon mot de passe. Une fois celui-ci trouvé, la connexion au compte via SSH permet de récupérer le flag (trophée numérique).

F.5. SCRIPT CTF1 À PLACER DANS LE TEMPLATE LXC AVANT LE CLONAGE

```
#!/bin/bash
# Script final pour VM TP1 Debian 12 -
# pour clonage LXC, jusqu'à 35 étudiants avec cycle de lettres A-X et rebelote

set -e

# On commence par récupérer le numéro de l'étudiant pour tout adapter à son cas
# : nom de VM, comptes, mot de passe, flag, etc.
read -p "Entrez le numéro de l'étudiant (ex: 06) : " NUMERO
if [ -z "$NUMERO" ]; then
    echo "Numéro requis."
    exit 1
fi

# Je m'assure que la locale en_US.UTF-8 est bien installée, sinon apt et perl
# râlent, surtout dans les conteneurs Debian.
export DEBIAN_FRONTEND=noninteractive

# Générer la locale si elle n'existe pas encore
if ! locale -a | grep -q "en_US.utf8"; then
    echo "en_US.UTF-8 UTF-8" >> /etc/locale.gen
    locale-gen
fi
```

```

# Configurer la locale par défaut
update-locale LANG=en_US.UTF-8
export LANG=en_US.UTF-8
export LC_ALL=en_US.UTF-8

# Je choisis une lettre (A à X) en fonction du numéro d'étudiant pour générer
un identifiant SSH unique (Adavid, Bdavid...). Pour se conformer aux lettres du
Lab Stormshield le 24eme etudiant aura la lettre A et ainsi de suite
LETTRE=$(echo {A..X} | awk -v n="$NUMERO" '{print $(( (n - 1) % 24 + 1 ))}')

USER_SSH="${LETTRE}david"
USER_POP3="admin"

# J'ai défini ici deux listes de mots de passe très simples exprès pour qu'ils
soient cassables facilement pendant le TP (bruteforce POP3 / SSH) sinon temps
de hydra trop long.
MOTS_POP3=(booboo kissme harley ronaldo iloveyou1 precious october inuyasha
peaches veronica chris 888888 adriana cutie james banana prince friend jesus1
crystal celtic zxcvbnm edward oliver diana samsung freedom angelo hotdog friday
summer winter monkey bubble sailor)
MOTS_ADMIN=(september december morgan mariposa maria gabriela iloveyou2 bailey
jeremy pamela kimberly gemini shannon pictures asshole sophie jessie hellokitty
claudia babygirl1 angelica austin mahalko victor horses tiffany mariana eduardo
barbara destiny welcome ashley 123qwe hello1 hockey)

PASSWD_SSH="${MOTS_POP3[$((NUMERO - 1))]}"
PASSWD_POP3="${MOTS_ADMIN[$((NUMERO - 1))]}"
FLAG="FLAG{TP1_${LETTRE}_${echo -n \"TP1_${LETTRE}\" | sha256sum | cut -c1-8})"
HOSTNAME="CTF-1-Etudiant$NUMERO"
LOGFILE="vm_tp1_full_setup_${NUMERO}.log"
exec > >(tee -a "$LOGFILE") 2>&1

# Je détecte ici automatiquement l'interface réseau active et l'adresse IP, car
elle peut changer selon le bridge utilisé dans Proxmox.
IFACE=$(ip -o link show | awk -F': ' '{ $2 !~ /^lo/ {gsub(/@.*/, "", $2); print
$2; exit}}')
if [ -z "$IFACE" ]; then
    echo "Interface réseau non détectée."
    ip a
    exit 1
fi

IP_CIBLE=$(ip -o -f inet addr show "$IFACE" | awk '{print $4}' | cut -d'/' -f1)
MASK=$(ip -o -f inet addr show "$IFACE" | awk '{print $4}' | cut -d'/' -f2)
GATEWAY=$(ip route | grep default | awk '{print $3}')

# Je configure le nom de la machine et j'installe les paquets nécessaires pour
le TP : mail, SSH, sudo...
hostnamectl set-hostname "$HOSTNAME"
echo "127.0.1.1 $HOSTNAME" >> /etc/hosts

debconf-set-selections <<< "postfix postfix/main_mailer_type string Local only"
DEBIAN_FRONTEND=noninteractive apt update
DEBIAN_FRONTEND=noninteractive apt install -y dovecot-pop3d postfix openssh-
server sudo net-tools passwd

# Je cree un compte formation
useradd -m -s /bin/bash formation || true
echo "formation:Morphe13@" | chpasswd
usermod -aG sudo formation

useradd -m -s /bin/bash $USER_POP3 || true
echo "$USER_POP3:$PASSWD_POP3" | chpasswd

```

```
if ! id TEMPLATEUSER &>/dev/null; then
    useradd -m -s /bin/bash TEMPLATEUSER
    echo "TEMPLATEUSER:$PASSWD_SSH" | chpasswd
fi

# Ici j'adapte la config de Dovecot pour qu'il accepte l'authentification
simple et qu'il stocke les mails au bon endroit (Maildir dans le home).
sed -i 's/^#disable_plaintext_auth = yes/disable_plaintext_auth = no/'
/etc/dovecot/conf.d/10-auth.conf
sed -i 's|^mail_location = .*|mail_location = maildir:~/Maildir|'
/etc/dovecot/conf.d/10-mail.conf
systemctl enable dovecot postfix ssh
systemctl restart dovecot postfix ssh

# Je renomme TEMPLATEUSER pour qu'il prenne le nom du compte SSH de l'étudiant
(par ex. Cdavid), et je renomme aussi son groupe et son home.
if id TEMPLATEUSER &>/dev/null && getent group TEMPLATEUSER &>/dev/null; then
    groupmod -n "$USER_SSH" TEMPLATEUSER
    usermod -l "$USER_SSH" -g "$USER_SSH" TEMPLATEUSER
    usermod -d "/home/$USER_SSH" -m "$USER_SSH"
else
    echo "TEMPLATEUSER manquant."
    exit 1
fi

# Je recrée proprement une Maildir vide pour admin, car je peux ensuite
injecter des mails sans erreur ensuite (retr 5).
rm -rf /home/$USER_POP3/Maildir
sudo -u $USER_POP3 maildirmake.dovecot /home/$USER_POP3/Maildir
MAILDIR="/home/$USER_POP3/Maildir/new"
mkdir -p "$MAILDIR"
chown -R $USER_POP3:$USER_POP3 /home/$USER_POP3/Maildir

# J'injecte ici des faux mails dans la boîte POP3 d'admin. Le 5e mail donne
l'identifiant SSH de l'étudiant, à retrouver avec retr 5 dans le TP.
cat <<EOF > "$MAILDIR/01_commande.$HOSTNAME"
From: amazon@fake.com
Subject: Votre commande n° 456123

Merci pour votre achat. Livraison prévue sous 3 jours.
EOF

cat <<EOF > "$MAILDIR/02_iphone.$HOSTNAME"
From: newsletter@random.org
Subject: Gagnez un iPhone 16 !

Cliquez ici pour participer au tirage au sort...
EOF

cat <<EOF > "$MAILDIR/03_securite.$HOSTNAME"
From: support@security.net
Subject: Alerte de connexion inhabituelle

Une connexion suspecte a été détectée sur votre compte.

Cordialement.
Le site security.net
EOF

cat <<EOF > "$MAILDIR/04_maj.$HOSTNAME"
From: urgent@cleanergy.local
Subject: Rappel expiration de votre mot de passe
```

Bonjour,
Conformément à la nouvelle politique de renouvellement des mots de passe décidé
il y a 6 mois, votre mot de passe arrive à expiration.
Veuillez le mettre à jour pour pouvoir continuer à accéder aux serveurs.

Bien cordialement.
Le service informatique.
EOF

```
cat <<EOF > "$MAILDIR/05_ssh.$HOSTNAME"  
From: Damien SCONTRINO <admin@localhost>  
To: $USER_POP3@cleanergy.local  
Subject: Accès SSH
```

Bonjour David,

Voici l'identifiant du compte que je t'ai créé pour l'accès SSH dont tu as
besoin : \$USER_SSH
Je t'ai transmis par SMS un mot de passe temporaire. Tu devras le changer dès
ta première connexion, c'est automatique.
Merci de ne pas prendre ça à la légère. Un mot de passe solide, ce n'est pas un
caprice d'informaticien... c'est simplement la base quand on prétend évoluer dans
ce métier.
Je sais que tu aimes faire les choses à ta manière, mais sur ce genre de point,
il faut apprendre à être rigoureux.
À toi de jouer.
Cordialement,
Damien SCONTRINO.
EOF

```
chown -R $USER_POP3:$USER_POP3 /home/$USER_POP3/Maildir  
doveadm index -u $USER_POP3 INBOX  
systemctl restart dovecot postfix ssh
```

```
# Je génère un flag unique pour l'étudiant à partir de sa lettre, et je le  
stocke dans son home. Il doit le retrouver pour valider l'étape.  
echo "$FLAG" > "/home/$USER_SSH/flag.txt"  
chown "$USER_SSH:$USER_SSH" "/home/$USER_SSH/flag.txt"  
chmod 644 "/home/$USER_SSH/flag.txt"
```

```
# Ce fichier /root/infos_tp1_NUMERO.txt permet à l'enseignant de retrouver tous  
les identifiants, l'IP, le flag... pour vérif ou dépannage.  
cat <<INFO > "/root/infos_tp1_${NUMERO}.txt"  
Etudiant n° : $NUMERO  
Lettre      : $LETTRE  
Nom SSH     : $USER_SSH  
Mot de passe SSH : $PASSWD_SSH  
Compte POP3  : $USER_POP3  
Mot de passe POP3 : $PASSWD_POP3  
IP          : $IP_CIBLE/$MASK  
Interface   : $IFACE  
Passerelle  : $GATEWAY  
Flag        : $FLAG  
INFO
```

```
# Je vérifie que dovecot, postfix, ssh tournent bien, et que les utilisateurs  
attendus sont créés. J'affiche aussi l'IP attribuée.  
for service in dovecot postfix ssh; do  
    systemctl is-active --quiet $service && echo "$service : actif" || echo  
"$service : inactif"  
done
```

```
for user in "$USER_POP3" "$USER_SSH" "formation"; do  
    id "$user" &>/dev/null && echo "Utilisateur '$user' OK" || echo "Utilisateur
```

```
'$user' manquant"
done

ip a show "$IFACE" | grep 'inet ' | awk '{print "Adresse IP détectée : " $2}'

echo "VM TP1 configurée pour étudiant $NUMERO ($LETTRE)"
echo "SSH : $USER_SSH / $PASSWD_SSH - POP3 : $USER_POP3 / $PASSWD_POP3 - FLAG : $FLAG"

# Le script s'efface lui-même à la fin, pour éviter qu'un étudiant, même s'il
# n'est pas censé voir la VM dans l'interface graphique de proxmox, puisse, je ne
# sais comment, le relancer ou récupérer les variables en lisant le code.
rm -- "$0"
```

F.6. SCRIPT CLONAGE LXC À LANCER DEPUIS LE SHELL DU NŒUD PROXMOX

```
#!/bin/bash

SCRIPT_IN_LXC="/root/script_ctf1.sh"
NAMESERVER="9.9.9.9"

# choix de l'ID du template LXC à cloner dans lequel le script est déjà présent
read -p "ID du conteneur modèle à cloner (ex: 5042) : " TEMPLATE_ID

# début de plage pour les IDs des nouveaux conteneurs
read -p "Base pour les IDs des clones (ex: 300000 pour un premier CTID à 300001) : " BASE_ID

# Choix du STOCKAGE dans la liste des stockages détectées
echo "Liste des stockages disponibles :"
mapfile -t STORAGE_LIST <<(pvesm status | awk 'NR>1 {print $1}')
```

```
for i in "${!STORAGE_LIST[@]}"; do
    echo " [${(i+1)}] ${STORAGE_LIST[$i]}"
done

while true; do
    read -p "Entrez le numéro du stockage sur lequel seront créés les container
de chaque étudiant : " STORAGE_NUM
    if [[ "$STORAGE_NUM" =~ ^[0-9]+$ ]] && (( STORAGE_NUM >= 1 && STORAGE_NUM
<= ${#STORAGE_LIST[@]} )); then
        STORAGE="${STORAGE_LIST[${(STORAGE_NUM-1)}]}"
        echo "Stockage sélectionné : $STORAGE"
        break
    else
        echo "Choix invalide. Réessayer."
    fi
done

# choix de l'interface réseau (avec internet c'est nécessaire)
echo ""
echo "Liste des bridges disponibles :"
mapfile -t BRIDGE_LIST <<(ip -o link show | awk -F': ' '{print $2}' | grep
'^vmb')
for i in "${!BRIDGE_LIST[@]}"; do
    echo " [${(i+1)}] ${BRIDGE_LIST[$i]}"
done

while true; do
    read -p "Entrez le numéro de l'interface réseau qui doit permettre une
connexion internet : " BRIDGE_NUM
    if [[ "$BRIDGE_NUM" =~ ^[0-9]+$ ]] && (( BRIDGE_NUM >= 1 && BRIDGE_NUM <= $
${#BRIDGE_LIST[@]} )); then
        BRIDGE="${BRIDGE_LIST[${(BRIDGE_NUM-1)}]}"
```

```

        echo "Bridge sélectionné : $BRIDGE"
        break
    else
        echo "Choix invalide. Réessayer."
    fi
done

# IP & Numéros d'étudiants
while true; do
    read -p "Préfixe IP (ex: 192.168.56) : " NETWORK_BASE
    [[ "$NETWORK_BASE" =~ ^([0-9]{1,3}\.){2}[0-9]{1,3}$ ]] && break
    echo "Format invalide. Exemple attendu : 192.168.56"
done

while true; do
    read -p "Masque réseau (valeurs autorisées : /8, /16, /24) : " MASK
    [[ "$MASK" =~ ^/(8|16|24)$ ]] && break
    echo "Seuls /8, /16 et /24 sont acceptés."
done

read -p "Adresse passerelle (ex: 192.168.56.1) : " GATEWAY
read -p "Numéro de départ (ex: 1) : " START_NUM
read -p "Numéro de fin (ex: 27) : " END_NUM

# === Confirmation ===
echo ""
echo "Résumé de la configuration :"
echo "  - Template source : $TEMPLATE_ID"
echo "  - ID de base : $BASE_ID"
echo "  - Stockage : $STORAGE"
echo "  - Bridge réseau : $BRIDGE"
echo "  - Réseau CIDR : $NETWORK_BASE.1XX$MASK"
echo "  - Passerelle : $GATEWAY"
echo "  - Numéros : $START_NUM à $END_NUM"
echo ""

read -p "Continuer avec ces paramètres ? (y/N) : " CONFIRM
[[ "$CONFIRM" != "y" && "$CONFIRM" != "Y" ]] && echo "Annulé." && exit 1

# Boucle pour le clonage
for i in $(seq "$START_NUM" "$END_NUM"); do
    NUM=$(printf "%02d" "$i")
    CTID=$((BASE_ID + i))
    HOSTNAME="CTF1-Etudiant${NUM}"
    IP="${NETWORK_BASE}.1${NUM}${MASK}"

    echo "Clonage CT $TEMPLATE_ID → $CTID ($HOSTNAME)..."

    if pct status "$CTID" &>/dev/null; then
        echo "Le CT $CTID existe déjà. Ignoré."
        continue
    fi

    if pct clone "$TEMPLATE_ID" "$CTID" --hostname "$HOSTNAME" --full true --
storage "$STORAGE"; then
        echo "Clone $CTID ($HOSTNAME) créé avec succès."
    else
        echo "Échec du clonage pour $CTID ($HOSTNAME)."
        continue
    fi

    pct set "$CTID" -net0 name=eth0,bridge="$BRIDGE",ip="$IP",gw="$GATEWAY"
    echo "Adresse IP $IP définie + passerelle $GATEWAY."
    pct set "$CTID" --nameserver "$NAMESERVER"

```

```
echo "DNS défini sur $NAMESERVER."

echo "Démarrage du CT $CTID ($HOSTNAME)..."
if pct start "$CTID"; then
    echo "CT $CTID démarré."
else
    echo "Échec au démarrage du CT $CTID."
    continue
fi

for j in {1..10}; do
    if [[ "$(pct status "$CTID")" == *"running"* ]]; then
        echo "CT $CTID prêt (état 'running' détecté)."
        break
    fi
    echo "En attente du statut 'running' pour $CTID..."
    sleep 2
done

if [[ "$(pct status "$CTID")" != *"running"* ]]; then
    echo "Le CT $CTID n'est pas prêt. Passage au suivant."
    continue
fi

echo "Exécution de $SCRIPT_IN_LXC dans $CTID..."
pct exec "$CTID" -- bash "$SCRIPT_IN_LXC" <<< "$i"
echo "Script exécuté dans $CTID."
echo "-----"
done
```
