Rapport Projet : grimoire

Guillaume RAISON - MCS 26.3

Date du rapport : 09/01/2026

1. Identification de la vulnérabilité

La vulnérabilité est un **Use-After-Free (UAF)** situé dans la gestion de la mémoire du "sortilège" (`active_spell`).

- **Fichier :** `grimoire.c`
- **Localisation :** Ligne 70, dans l'option 3 "Write in Diary". `c if (active_spell) free(active_spell);`

2. Analyse et calcul de l'offset

Le programme présente des caractéristiques spécifiques qui facilitent l'exploitation :

1. **Mémoire cxécutable :** La fonction `setup_spell_memory` utilise `mprotect` pour rendre la zone du heap exécutable RWX. Nous n'avons donc pas besoin de ROP, un simple shellcode suffit.
2. **Taille fixe :** Les allocations pour le sort (`choice 1`) et pour le journal (`choice 3`) sont identiques : `CHUNK_SIZE` (128 octets).
3. **Comportement de l'allocateur :** Sur Linux, l'allocateur mémoire (glibc malloc) a tendance à recycler le dernier chunk libéré s'il correspond à la taille demandée (LIFO / tcache).

3. Stratégie d'exploitation

L'objectif est d'obtenir un shell **Root** (UID 0). Comme le binaire est SUID, nous devons appeler `setuid(0)` avant de lancer `/bin/sh`.

Étapes de l'attaque : 1. **Allocation :** Allouer un sort légitime. `active_spell` pointe vers l'adresse `A`. 2. **Libération :** Écrire "Voldemort" dans le journal. * Cela déclenche `free(active_spell)`. * L'adresse `A` est marquée comme libre, mais `active_spell` pointe toujours vers `A`. 3. **Réallocation & injection:** Écrire une nouvelle entrée dans le journal sans "Voldemort". * `malloc(128)` va recycler l'adresse `A` pour le journal. * Nous écrivons notre **Shellcode** dans ce buffer. * Maintenant, `active_spell` pointe vers notre Shellcode. 4. **Exécution (Choix 2) :** Lancer le sort. Le programme exécute le contenu de l'adresse `A`, qui est maintenant notre code malveillant.

Payload utilisé : * `setreuid(0, 0)` : Pour restaurer les privilèges root effectifs. * `execve("/bin/sh", ...)` : Pour lancer le shell.

4. Résultats

L'exécution du script `exploit.py` automatise ce processus. * Le script détecte le prompt. * Il effectue la manipulation du heap. * Il injecte le shellcode généré par `pwntools`. * Le message final confirme l'obtention d'un shell avec les droits root (`uid=0`).