Rapport Projet : Grimoire

Auteur : Guillaume RAISON **Module :** Exploitation Binaire (Use-After-Free)

1. Analyse de la vulnérabilité

La faille se situe dans la gestion de la mémoire du pointeur `active_spell` (Use-After-Free).

- **L'erreur :** Dans l'option 3 ("Write in Diary"), si l'utilisateur écrit "Voldemort", le programme appelle `free(active_spell)`.
- **La persistance :** Le programme ne remet pas le pointeur `active_spell` à `NULL` après la libération.
- **La conséquence :** Le pointeur reste "dangling" (il pointe vers une zone mémoire considérée comme libre). Si l'on effectue une nouvelle allocation (via le journal), l'allocateur (malloc) réutilise cette même zone mémoire.

2. Stratégie d'Exploitation

Le but est d'injecter un **Shellcode** qui nous donne les droits root, puisque le binaire possède le bit SUID.

A. Manipulation du Heap

1. **Allocation :** J'utilise l'option 1 pour allouer le sort. Le pointeur `active_spell` est initialisé.
2. **Libération (UAF) :** J'utilise l'option 3 avec le mot "Voldemort" pour libérer la mémoire, tout en gardant le pointeur actif.
3. **Réallocation & Injection :** J'utilise l'option 3 pour écrire une nouvelle entrée. `malloc` me redonne la même adresse. J'y écris mon payload.

B. Construction du Payload

La zone mémoire est rendue exécutable par le programme (`mprotect`). Je peux donc y placer directement des instructions assembleur.

- **NOP Sled :** J'ajoute 60 octets de `\x90` (NOP) au début. Cela permet de "glisser" vers le code même si l'alignement mémoire varie légèrement.
- **Shellcode :** J'utilise un shellcode Linux x64 classique :
  1. `setuid(0)` : Pour restaurer les privilèges root effectifs.
  2. `execve("/bin/sh", ...)` : Pour lancer le terminal.

3. Exécution

J'utilise l'option 2 ("Cast Spell"). Le programme exécute la fonction située à l'adresse `active_spell`. Comme j'ai remplacé son contenu par mon shellcode, c'est mon code qui s'exécute avec les privilèges du propriétaire du fichier (Root).

4. Résultat

L'exécution du script `exploit.py` permet d'obtenir un shell interactif. La commande `id` retourne `uid=0(root)`, confirmant la réussite de l'élévation de privilèges.