Filière Sécurité des réseaux et infrastructures informatiques

CSC_4CS03_TP : Cyberattaques : menaces et mises en œuvre

Exploitation des chaînes de format (Format String)

Durée : 2 jours

Sujet : Exploitation d'une vulnérabilité de type *Format String* et mise en œuvre d'une attaque ROP dans un programme 64 bits sous Linux

L'objectif de ce travail est d'analyser et d'exploiter une vulnérabilité de type *format string* présente dans un programme 64 bits sous Linux, puis de construire une chaîne d'attaque permettant l'exécution d'un *shellcode* en utilisant la technique *Return-Oriented Programming (ROP)*.

L'attaque devra être menée en plusieurs étapes, en réutilisant successivement la même fonction vulnérable, ce qui impose la création d'une boucle d'exploitation.

1. Analyse préliminaire du programme

Vous devez commencer par **tester le programme fourni** afin de confirmer la présence d'une vulnérabilité de type *format string*. L'analyse consistera notamment à :

- Déterminer si les données fournies en entrée sont utilisées comme format dans une fonction d'affichage.
- Vérifier si la vulnérabilité permet d'exfiltrer des données depuis la pile ou de modifier des valeurs en mémoire.

Il est attendu que vous identifiez et justifiez la nature exacte de la vulnérabilité détectée.

2. Extraction d'informations à l'aide d'un débogueur

L'utilisation d'un débogueur (par ex. gdb) est obligatoire pour extraire les informations nécessaires à la construction de l'attaque.

Vous devez en particulier :

- Examiner les données exfiltrées via la vulnérabilité et **analyser précisément les mots n°7 et n°27** dans la sortie obtenue.
- Déterminer l'organisation exacte de la pile lors de l'exécution de la fonction vulnérable.

Les observations doivent être documentées et justifiées.

3. Calcul des distances nécessaires à la construction de la chaîne ROP

À l'aide du débogueur, vous devez déterminer :

1. La distance entre l'adresse de retour de la fonction vulnérable et l'adresse d'un gadget ROP de type : pop rdi ; ret
2. La distance entre cette même adresse de retour et l'entrée de la fonction system dans la table PLT.

Ces deux informations sont indispensables pour construire la chaîne ROP finale.

4. Préparation de la charge utile

Puisque l'attaque s'appuie sur une exécution répétée de la fonction vulnérable (makePing), et que les données situées au-dessus du *frame* de cette fonction ne sont pas réutilisées par la suite, vous devez :

- Placer la chaîne **"/bin/sh\0"** dans une zone stable de la pile, par exemple à l'emplacement représenté par le **mot n°28** parmi les informations exfiltrées.
- Insérer le *shellcode* en mémoire dans l'ordre inverse pour faciliter sa reconstruction au fil des itérations.

Ces choix doivent être expliqués et justifiés.

5. Construction de la boucle d'exploitation

Étant donné que l'entrée vulnérable n'est disponible qu'une seule fois par exécution, il est nécessaire de mettre en place une boucle d'exploitation consistant à :

- Réappeler la fonction vulnérable à chaque étape (Vous pouvez vous inspirer du script Python fourni avec le devoir).
- Modifier progressivement des portions de l'adresse de retour.
- Préparer la chaîne ROP finale pour la dernière itération.

Vous devrez démontrer de manière précise comment la fonction makePing peut être réinvoquée durant l'exploitation et pourquoi cette technique est indispensable.

Indication :
L'adresse de retour est située dans la fonction main, à une distance de cinq octets de l'appel à *makePing*. Par ailleurs, lorsqu'une exécution utilise l'option PIE (Position Independent Executable), l'adresse exacte des instructions varie à chaque lancement du programme. Toutefois, l'octet de poids faible de ces adresses demeure inchangé d'une exécution à l'autre.
Par conséquent, il suffit de modifier uniquement l'octet de poids faible à l'emplacement où l'adresse de retour est sauvegardée pour rediriger efficacement le flux d'exécution.

6. Modification finale de l'adresse de retour

Il n'est possible de modifier l'adresse de retour **qu'à la fin**, et en une seule modification effective. La méthodologie attendue consiste à :

- Utiliser le gadget *ret* le plus proche de l'adresse de retour (par exemple celui se trouvant en fin de la fonction *main*).
- Expliquer le rôle et l'utilité du gadget *ret* dans le cadre d'une attaque ROP.
- Modifier uniquement **l'octet de poids faible** de l'emplacement de l'adresse de retour, ce qui constitue la méthode la plus simple et la plus fiable dans ce contexte.

La forme finale du shellcode à exécuter en utilisant ROP est le suivant :

- Adresse d'un gadget « RET »
- Adresse d'un gadget « POP RDI ; RET »
- Adresse de la chaine « /etc/sh\0 »
- Adresse de l'entrée correspondante à la fonction *system* dans PLT

Vous devez impérativement expliquer pourquoi seule la modification de l'octet de poids faible est suffisante et comment elle permet de rediriger le flux d'exécution vers le gadget souhaité.

Livrables attendus

Vous devez fournir :

- Une analyse complète de la vulnérabilité.
- Les sorties pertinentes du débogueur, accompagnées de leurs interprétations.
- Le calcul détaillé de toutes les distances utilisées dans l'attaque.
- Les différentes charges utiles intermédiaires.
- La charge utile finale accompagnée d'une explication complète de son fonctionnement.
- Une démonstration de l'obtention d'un shell.