Obfuscated CrackMe

Groupe Crack_en:

- BAULU Evan
- MORALES Raphaël
- FRANÇOIS Enzo
- BRIOL--DUHALDE Damien

Resolution BASH command

```
minute=$(date +"%M") && ascii1=$(printf "\\$(printf "%o" "$((${minute#0} + 0x20))")") && ascii2=$(printf "\\$(printf "%o" "$((${minute#0} + 0x43))")") && echo "crack_en$ascii1$HOME$ascii2"
```

Exécuter le script ./check.sh pour la solution

Table of contents

- Obfuscated CrackMe
  - Resolution BASH command
  - Table of contents
  - Presentation
  - Control Flow obfuscation
  - Plan fonctions anonymes
  - Encrypted Obfuscation
  - Time Based Password
  - Anti Debugger
  - Options gcc
  - Dead code
  - Pistes d'amélioration
  - Structure finale du mot de passe

Presentation

Élaboration un "crackme" éducatif, un défi de reverse engineering, où l'exécutable affiche "OK\n" sur STDOUT et retourne zéro lorsque le bon mot de passe est entré.

- Exigences de l'exécutable
  - Livrable : un exécutable ELF pour Linux x64, avec une taille maximale de 16KB.
  - Le fichier binaire doit un script ./BUILD.sh doit le compiler indépendamment.
- Spécification du CrackMe
  - Le mot de passe doit être passé via `argv[1]` OU `argv[2]` OU `argv[1] et argv[2]`.
  - L'exécutable doit afficher "OK\n" sur STDOUT et retourner zéro si le mot de passe est correct.
  - Doit fonctionner sur toute distribution Linux moderne AMD64

- Techniques d'obfuscation (objectif 4) :
  - Control Flow obfuscation
  - Encrypted Obfuscation
  - Time Based Password
  - Anti Debugger
  - Option GCC
  - Dead code
- Ethique
  - Aucun comportements malveillants dans l'exécutable.

Control Flow obfuscation

L'objectif de cette méthode d'obfuscation est de générer des comportements perturbateurs dans le code. Cette méthode a été surtout été utilisé dans les fonctions d'assembler pour complexifier toute tâche de reverse. De nombreux sauts de fonctions permettent d'embrouiller le comportement du code.

Plan fonctions anonymes

- function001 -> inutile (manipule les registres mais n'a pas d'impact sur le comportement général du crackMe)
- function002 -> chaine de caractère "crack_en" formant la 1ère partie du code
- function003 -> inutile (manipule les registres mais n'a pas d'impact sur le comportement général du crackMe)
- function004 -> chaine de caractère "$ε₳₹¢hHıḋ€η₭e¥₣or₮he₩i₦" -> search hiden key for the win . Arg1 contient la clé et arg2 contient une fausse clé; Cette clé sert à déchiffre tout cryptage XOR du notre programme
- function005 -> caractère changeant chaque minute (offset 0x20) formant la 2ème partie du code
- function006 -> inutile (manipule les registres mais n'a pas d'impact sur le comportement général du crackMe)
- function007 -> Méthode de chiffrement XOR
- function008 -> permet de récupérer la chaine de caractère "HOME"
- function009 -> récupération de la vairable d'environnement saisie en argument, formant la 3ème partie du code
- function011 -> caractère changeant chaque minute (offset 0x43) formant la 4ème partie du code
- function012 -> my_strcmp avec dead code
- test_bloc1 -> vérification de la longueur de la chaine de caractère version 1
- test_bloc2 -> vérification de la longueur de la chaine de caractère version 2
- test_bloc3 -> vérification de la longueur de la chaine de caractère version 3

Encrypted Obfuscation

Pour réussir le CrackMe, les utilisateurs devront retrouver le code crypté et sa méthode de cryptage pour valider le challenge. L'entrée de l'utilisateur est crypté à l'aide de la clé de chiffrement "$ε₳₹¢hHıḋ€η₭e¥₣or₮he₩i₦". Une fois chiffré, la chaine de caractère est comparée au 1er bloc du mot de passe du crackMe

Time Based Password

Cette méthode permet de changer le mot de passe en fonction de l'heure d'execution. En ayant un mot de passe changeant à chaque minutes, nous complexifions grandement le travail du reverse engineering. En fonction de l'offset choisi, nous parcourons la table ASCII.

Présentation de la méthode

Anti Debugger

Cette méthode utilise une commande UNIX (donc inutile sur Windows). Cette commande va rechercher dans les programmes en cours d'utilisation s'il y a un quelconque debugger (liste saisie en dur dans le code) et va s'arrêter s'il en trouve.

Options gcc

Pour dissimuler des techniques d'obfuscation, on utilise les options de gcc :

```
gcc -no-pie -s -fvisibility=hidden ./main_CrackMe.c ./fonctionsASM/*.o -o
main -Wl,--strip-all
```

- -no-pie permet au compilateur de produire un exécutable position-dépendant, ce qui signifie que l'exécutable attendra d'être chargé à une adresse mémoire fixe
- -s sert à supprimer toutes les informations de symbole du fichier exécutable généré
- -fvisibility=hidden réduit la visibilité des symboles, ce qui peut limiter l'exposition des fonctions et des variables aux outils de reverse engineering.
- -Wl,option permet de passer option directement à l'éditeur de liens. Ici, --strip-all est l'option passée à l'éditeur de liens, qui lui demande de supprimer toutes les informations de débogage et de symbole du fichier exécutable généré.

Dead code

Comme son nom l'indique, le deadcode ajout du code mort, n'impactant pas le fonctionnement du code.

Pistes d'amélioration

- Incorporation de Code Auto-Modifiant: Le code qui modifie son propre comportement en cours d'exécution peut semer la confusion et compliquer le reverse engineering.
- Utilisation de Macros et de Fonctions Inline Complexes: L'utilisation intensive de macros et de fonctions inline peut rendre le code source visuellement complexe et difficile à suivre.
- Packers et Compresseurs: Utiliser des packers ou des compresseurs pour réduire la lisibilité du code binaire. Cependant, des outils comme Ghidra sont souvent capables de décompresser ou de déballer ces programmes.
- Crypter les chaines de caractères présentes dans les .s pour éviter une résolution trop rapide.

Structure finale du mot de passe

- Bloc 1 : nom du groupe (crack_en). La comparaison avec l'entrée utilisateur est crypté avec la méthode XOR et la clé $ε₳₹¢hHıd€ŋ₭e¥₣or₮he₩i₦.
- Bloc 2 : caractère changeant avec un offset de 0x20
- Bloc 3 : Variable d'environnement HOME de l'utilisateur
- Bloc 4 : caractère changeant avec un offset de 0x43