**Binary Security**
Übungsblatt 6

# Hackerpraktikum

FAU Erlangen-Nürnberg, Lehrstuhl für Informatik 1

### Basic Reverse Engineering (Aufgabe 1)

Im StudOn finden Sie die beiden Programme `reverse-me.32` und `reverse-me.64`, die sich in ihrer Architektur unterscheiden (IA32 und AMD64) nicht aber in ihrer Funktion. Es handelt sich um drei Passwort-basierte Challenges, die sie lösen können indem Sie jeweils eines der Programme betrachten.

- Wie lautet das Passwort für Challenge 1? (0.25 P.)
  **Tipp:** Durchsuchen Sie das Programm mit einem Werkzeug wie `strings` nach Zeichenketten.
- Wie lautet das Passwort für Challenge 2? (0.75 P.)
  **Tipp:** Schauen Sie sich den Disassembler der Funktion `checkMedium` mit einem statischen Analyse-Tool wie `objdump`, IDA Pro oder Hopper an, um die Passwortgenerierung nachzuvollziehen.
- Wie lautet das Passwort für Challenge 3? (1.5 P.)
  **Tipp:** Analysieren Sie das Programm dynamisch mit einem Debugger wie `gdb`, um `checkHard` zur Laufzeit zu betrachten. Sie müssen den Passwort-Algorithmus nicht komplett verstehen oder nachbauen, sondern können sich das Passwort im Debugger ausgeben lassen.

2.5 P.

### Basic Exploitation (Aufgabe 2)

Die Programme `hack-me.32` und `hack-me.64` erwarten als Argument einen String und geben dann die Länge des Strings sowie den String selbst aus.

1. Bringen Sie beide Programme mit einer langen Eingabe zum Absturz. (0.25 P.)
2. Erklären Sie, wie es zum Absturz der Programme kommt. Welche Anweisung ist fehlerhaft und was passiert mit dem *Return Instruction Pointer*? Zeichnen Sie den Stack vor und nach dieser Anweisung als `ASCII`-Art. (0.75 P.)
3. Manipulieren Sie mit Ihrer Eingabe den Programmfluss von `hack-me.32` so, dass die Funktion `secret` ausgeführt wird:
   a) Bestimmen Sie die Adresse der Funktion `secret` in `hack-me.32`. (0.25 P.)
   b) Schreiben Sie ein Skript, das die Adresse hinreichend oft konkateniert. (0.25 P.)
   **Tipp:** Achten Sie dabei auf die Little-Endian Darstellung von Adressen bei Intel x86.
   c) Injizieren Sie die Ausgabe Ihres Skripts als Eingabe in `hack-me.32` so dass schließlich `secret` ausgeführt wird. (0.5 P.)
4. Injizieren Sie einen ähnlichen Angriffsvektor in das Programm `hack-me.64` um dort ebenfalls `secret` zur Ausführung zu bringen. (0.5 P.)
   **Tipp:** Achten Sie auf Nullbytes bei 64-bit Adressen, da diese das Ende von `strcpy()` bedeuten.

2.5 P.

**Game Labrys (Aufgabe 3)**

Laden Sie sich das Spiel "Labrys" aus dem StudOn herunter. Für Hinweise zur Installation und Steuerung des Spiels lesen Sie bitte die `README.txt`. Neue Level können Sie mit einem Text-Editor erstellen, bspw. indem Sie die Datei `./level5/labyrinth` editieren. Alle folgenden Aufgaben können Sie wahlweise mit `labrys.32` oder `labrys.64` lösen.

1. **Cracking**: Sie wollen für das Spiel nicht zahlen und entscheiden sich, es zu cracken.
   a) *Keygen*:
      i. Erstellen Sie einen gültigen Lizenzschlüssel für das Spiel. (1 P.)
      ii. Schreiben Sie einen Key-Generator, der mehrere Schlüssel generiert. (1 P.)
   b) *Patch*: Modifizeren Sie das Spiel, so dass es auch ohne Lizenz startet. (1 P.)
2. **Cheating**: Sie verschaffen sich Vorteile gegenüber Mitspielern indem Sie cheaten.
   a) *Wallhack*: Modifizieren Sie das Spiel, so dass Sie horizontal durch Wände laufen können ohne dabei herunterzufallen. (1 P.)
   **Tipp:** Verändern Sie die Methode(n) `collide`.
   b) *Flyhack*: Modifizieren Sie das Spiel, so dass Sie fliegen können. (1 P.)
   **Tipp:** Verändern Sie die Methode `gravity`.
   c) *Speedhack*: Modifizieren Sie das Spiel, so dass Sie schneller werden. (1 P.)
   **Tipp:** Verändern Sie den Multiplikator in `step_forward` oder `save_state`.
3. **Exploitation**: Sie verteilen selbst erstellte Levels um Ihre Mitspieler zu exploiten.
   a) *Shellcode Injection*: Gestalten Sie Level 5 derart, dass Schadcode auf dem Stack des Spielers ausgeführt wird. Beachten Sie, dass ASLR eingeschaltet und NX ausgeschaltet sind. (4 P.)
   **Tipp:** Verwenden Sie zur Demonstration den Shellcode von den Folien oder beliebigen Shellcode aus dem Internet, um `/bin/sh` auszuführen.
   b) *Return Oriented Programming*: Verwenden Sie nun die `labrys_rop.{32,64}` Binaries und beachten Sie, dass diese mit NX-Unterstützung kompiliert sind, d.h. starten Sie beliebigen Schadcode ohne ausführbaren Stack. (5 P.)
   **Tipp:** Tools wie `github.com/JonathanSalwan/ROPgadget` und `github.com/sashs/Ropper` unterstützen Sie bei der Suche nach Gadgets.

15 P.

**Obfuscated Binaries (Aufgabe 4)**

Bei `obfusticated.32` und `obfusticated.64` handelt es sich um eine klassische Reversing-Challenge im Stil von Aufgabe 1. Jedoch wehren sich die Programme aktiv gegen eine statische und dynamische Analyse.

1. Beschreiben Sie mit welchen Techniken sich die Programme schützen. (1 P.)
2. Wo findet die Überprüfung des Passworts statt? Wie lautet das Lösungswort? (3 P.)

4 P.

**Filereader (Aufgabe 5)**

Laden Sie sich das Binary `filereader` und die beiliegende `libc.so.6` aus dem StudOn herunter. Ziel ist es, auf der VM `10.0.23.80:31337` eigenen Code auszuführen und die Flagge gespeichert in `/home/filereader/flag` auszulesen. Dokumentieren Sie außerdem in kurzen Stichpunkten Ihr Vorgehen.

1. Schreiben Sie einen Exploit für das Binary `filereader`, der mit Hilfe der beiliegenden Libc „/bin/sh“ ausführt. (5 P.)
   **Tipp:** Die Python-Bibliothek `https://github.com/Gallopsled/pwntools` bietet viele nützliche Features für Binary-Exploitation.
   **Tipp:** Der Exploit ist deutlich schwieriger mit einer aktuellen Version der Glibc durchzuführen, verwenden Sie deshalb auch zum lokalen Testen des Exploits die beiliegende `libc-2.23`.

2. Führen Sie ihren Exploit remote gegen `Host:10.0.23.80, Port:31337` durch und stehlen Sie die Flagge in `/home/filereader/flag`. (1 P.)

**Hinweis:**

1. Geben Sie sowohl ihren Exploit als auch die Flagge ab. Der Exploit muss remote gegen die VM funktionieren.
2. Die VM hat einen Timeout von 60 Sekunden und schließt danach die Verbindung. Zudem wird alle 10 Minuten ein Hardreset durchgeführt.

6 P.

$2.5 + 2.5 + 15 + 4 + 6 =$ **30 Punkte**

(Hinweis: Die Bearbeitungszeit dieses Übungsblattes beträgt 3 Wochen, so dass es 30 Punkte statt 20 Punkte gibt.)