michael.engel [☺] hs-coburg.de

Praktikum zur Vorlesung
Malware-Analyse und Reverse Engineering
Sommersemester 2017

Praktikumsaufgabe 4

Besprechung am Donnerstag, 27. April 2017

Hinweise: Praktikumsaufgaben müssen in der vorgegebenen Zeit in den eingeteilten Gruppen bearbeitet werden. Die Lösung der Aufgaben wird bewertet, jedes Gruppenmitglied soll Fragen zu den einzelnen Aufgaben beantworten können. Bei Unklarheiten und Rückfragen wird selbstverständlich eine Hilfestellung gegeben.

4.1 Return-Oriented Programming

Nutzen Sie Buffer Overflows mit Hilfe von Return Oriented Programming aus.
Dazu verwenden wir eine Erweiterung des Debuggers gdb, die die Analyse von Programmen und die Entwicklung von Exploits erleichtert: PEDA – Python Exploit Development Assistance https://github.com/longld/peda
Führen Sie zur Vorbereitung zunächst die folgenden Schritte durch:

1. Installieren Sie die Pakete make, nasm und sl:

```
sudo apt-get install -y git gdb sl make nasm
```

2. Installieren Sie das gdb-Plugin peda:

```
git clone https://github.com/longld/peda.git ~/peda
echo "source ~/peda/peda.py" >> ~/.gdbinit
```

3. Deaktivieren Sie ASLR (Address Space Layout Randomization):

```
sudo bash -c 'echo "0" > /proc/sys/kernel/randomize_va_space'
```

   Sie können überprüfen, ob ASLR aktiv ist, indem Sie folgende Befehle ausführen. Bei aktiver ASLR sollten sich nach jedem Aufruf des Linkers “ld” die Adressen der shared libraries ändern.

```
# check current value with
cat /proc/sys/kernel/randomize_va_space # should return 0
# run ld multiple times and check the addresses of the dynamically linked libraries
ldd <your-binary> # --> check the addresses of the dynamically linked libraries
```

Bearbeiten Sie nun die folgenden Aufgabenteile:

1. Schreiben Sie ein C-Programm, das Benutzereingaben mit einer der ausnutzbaren “problematischen” Funktionen (z.B. gets oder scanf) einliest und diese wieder ausgibt (mit printf oder puts).
2. Übersetzen Sie das Programm *mit deaktivierter stack smashing protection* und aktivieren Sie Debug-Symbole (Compiler-Option –g).
3. Überprüfen Sie, dass Ihr C-Programm anfällig für Buffer Overflows ist und finden Sie den Offset des auf dem Stack zu überschreibenden Returnadresse.
4. Starten Sie Ihr Programm mit dem Debugger gdb
   (a) Setzen Sie einen Breakpoint auf die `main`-Funktion
   (b) Starten Sie das Programm
   (c) Finden Sie die Adresse der libc-Funktion `system`
   (d) Finden Sie die Adresse des Strings “sl” in der libc:
   (e) Finden Sie ein ROP-Gadget, das die Adresse dieses Strings in Register `rdi` lädt

   Sie können im gdb Hilfe zu den peda-Funktionen mit dem Befehl “peda” aufrufen. Nützliche Hinweise dazu finden Sie in folgender Präsentation (ab Folie 17):

   http://ropshell.com/peda/Linux_Interactive_Exploit_Development_with_GDB_and_PEDA_Slides.pdf
5. Testen Sie Ihren Exploit (z.B. mit “echo”, wie schon in der vorherigen Aufgabe)
6. Optional: Versuchen Sie, mit Ihrem Exploit eine Shell aufzurufen

   Tip: Wenn Sie den Exploit-String mit “echo” übergeben, nutzen Sie:

   ```
   echo -e "meintollerstringdenichuebergebenwill" > buffer_file
   cat buffer_file - | ./meintollesprogramm
   ```

   Wichtig ist hier das “-” als letzter Parameter des “cat”-Befehls, dieser sorgt dafür, dass die aufgerufene Shell eine Eingabe erhält (ohne das “-” wäre diese nach dem letzten Zeichen des Strings, der an echo übergeben wird, zu Ende und die Shell beendet sich gleich wieder). Ende