michael.engel [☺] hs-coburg.de

Praktikum zur Vorlesung
Malware-Analyse und Reverse Engineering
Sommersemester 2017

Praktikumsaufgabe 1

Besprechung am Donnerstag, 20. Oktober 2016

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.

1.1 Linux in a Strange World

Die Praktikumsaufgaben zu MARE werden zu einem großen Teil in einer Linux-Umgebung ablaufen. Hierzu verwenden wir das in Windows 10 verfügbare "Windows Subsystem for Linux", das eine Linux-Umgebung in Windows bereitstellt, ohne eine virtuelle Maschine zu benötigen. Die aktuelle Umgebung basiert auf Ubuntu Linux 14.04.
Installieren Sie zunächst Ihre persönliche Linux-Umgebung auf Ihrem Laborrechnern anhand der Anleitung unter https://msdn.microsoft.com/de-de/commandline/wsl/install_guide
Beginnen Sie dabei beim Punkt "Run Bash on Windows" (und denken Sie daran, auf der zweite Seite mit "read more" weiterzulesen). Die vorherigen Schritte haben wir bereits für Sie erledigt. Das "bash"-Icon finden Sie bereits auf Ihrem Desktop.
Wichtig: Merken Sie sich den angelegten Benutzernamen und das zugehörige Passwort!

1.2 Installation der wichtigsten Tools

Zur Programmanalyse werden eine Reihe von Werkzeugen benötigt, die evtl. noch nicht in der Linux-Umgebung installiert sind.
Sie haben in Ihrer Linux-Umgebung optional Administratorrechte ("root") – diese gelten aber (sinnvollerweise...) nicht für das umgebende Windows. Installieren Sie mit Hilfe des Programms "apt-get" die Pakete gcc (Compiler) und binutils (Linker, Disassembler usw.) nach (das "$"-Zeichen zeigt das Shellprompt an, nicht mit eingeben!):

```
$ sudo apt-get install gcc
$ sudo apt-get install binutils
```

Beantworten Sie aufkommende "Sind Sie sicher"-Fragen mit "J".

1.3 Programm erstellen

Schreiben Sie ein C-Programm, das folgende Komponenten enthält:

- Eine globale *Konstante* mit Initialisierung "a", eine globale Variable mit Initialisierung "b" und eine globale Variable ohne Initialisierung "c".
- Jeweils eine Funktion **foo** und **bar**. Jede dieser Funktionen soll eine statische lokale Integer-Variable "counter" deklarieren, diese bei jedem Aufruf der Funktion inkrementieren und den Wert zurückgeben.
- Eine main-Funktion, die in einer Schleife **foo** und **bar** aufruft, die Rückgabewerte addiert und diese Summe nach Schleifenende als return-Wert von **main** zurückgibt.

Hinweise: Sie können zum Editieren des Sourcecodes einen üblichen Linux-Editor (vi, joe, emacs – evtl. mit apt-get nachinstallieren) verwenden. Es ist aber auch möglich, einen Windows-Editor zu verwenden, z.B. Notepad.

Dazu können Sie mit Notepad eine Quelldatei z.B. auf Ihrem Desktop unter Windows anlegen. Von Linux aus können Sie auf Dateien in Ihrem Windows-Homeverzeichnis unter dem Pfad **/mnt/c/Users/bla4711s/** zugreifen ("bla4711" steht für Ihre Benutzerkennung!), also z.B. auf Dateien auf dem Windows-Desktop unter **/mnt/c/Users/bla4711s/Desktop** Allgemeine Dokumentation zum Windows Subsystem for Linux finden Sie unter http://msdn.microsoft.com/commandline/wsl/about

1.4 Programm übersetzen und analysieren

Übersetzen Sie nun zuerst Ihr C-Programm (hier "foo.c" genannt) und testen es – "echo $?" gibt den von return in **main** zurückgegebenen Wert aus. Wie lautet die Ausgabe?

```
$ gcc -o foo foo.c
$ ./foo ; echo $?
```

Analysieren Sie nun das Programm mit den Tools nm und readelf wie in Vorlesung 2 gezeigt.

- In welchen ELF-Sektionen liegen die Variablen a, b und c?
- Was fällt Ihnen bei der Variable "counter" auf? Was könnte der Grund für diesen Effekt sein?
- An welchen Adressen liegen die einzelnen ELF-Sektionen?