michael.engel [☺] hs-coburg.de

Praktikum zur Vorlesung
Malware-Analyse und Reverse Engineering
Sommersemester 2017

Praktikumsaufgabe 5

Besprechung am Donnerstag, 4. Mai 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.

5.1 Stack Smashing Protection

Stack smashing protection in gcc verwendet einen speziellen “canary”-Wert, der auf dem Stack zwischen potentiell überlaufenden lokalen Variablen und dem gesicherten Base Pointer und der Returnadresse abgelegt wird. Der canary wird vor einem return aus der Funktion überprüft; im Falle eines veränderten Wertes wird das Programm abgebrochen. Finden Sie eine Methode, stack smashing protection selbst zu implementieren:

1. Unter ausschließlicher Verwendung von C-Code!
2. (Optional) Unter Verwendung von Assembler-Code, den Sie in den erzeugten Assembler-Quelltext einfügen (manuell oder sogar automatisiert mit Hilfe von perl/python/awk etc.). Die gcc-Option “-S” erzeugt die Assembler-Quelltextdatei “foo.s” aus dem C-Quellcode “foo.c”, diese können Sie danach wie eine normale C-Sourcedatei z.B. mit “gcc -o foo foo.s” compilieren.

Können Sie einen Exploit für Ihren Schutz finden?

Arbeiten Sie dazu in Ihrer Gruppe (wenn möglich) ausnahmsweise einmal gegeneinander – Schutz-Entwickler gegen “böse” Hacker. Versuchen Sie, mehrere Iterationen zu realisieren (einfachen Schutz entwickeln – Hacken – besseren Schutz entwickeln – usw.).

Selbstverständlich müssen Sie zum Entwickeln der eigenen stack smashing protection Ihren Quellcode wie bisher mit der gcc-Option “-fno-stack-protector” übersetzen.