ATIS - Software Reverse Engineering

Prof. Dr. Christian Dietrich
dietrich@internet-sicherheit.de

1 Lab: Assembly

Aufgabe

Das Ziel dieser Praktikumsaufgabe besteht darin, sich mit Assembly für die x86-Architektur vertraut zu machen. Dazu soll ein Programm für Intel x86 (IA-32) geschrieben werden, dass die ersten 100 **ungeraden** Ganzzahlen aus der Menge der natürlichen Zahlen (typischerweise als $\mathbb{N}_0$ bezeichnet) aufsummiert.

Abzuliefernde Ergebnisse

- Assemblycode in Textform in Intel-Notation

Hinweise

- Nutzen Sie ausschließlich x86-Instruktionen für 32-bit.
- Nutzen Sie ausschließlich die General-Purpose-Register `eax, ebx, ecx, edx, esi, edi, ebp, esp`; nutzen Sie keinen Hauptspeicher. Das Ergebnis der Berechnung soll am Ende im Register `ebx` liegen.
- Geben Sie ausschließlich den Code für die Berechnungen an und lassen Sie jeglichen "Boilerplate-Code", also Code, der nicht direkt mit der Berechnung zu tun hat, weg.
- Geben Sie den Assemblycode in Intel-Notation an.
- Die SREVM enthält den Assembler **keystone** (`www.keystone-engine.org`), der von der Kommandozeile und aus Python heraus angesprochen werden kann. Sie können keystone (oder auch jeden beliebigen anderen Assembler) benutzen, um ihre Lösung assemblieren zu lassen. keystone besitzt das Kommandozeilentool kstool, das sich zur rudimentären Syntaxprüfung des Assemblycodes eignet:

```
1 # Example of a correct instruction
2 $ kstool x32 'mov eax, 1'
3 mov eax, 1 = [ b8 01 00 00 00 ]
4
5 # Example of an error: copy is not a valid mnemonic
6 $ kstool x32 'copy eax, 1'
7 ERROR: failed on ks_asm() with count = 0, error = 'Invalid mnemonic (
      KS_ERR_ASM_MNEMONICFAIL)' (code = 514)
```

  Die Ausgabe des Tools beinhaltet unter anderem den Maschinencode in Hexadezimaldarstellung.

- In der Vorlesung wurden Mnemonics zu den Bereichen data movement und arithmetics besprochen, die zur Lösung der Aufgabe verwendet werden können.