RUB

RUHR-UNIVERSITÄT BOCHUM

1. Übung Systemsicherheit

Niklas Entschladen (108017221280), Dennis Rotärmel (108017215383), Tobias Ratajczyk (108017237071)

Deadline: April 23, 2019.

1 1. Übung Systhemsicherheit

1.1 Grundlagen des Debuggens

Bis zur Eingabe ist die Zahl "Input" 0. In Zeile 8, also vor der Ausführung dieser Zeile, ist die Variable Input 23. Hier kann ein Breakpoint mit `b <Zeile>` gesetzt werden. Danach wird die "super_complicated_function" ausgeführt. Diese wird 4 mal aufgerufen und alle 2 "Steps" ändert sich Input. Den nächsten step erzwingt man mit dem Befehl `s` bzw. `step`. Den Wert der Variable erhält man mit `print input`.
$23 \rightarrow 230 \rightarrow 644 \rightarrow 1472 \rightarrow 3144$
3144 ist dann der letzte Wert, bis das Programm durchgelaufen ist.

1.2 Manipulation des Programmzustandes

In Zeile 8, wo beim nächsten "Step" die if-Abfrage ausgeführt wird, muss die Variable "access_level" auf 1 verändert werden. Dies wird mit dem Befehl `set access_level = 1` erreicht. Dieser Befehl verändert während der Ausführung des Programms die angegebene Variable. Die Flag aus der Funktion lautet `6952fb0c8b0ecf136b28705536794ced1c419599`

1.3 Reverse Engineering 101

```
(gdb) disassemble calculate
Dump of assembler code for function calculate:
   0x000000000000076a <+0>:     push   %rbp
   0x000000000000076b <+1>:     mov    %rsp,%rbp
   0x000000000000076e <+4>:     mov    %edi,-0x4(%rbp)
   0x0000000000000771 <+7>:     mov    %esi,-0x8(%rbp)
   0x0000000000000774 <+10>:    mov    -0x8(%rbp),%eax
   0x0000000000000777 <+13>:    add    %eax,-0x4(%rbp)
   0x000000000000077a <+16>:    mov    -0x4(%rbp),%eax
   0x000000000000077d <+19>:    imul   -0x8(%rbp),%eax
   0x0000000000000781 <+23>:    mov    %eax,-0x4(%rbp)
   0x0000000000000784 <+26>:    mov    -0x4(%rbp),%eax
   0x0000000000000787 <+29>:    pop    %rbp
   0x0000000000000788 <+30>:    retq
End of assembler dump.
```

Dieser "Assembler Dump" kann erreicht werden, indem man folgende Befehle eingibt:

```
gcc exercise03
gdb a.out
disessemble calculate
```

Der Befehl **gcc** ist ein Compiler, der eine C++ Datei nimmt und diese kompiliert. Nach dem Kompilieren wird der Debugger mittels **gdb** aufgerufen. Mit **disessemble** wird die angegebene Funktion auf Assembler-Level wiedergegeben.