UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR IT-SICHERHEIT

Praktikum

GDB Cheat Sheet

Cybersecurity im Sommersemester 2023

Jan Wichelmann, Anja Köhl

Allgemein

Die Interaktion mit GDB läuft über eine Terminal-ähnliche Oberfläche, in die Befehle eingegeben und per Entertaste ausgeführt werden können. Es ist ebenfalls möglich, bereits einmal eingegebene Befehle durch Drücken der Pfeiltasten erneut aufzurufen, oder unvollständige Befehle mit der Tab-Taste zu vervollständigen.

Darüber hinaus hat GDB einige Besonderheiten:

- Viele wichtige Befehle haben eine Kurzform (z.B. `i r` statt `info registers`), die meistens aus den ersten Buchstaben besteht. Auf diesem Cheat-Sheet wird jeweils die Langform benutzt.
- Wenn ohne Eingabe eines Befehls die Entertaste gedrückt wird, wird der zuletzt ausgeführte Befehl wiederholt. Dies ist z.B. hilfreich, wenn man mehrere Einzelschritte nacheinander machen möchte.

Konfiguration

Umschalten der verwendeten Syntax zur Laufzeit:

```
(gdb) set disassembly-flavor <type>
```

Um Einstellungen permanent zu speichern, legen Sie die Datei `~/.gdbinit` an und schreiben Sie dort die Befehle zur Änderung der Optionen hinein.

Für das Praktikum wird die folgende Einstellung dringend empfohlen:

```
set disassembly-flavor intel
```

Zusätzlich zur manuellen Konfiguration können auch fertige Konfigurationen eingesetzt werden, die die Arbeit mit GDB vereinfachen und die standardmäßig eher karge Oberfläche mit vielen Informationen anreichern. Ein bekannter Vertreter ist hier *GEF*[1]. Die Installation ist denkbar einfach: Es genügt, einen der auf der Hauptseite aufgelisteten Befehle einzugeben, um die aktuelle Version herunterzuladen und in die `~/.gdbinit`-Datei zu schreiben.

[1] `https://github.com/hugsy/gef`

Start, Stop, Hilfe

GDB starten, um das Programm `a.out` zu debuggen:

```
root@kali:~$ gdb ./a.out
```

Das Programm `a.out` selbst wird hierbei noch nicht gestartet.

Ausführung von `a.out` beginnen:

```
(gdb) start [<arg1> [<arg2> [...]]]
```

Die Argumente sind hierbei die von `a.out` erwarteten.

GDB beenden:

```
(gdb) quit
```

Hilfe anzeigen:

```
(gdb) help
```

Hilfe für bestimmten Befehl anzeigen:

```
(gdb) help <command>
```

Informationsbeschaffung

Auflisten aller enthaltenen Funktionen:

```
(gdb) info functions
```

Disassemblierung

Disassemblierung einer Funktion:

```
(gdb) disassemble <function_name>
```

Wenn keine Funktion angegeben wird, wird die aktuell ausgeführte disassembliert.

Kurzform: `disas`

Debugging

Breakpoint setzen:

```
(gdb) break <function_name|addr>
```

Hier ist entweder ein Funktionsname oder eine hexadezimale Adresse mit vorangestelltem `*` anzugeben.

Alternativ kann man auch einen Offset einer Funktion angeben:

```
(gdb) break *<function_name> + <offset>
```

Der Stern vor dem Namen und die Leerzeichen sind wichtig!

Breakpoints anzeigen:

```
(gdb) info breakpoints
```

Breakpoint deaktivieren:

```
(gdb) disable <break_point_ID>
```

Zur Aktivierung das `disable` durch `enable` ersetzen. Mittels `delete <ID>` werden Breakpoints entfernt.

Fortsetzen des Programms bis zum nächsten Breakpoint:

```
(gdb) continue
```

Fortsetzen des Programms bis zum Funktions-/Programmende:

```
(gdb) finish
```

Ausführen der nächsten Instruktion:

```
(gdb) stepi
```

Kurzform: `si`

Ausführen bis die aktuelle Funktion endet:

```
(gdb) finish
```

Kurzform: `fin`

Letzterer Befehl kann nützlich sein, wenn die Ausführung Sie in eine Funktion hineinführt, die Sie nicht interessiert (beispielsweise `printf`).

Register auslesen

Alle oder ein bestimmtes Register auslesen:

```
(gdb) info register [<name>]
```

Insbesondere kann als Registerbezeichnung `eflags` gewählt werden, um alle gesetzten Flags anzuzeigen.

Speicher auslesen

Speicher an `<addr>` auslesen:

```
(gdb) x/<count>[<size>]<format> <addr>
```

- `<count>` gibt die Anzahl der auszulesenden Blöcke an.
- `<size>` (optional) gibt den auszulesenden Datentypen an:
  - `b`: Byte (8 Bit)
  - `h`: Half Word (16 Bit)
  - `w`: Word (32 Bit)
  - `g`: Giant Word (64 Bit)
- `<format>` gibt das gewünschte Format an, z.B.:
  - `x`: Hexadezimal
  - `d`: Dezimal
  - `u`: Dezimal (vorzeichenlos)
  - `t`: Binär
  - `f`: Gleitkommazahl
  - `a`: Addresse

  - `c`: Char
  - `s`: String
  - `i`: Instruktion (führt zu Disassemblierung)

Beispiele:

- Lese vier 64-Bit-Werte an Adresse `0x12345678` und gib diese in Hexadezimalform aus:

```
(gdb) x/4gx 0x12345678
```

- Lese einen String von der Adresse, auf die `rax` gerade zeigt:

```
(gdb) x/s $rax
```

Weiterführende Links

- `http://sourceware.org/gdb/current/onlinedocs/gdb/`