Razhroščevalnik

Seminarska naloga pri predmetu Sistemska programska oprema
Mentor: doc. Tomaž Dobravec

Jan Mrak
Fakulteta za računalništvo in informatiko, Univerza v Ljubljani

14. januar 2025

Povzetek

Predstavitev implementacije in delovanja rezhroščevalnika na različnih platformah in za različne jezike.

1 Uvod

Razhroščevalnik je program, ki nam omogoča testiranje in upravljanje nekega drugega programa. Omogoča nam branje in pisanje spomina in registrov, poljubno ustavljanje programa, izvajanje po vrsticah ali ukazih in drugo. Poznamo različne razhroščevalnike, kot so GDB, LLDB, x64dbg, ki so bolj generalni. Obstajajo pa tudi drugačni razhroščecalniki, kot je Valgrind, ki nam omogoča pregled nad pomnilnikom procesa in recimo zaznava uhajanje spomina (angl. memory leak).

2 Razhroščevalnik na sistemu Linux

Na Unix in Unix-like sistemih je ponujen sistemski klic

```
long ptrace(enum __ptrace_request op, pid_t pid, void *addr, void *data);
```

, ki nam omogoča, da lahko dostopamo do prcesa podanega s `pid`. Da pa bo sistemski klic uspel, mora proces, do katerega želimo dostopati, dovoliti dostop do njega. To pa lahko naredi s klicem sistemske funkcije `ptrace`:

```
// pid, addr in data argumenti so ignorirani
ptrace(PTRACE_TRACEME, 0, NULL, NULL);
```

Po tem bo razhroščevalnik lahko dostopal do tega procesa.

Razhroščevalnik lahko sam ustvari proces, ki potem pokliče `ptrace` z argumentom `PTRACE_TRACEME`,

```
int pid = fork();
if (pid == 0) {
    ptrace(PTRACE_TRACEME, 0, NULL, NULL);
    execve(...);
}
```

lahko pa se priklopi na nek obstoječi proces z uporabo `PTRACE_ATTACH`, ki pošlje signal `SIGSTOP`, da se proces ustavi, ali pa `PTRACE_SEIZE`, ki ne ustavi procesa.

```
ptrace(PTRACE_ATTACH, pid, NULL, NULL);
// ali
ptrace(PTRACE_SEIZE, pid, NULL, PTRACE_O_flags);
```

Če želimo ustaviti proces, ga lahko ustavimo kadarkoli s klicem `ptrace` in za argument `op` izberemo `PTRACE_INTERRUPT`. Ko pa proces ustavimo, imamo na voljo veliko različnih možnosti za upravljanje s procesom.

Možnosti za pridobivanje in upravljanje z informacijami:

- `PTRACE_PEEKDATA` ali `PTRACE_PEEKTEXT`, ki nam omogočata, da beremo iz procesovega spomina
- `PTRACE_POKEDATA` ali `PTRACE_POKETEXT`, ki nam omogočata, da pišemo v spomin procesa
- `PTRACE_GETREGS` ali `PTRACE_GETFREGS`, ki nam omogočata, da preberemo splošno namenske registre ali registre za delanje s plavajočo vejico
- `PTRACE_SETREGS` ali `PTRACE_SETFREGS`, podobno kot pri prejšnjem primeru, dobimo dostop do registrov in v njih lahko zapišemo vrednosti
- `PTRACE_GETSIGINFO`, ki pridobi informacije o signalu, ki je ustavil proces
- `PTRACE_PEEKSIGINFO`, enako pridobi informacije o signalu, vendar ga ne vzame iz vrste signalov

Možnosti za upravljanje poteka procesa:

- `PTRACE_CONT`, ki znova zažene ustavljen proces, da nadaljuje z delovanjem
- `PTRACE_SINGLESTEP`, ki izvede le en ukaz
- `PTRACE_SYSCALL`, ki se vede kot `PTRACE_CONT`, vendar se preces, ki ga razhroščijemo ustavi tik pred vstopom v sistemski klic, oziroma ob izstopu sistemskega klica
- `PTRACE_KILL`, ki procesu pošlje signal `SIGKILL` in ga tako prisilno zaključi
- `PTRACE_INTERRUPT`, ki ustavi proces

Obstaja še veliko drugih možnosti za delo s procesom, ki pa so razložene v priročniku man za ptrace [4].

2.1 DWARF format

Razhroščevalnik nam ponavadi ponujajo tudi neke dodatne možnosti in olajšave pri razhroščevanju programa, ki pa so omogočene kadar imamo na voljo informacije o programu oz, razhroščevalne informacije (angl. debug information). Te možnosti so lahko, premikanje po vrsticah kode, namesto samo po ukazih, razstava (angl. disassembly) ukazov in drugo. To pa nas pripelje do datotečnega formata razhroščevalnih informacij `DWARF` (Debugging With Arbitrary Record

Formats) [1], ki je široko uporabljen na Unix, Linux in drugih operacijskih sistemih. Delo s temi datotekami/zapisi je občasno lahko mučno, zato že obstajajo knjižnice, ki nam pomagajo pri obdelavi tega formata (npr. `libdwarf` [2]).

DWARF uporablja vnose razhroščevalnik informacij (angl. Debugging Information Entry ozirom `DIE`) za definicijo nizko nivojskie predstavitve izvornega programa. Vsak vpis je sestavljen iz identifikacijske oznake in nizem atributov. Oznaka nam pove, kateremu razredu pripada ta vpis, atributi pa opisujejo latnosti tega vpisa.

Primeri oznak so:

- `DW_TAG_array_type`
- `DW_TAG_label`
- `DW_TAG_class_type`
- `DW_TAG_condition`
- `DW_TAG_const_type`
- `DW_TAG_constant`

Razhroščevalne informacije so predstavljene kot drevo, katerega vozlišča so `DIE` vpisi, tako ima lahko vsak vpis svoje otroke, kar pomeni da je trenutno vozlišče odvisno od scojih otrok. Zapisani so kot sploščeno drevo, in sicer tako da, če vozlišče nima otrok, je naslednji vpis njegov sorojenec, če pa im otroke, je nasledni vpis njegov otrok. Veriga sorojencev se knoča praznim vpisom.

```
// recimo, da ima volišče dva otroka
vozlišče, otrok, otrok, null
```

`DIE` vpisi se ponavadi nahajajo v zaglavju izvedljive ali objektne datoteke, pod razdelkom `.debug_info` in/ali razdelkom `.debug_info.dwo`. Te informacije nam lahko povejo na primer, imena spremenljivk in funkcij, ali kateri ukazi se nanašajo na katero vrstico v izvorni kodi, ali kje v kodi se nahajajo vstopi v funkcije in sše veliko več.

3 Razhroščevalnik na sistemu Windows

Microsoft Windows tudi ponuja razhroščevanli vmesnik, imenovan `debugapi` [7], le ta pa nam podaja nekaj funkcij za uporabo.

- `CheckRemoteDebuggerPresent` programu pove, ali je v teku razhroščevanje nad podanim procesom
- `ContinueDebugEvent` nadaljuje izvajanje procesa
- `DebugActiveProcess` omogoči razhroščevalniku, da se priklopi na aktiven preces
- `DebugActiveProcessStop` ustavi razhroščevanje procesa
- `DebugBreak` ustavi izvajanje procesa

- `IsDebuggerPresent` preveri, ali razhroščevalnik (v uporabniškem načinu) opravlja trenutni proces
- `WaitForDebugEvent` ali `WaitForDebugEventEx` počaka, da se zgodi razhroščevalni dogodek v razhroščevanem procesu
- …

Ostale funkcionalnosti pa ponuja tudi `Win32` API [5]. Med njimi tudi `Memoryapi` [6], ki nam omogoča dostop do pomnilnika procesa.

Delovanje razhroščevalnika je omejeno na operacijski sistem Windows. Implementacija pa se ne razlikuje preveč od implementacije na sistemih Unix, vendar je mogoče malo bolj zakomplicirana.

3.1 PDB

`PDB` [3](Program Database) datotečni format je bil izumljen s strani podjetja Microsoft. Datoteke vsebujejo razhroščevalne informacije, ki jih lahko uporabljajo razhroščevalniki in druga orodja. Microsoft ponuja vmesnike in orodja za delo s temi datotekami, kar pomaga uporabnikom, da ne potrebujejo vedeti celotne zgradbe teh datotek. Vendar je občasno dobro vedeti, kako je ta datoteka sestavljena, zato bomo pogledali nekaj malenkosti o tem formatu.

`PDB` datoteka je oblike MSF (Multi-Stream Format), in MSF je "datotečni sistem znotraj datoteke". Datoteka vsebuje več različnih tokov (angl. stream), ki opisujejo različne informacije, kot so tipi, simboli, izvorne datoteke, …

Datoteka vsebuje:

- `Old Directory` - prejšnji MSF točni imenik (angl. stream directory)
- `PDB tok` - osnovne informacije, …
- `DBI tok` - rezhroščevalne informacije
- …

Eden najpomembnejših tokov je `DBI` tok, ker vsebuje podatke o tem, kako se je program prevedel, objektnih datotekah, ki so bile uporabljene pri povezovanju, izvorne datoteke, in pa tudi reference na druge tokove, ki posedujejo več podrobnosti o zbranih (angl. compiled) datotekah. Na primer `CodeView` simbolni zapisi.

4 Razhroščevalnik za jezik Java / Python ?

Literatura

[1] DWARF Debugging Standard Website. [Online] Dosegljivo: `https://dwarfstd.org/`. Zadnji obisk 14. 1. 2025.

[2] IBM - libdwarf documentation. [Online] Dosegljivo: `https://www.ibm.com/docs/en/zos/3.1.0?topic=utilities-libdwarf`. Zadnji obisk 14. 1. 2025.

[3] LLVM - The PDB File Format. [Online] Dosegljivo: https://llvm.org/docs/PDB. Zadnji obisk 14. 1. 2025.

[4] ptrace(2) - Linux manual page. [Online] Dosegljivo: https://man7.org/linux/man-pages/man2/ptrace.2.html. Zadnji obisk 14. 1. 2025.

[5] Win32 API. [Online] Dosegljivo: https://learn.microsoft.com/en-us/windows/win32/api. Zadnji obisk 14. 1. 2025.

[6] Windows Memora API. [Online] Dosegljivo: https://learn.microsoft.com/en-us/windows/win32/api/memoryapi. Zadnji obisk 14. 1. 2025.

[7] Windows Debugging API. [Online] Dosegljivo: https://learn.microsoft.com/en-us/windows/win32/api/debugapi/1. Zadnji obisk 14. 1. 2025.