Linux a reverzing

Peter Košinár

kosinar[zavináč]eset.sk

Rýchlokurz Linuxu

- Príkazy sú zadávané interpreteru – tzv. shell (bash, tcsh, zsh, ash, ...)
- Zopár ich pozná a vie vykonať priamo, väčšinu ale riešia externé programy – hľadanie podľa premennej prostredia $PATH.
- Pojem „prípony“ de-facto neexistuje.
- Na rozdiel od Windowsu, „aktuálny adresár“ zvyčajne v $PATH nie je:

  `./program` namiesto `program`
- Program štandardne má vstup (stdin, 0), výstup (stdout, 1) a chybový výstup (stderr, 2), všetky tri možno presmerovať z/do súboru:

```
./program < vstup > vystup 2> chybovystup
```

Rýchlokurz Linuxu

- Všetko zaujímavé v systéme je (virtuálny) súbor – nie sú špeciálne API funkcie na zisťovanie voľnej pamäte, bežiacich procesov:

  `/proc/self` = informácie o „mne“

  `/proc/self/maps` = mapa mojej pamäte, ...
- Vytvorenie nového procesu: fork() = mitóza ☺
- Interakcia medzi procesmi – signály (`signal`, `kill`), poprípade zdieľaná pamäť (`shm*`)
- „Debugovacia“ interakcia medzi procesmi je sústredená do funkcie ptrace() – sledovanie vykonávania systémových volaní, vie čítať a meniť pamäť a registre...

Rýchlokurz Linuxu

Užitočné vecičky:

- cat – výpis obsahu súboru

```
cat /proc/self/maps
```

- grep – filtrovanie dát

```
grep rwx /proc/self/maps
```

- `/dev/null = „do bažín“`
- man – dokumentácia k príkazu alebo funkcii (aj C-čkové funkcie)

```
man strlen
```

- “Koniec riadku” je iba “\n”, nie “\r\n”

Linux a spustiteľné súbory

- Interpretované skripty (Bash, Perl, Python, Ruby, PHP, …)
- Interpretované/JITované bytecody (Java, ...)
- Natívne binárne súbory (ELF)

ELF

- Základné koncepty podobné ako v prípade PE súborov:
  - Má hlavičku so základnými informáciami.
  - Pozostáva z blokov, ktorých relatívne rozloženie v súbore zvyčajne nezodpovedá 1:1 umiestneniu vo virtuálnej pamäti.
  - Môže potrebovať funkcie poskytované niekým iným („importy“)
  - Vie mať formu knižnice („DLL“) a funkcie poskytovať („exporty“).
  - Dajú sa k nemu pridať aj ladiace informácie.
- Formát samotný je definovaný pomerne dobre – ale, podobne ako pri PE súboroch, aj jeho používatelia občas až príliš benevolentní.

Prvý pohľad na ELF

Ako zistiť, že ide o ELFa?

- Prvé štyri znaky sú \x7f\x45\x4c\x4f (čítaj: „ELF“).
- Povie to utilitka „file“:

```
zri@zri: $ file program
1-program:  ELF 32-bit LSB executable, Intel 80386,
version 1 (SYSV), dynamically linked (uses shared libs),
not stripped

zri@zri: $ file library
2-library:  ELF 32-bit LSB shared object, Intel 80386,
version 1 (SYSV), dynamically linked, not stripped
```

Ako vyčítať niečo viac?

- Dva základné nástroje: **objdump** a **readelf**:
  - **objdump** je univerzálnejší (rozumie mnohým formátom)
  - **readelf** zase rozumie ELFom lepšie
- Hromada prepínačov, --help sa vie hodiť.
- Pozor na výstup: Oba očakávajú *slušné* správanie a nie zneužívanie hraničných podmienok!

- http://www.skyfree.org/linux/references/ELF_Format.pdf

readelf -h <program>

```
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF32
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           Intel 80386
  Version:                           0x1
  Entry point address:               0x8048320
  Start of program headers:          52 (bytes into file)
  Start of section headers:          6132 (bytes into file)
  Flags:                             0x0
  Size of this header:               52 (bytes)
  Size of program headers:           32 (bytes)
  Number of program headers:         9
  Size of section headers:           40 (bytes)
  Number of section headers:         30
  Section header string table index: 27
```

Sekcie

- Podobne ako v PE súboroch, aj ELFy majú sekcie.
- Sekcie majú svoj „typ“ – PROGBITS, REL, STRTAB, NOBITS, ...
- Konkrétne názvoslovie a sémantika je vecou kompilátora, ale niektoré sú štandardnejšie ako iné: .text, .data, .bss, ...
- Tabuľka názvov sekcií býva tiež v samostatnej sekcii (shstrtab).

readelf -S <program>

```
There are 30 section headers, starting at offset 0x17f4:

Section Headers:
  [Nr] Name              Type            Addr     Off    Size   ES Flg Lk Inf Al
  [ 0]                   NULL            00000000 000000 000000 00      0   0  0
  [ 1] .interp           PROGBITS        08048154 000154 000013 00   A  0   0  1
  [ 2] .note.ABI-tag     NOTE            08048168 000168 000020 00   A  0   0  4
  [ 3] .note.gnu.build-i NOTE            08048188 000188 000024 00   A  0   0  4
  [ 4] .gnu.hash         GNU_HASH        080481ac 0001ac 000020 04   A  5   0  4
  [ 5] .dynsym           DYNSYM          080481cc 0001cc 000050 10   A  6   1  4
  [ 6] .dynstr           STRTAB          0804821c 00021c 00004a 00   A  0   0  1
  [ 7] .gnu.version      VERSYM          08048266 000266 00000a 02   A  5   0  2
  [ 8] .gnu.version_r    VERNEED         08048270 000270 000020 00   A  6   1  4
  [ 9] .rel.dyn          REL             08048290 000290 000008 08   A  5   0  4
  [10] .rel.plt          REL             08048298 000298 000018 08  AI  5  12  4
  [11] .init             PROGBITS        080482b0 0002b0 000023 00  AX  0   0  4
  [12] .plt              PROGBITS        080482e0 0002e0 000040 04  AX  0   0 16
  [13] .text             PROGBITS        08048320 000320 0001a2 00  AX  0   0 16
  [14] .fini             PROGBITS        080484c4 0004c4 000014 00  AX  0   0  4
  [15] .rodata           PROGBITS        080484d8 0004d8 000013 00   A  0   0  4
  [16] .eh_frame_hdr     PROGBITS        080484ec 0004ec 00002c 00   A  0   0  4
  [17] .eh_frame         PROGBITS        08048518 000518 0000bc 00   A  0   0  4
```

readelf -S <program>

```
  [11] .init             PROGBITS        080482b0 0002b0 000023 00  AX  0   0  4
  [12] .plt              PROGBITS        080482e0 0002e0 000040 04  AX  0   0 16
  [13] .text             PROGBITS        08048320 000320 0001a2 00  AX  0   0 16
  [14] .fini             PROGBITS        080484c4 0004c4 000014 00  AX  0   0  4
  [15] .rodata           PROGBITS        080484d8 0004d8 000013 00   A  0   0  4
  [16] .eh_frame_hdr     PROGBITS        080484ec 0004ec 00002c 00   A  0   0  4
  [17] .eh_frame         PROGBITS        08048518 000518 0000bc 00   A  0   0  4
  [18] .init_array       INIT_ARRAY      08049f08 000f08 000004 00  WA  0   0  4
  [19] .fini_array       FINI_ARRAY      08049f0c 000f0c 000004 00  WA  0   0  4
  [20] .jcr              PROGBITS        08049f10 000f10 000004 00  WA  0   0  4
  [21] .dynamic          DYNAMIC         08049f14 000f14 0000e8 08  WA  6   0  4
  [22] .got              PROGBITS        08049ffc 000ffc 000004 04  WA  0   0  4
  [23] .got.plt          PROGBITS        0804a000 001000 000018 04  WA  0   0  4
  [24] .data             PROGBITS        0804a018 001018 000008 00  WA  0   0  4
  [25] .bss              NOBITS          0804a020 001020 000004 00  WA  0   0  1
  [26] .comment          PROGBITS        00000000 001020 000048 01  MS  0   0  1
  [27] .shstrtab         STRTAB          00000000 001068 000106 00      0   0  1
  [28] .symtab           SYMTAB          00000000 001170 000430 10     29  45  4
  [29] .strtab           STRTAB          00000000 0015a0 000253 00      0   0  1
Key to Flags:
  W (write), A (alloc), X (execute), M (merge), S (strings)
  I (info), L (link order), G (group), T (TLS), E (exclude), x (unknown)
  O (extra OS processing required) o (OS specific), p (processor specific)
```

Program headers

- Pri spúšťaní nehrajú sekcie žiadnu rolu – a v súbore nemusia vôbec existovať / obsahovať platné údaje!
- Skutočné informácie o tom, čo a kam sa má nahrať a čo sa má spustiť, sú inde: program headers („segmenty“).
- Tieto hovoria o tom, čo sa má kam do virtuálneho priestoru nahrať a aký prístup má byť k daným stránkam dovolený.
- Neexistuje celkom ekvivalent „imagebase“, ale najčastejšie je v súboroch vídaná začiatočná adresa `0x8048000` (32-bitové).

readelf -l <program>

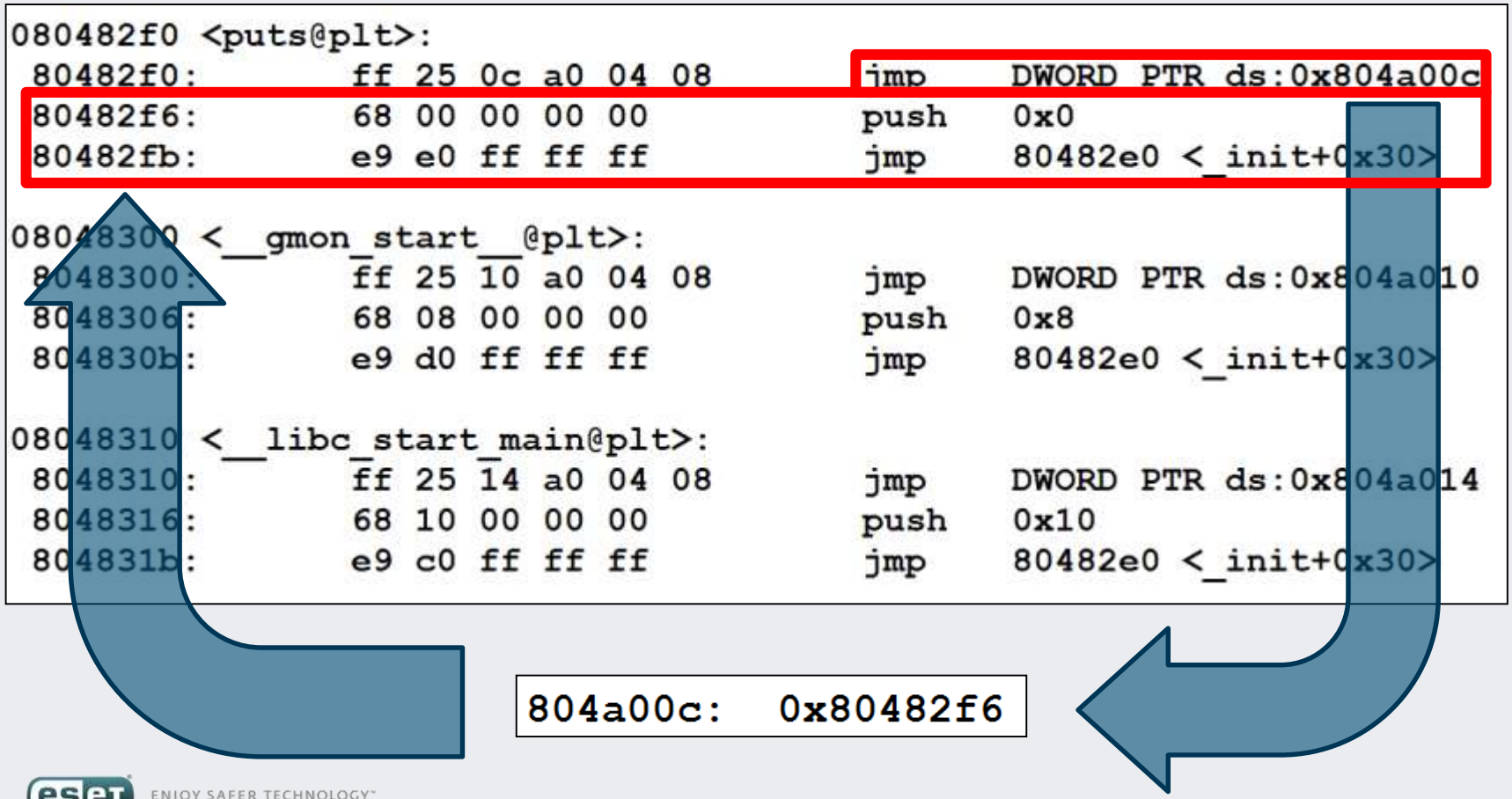
```
Elf file type is EXEC (Executable file)
Entry point 0x8048320
There are 9 program headers, starting at offset 52

Program Headers:
  Type           Offset   VirtAddr   PhysAddr   FileSiz MemSiz  Flg Align
  PHDR           0x000034 0x08048034 0x08048034 0x00120 0x00120 R E 0x4
  INTERP         0x000154 0x08048154 0x08048154 0x00013 0x00013 R   0x1
      [Requesting program interpreter: /lib/ld-linux.so.2]
  LOAD           0x000000 0x08048000 0x08048000 0x005d4 0x005d4 R E 0x1000
  LOAD           0x000f08 0x08049f08 0x08049f08 0x00118 0x0011c RW  0x1000
  DYNAMIC        0x000f14 0x08049f14 0x08049f14 0x000e8 0x000e8 RW  0x4
  NOTE           0x000168 0x08048168 0x08048168 0x00044 0x00044 R   0x4
  GNU_EH_FRAME   0x0004ec 0x080484ec 0x080484ec 0x0002c 0x0002c R   0x4
  GNU_STACK      0x000000 0x00000000 0x00000000 0x00000 0x00000 RW  0x10
  GNU_RELRO      0x000f08 0x08049f08 0x08049f08 0x000f8 0x000f8 R   0x1
```

Symboly

- Importy aj exporty sú špeciálny druh *symbolov* – v tabuľke symbolov.
  - Tabuľky môžu byť dve – jedna „nevyhnutná“ (DSYMTAB) a druhá „informatívna“ (SYMTAB).
  - Informatívne symboly možno odstrániť a nič zlé sa nestane – akurát toho bude menej vidieť.
- Na rozdiel od PE súborov, importy nie sú viazané na konkrétnu knižnicu – t.j. symbol môže dodať ktorákoľvek knižnica.
- Častý lazy binding – symbol sa dohľadá, až keď je prvýkrát potrebný.
- Nie sú to ladiace informácie – tie bývajú ešte oveľa detailnejšie (DWARF).

PLT a GOT



Statické vs. dynamické linkovanie

- Súbor nepoužívajúci žiadne externé knižnice je linkovaný staticky:

```
gcc -static
```

- Výhoda: Program je nezávislý na knižniciach, nemusí riešiť, či je nejaká nainštalovaná a ak áno, v ktorej verzii...
- Nevýhody: Výrazne väčšia veľkosť; všetky knižnicové chyby zostávajú „navždy “ – v prípade objavenia treba program prekompilovať.
- Nevýhoda pre reverzera: Viac kódu na pozeranie, špeciálne v prípade neznámych knižníc, často úplná absencia symbolov.

Position-Independent Code

- Kód, ktorý nie je potrebné relokovať – t.j. môže byť kompletne zdieľaný medzi viacerými procesmi.
- Odkazy na externé symboly sa musia dosadiť „za behu“, ale tie sú súčasťou dát, nie kódu – a sú (očakávateľne) variabilné.
- Vzdialenosť medzi kódom a dátami je fixná – takže sa používa relatívne adresovanie voči aktuálne vykonávanej inštrukcii.

Knižničné vs systémové volania

- Čo sa deje vnútri procesu, zostáva v vnútri procesu.
- Zavolanie funkcie z knižnice robí iba skok / volanie v rámci procesu, samé o sebe nedokáže spraviť nič, čo by nedokázal program spraviť aj sám.
- Akákoľvek interakcia so systémom (prístup k súborom, k iným objektom, k hardvéru, ...) ide cez niektoré systémové volanie.
- Pre väčšinu systémových volaní existujú „štandardné“ knižničné obálky, ktoré dáta vhodne predspracujú, zavolajú systémové volanie a prípadne spracujú chybu.

Syscall (x86)

Systémových volaní existuje pomerne málo:

```
        /usr/include/i386-linux-gnu/asm/unistd_32.h
#define __NR_restart_syscall 0
#define __NR_exit 1
#define __NR_fork 2
#define __NR_read 3
#define __NR_write 4
#define __NR_open 5
#define __NR_close 6
...
#define __NR_seccomp 354
```

Syscall (x86)

- Systémové volanie sa vyvoláva cez `int 0x80`
- Číslo volania je v EAX, parametre v EBX, ECX, EDX, ESI, EDI, EBP:

```
open(“xyz”, O_RDONLY):
        mov eax, 5
        mov ebx, offset “xyz”
        xor ecx, ecx      // O_RDONLY = 0
        int 0x80
```

`/usr/include` = dobrý kandidát na grepovanie!

Reverzerské nástroje

- Statická analýza:
  - IDA ELFom rozumie ☺
  - `objdump -d` vie tiež všeličo dobré povedať; pre Intelistov sa silne odporúča (aj) parameter `-M intel`

- Dynamická analýza:
  - strace – záznam systémových volaní (na báze ptrace)
  - ltrace – knižnicové volania – len pre externé knižnice!
  - LD_PRELOAD trik – podvrhnutie vlastne knižnice s funkciami
  - gdb – GNU debugger a všetky jeho nadstavby / frontendy

Ďakujem za pozornosť

Peter Košinár

kosinar[zavináč]eset.sk