Linux debugging & anti-debugging

aczid

September 8, 2012

(Many thanks to my online friends, you know who you are.)

Now is a good time to download `http://www.hackintherandom2600nldatabox.nl/archive/slides/2012/antidebugging.tgz`

Making Linux ELF binaries

Create easy.c

```
#include <stdio.h>
#include <string.h>
char sekrit[20] = "|OXSuyOIXO^";
int haxxor = 0;
void show_pw(char* foo){
        haxxor = 1;
        memfrob(foo, strlen(foo));
        printf("The password is %s\n", foo);
}
int main(int argc, char** argv){
        printf("Hello, HITR2NLDB!\n");
        if(haxxor){
                show_pw(sekrit);
        } else {
                kill(NULL, 9);
        }
        return 0;
}
```

Compile with:

```
gcc -o easy easy.c
```

Or:

```
CFLAGS="$CFLAGS" make easy
```

ELF layout recap

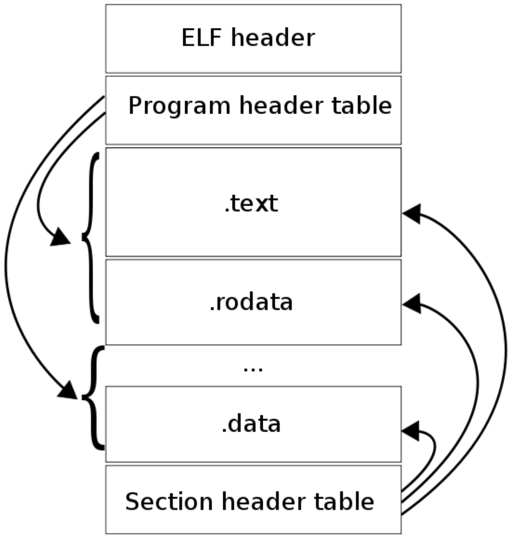


Figure: Elf-layout–en.svg (Wikimedia commons)

strings

```
$ strings -a ./easy
    238 /lib64/ld-linux-x86-64.so.2
    361 __gmon_start__
    370 libc.so.6
    37a memfrob
    382 puts
    387 kill
    38c printf
    393 __libc_start_main
    3a5 GLIBC_2.2.5
    5a4 fff.
    707 l$ L
    70c t$(L
    711 |$0H
    77c The password is %s
    790 Hello, HITR2NLDB!
    80f ;*3$"
   1040 |OXSuyOIXO^
   1054 GCC: (Ubuntu/Linaro 4.6.3-1ubuntu5) 4.6.3
   107f .symtab
   1087 .strtab
   108f .shstrtab
   1099 .interp
   10a1 .note.ABI-tag
```

ldd

```
$ ldd easy
        linux-vdso.so.1 =>  (0x00007fff19f0b000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007fa347192000
        /lib64/ld-linux-x86-64.so.2 (0x00007fa34757c000)
```

`ld` is actually Linux's interpreter for dynamically linked ELF files

```
$ /lib64/ld-linux-x86-64.so.2 ./easy
Hello, HITR2NLDB!
Killed
```

`vdso/gate` is a virtual .so that lives in the kernel and allows for syscalls to be virtualized through `sysenter` instructions rather than traditional interrupts.

readelf

```
$ readelf -a easy
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           Advanced Micro Devices X86-64
  Version:                           0x1
  Entry point address:               0x4004f0
  Start of program headers:          64 (bytes into file)
  Start of section headers:          4416 (bytes into file)
  Flags:                             0x0
  Size of this header:               64 (bytes)
  Size of program headers:           56 (bytes)
  Number of program headers:         9
  Size of section headers:           64 (bytes)
  Number of section headers:         30
  Section header string table index: 27

Section Headers:
```

objdump

```
$ objdump -d easy

easy:   file format elf64-x86-64


Disassembly of section .init:

0000000000400390 <_init>:
  400390:       48 83 ec 08             sub    $0x8,%rsp
  400394:       e8 63 00 00 00          callq  4003fc <call_gmon_start>
  400399:       e8 f2 00 00 00          callq  400490 <frame_dummy>
  40039e:       e8 cd 01 00 00          callq  400570 <__do_global_ctors_aux>
  4003a3:       48 83 c4 08             add    $0x8,%rsp
  4003a7:       c3                      retq

Disassembly of section .plt:

00000000004003b0 <__libc_start_main@plt-0x10>:
  4003b0:       ff 35 3a 0c 20 00       pushq  0x200c3a(%rip)        # 600ff0 <_GLOBAL_OFFSET_TA
  4003b6:       ff 25 3c 0c 20 00       jmpq   *0x200c3c(%rip)        # 600ff8 <_GLOBAL_OFFSET_T
  4003bc:       0f 1f 40 00             nopl   0x0(%rax)
```

ELF cracking

```
$ vim easy
:%!xxd
```

find and nop the branch test

```
:%!xxd -r
:wq
```

strace

```
$ strace ./easy
execve("./easy", ["./easy"], [/* 52 vars */]) = 0
brk(0)                                  = 0x13fb000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f0abe0d0000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=173329, ...}) = 0
mmap(NULL, 173329, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f0abe0a5000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\200\30\2\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=1802936, ...}) = 0
mmap(NULL, 3917016, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f0abdaf3000
mprotect(0x7f0abdca6000, 2093056, PROT_NONE) = 0
mmap(0x7f0abdea5000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b20
mmap(0x7f0abdeab000, 17624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) =
close(3)                                = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f0abe0a4000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f0abe0a3000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f0abe0a2000
arch_prctl(ARCH_SET_FS, 0x7f0abe0a3700) = 0
mprotect(0x7f0abdea5000, 16384, PROT_READ) = 0
mprotect(0x600000, 4096, PROT_READ)     = 0
mprotect(0x7f0abe0d2000, 4096, PROT_READ) = 0
munmap(0x7f0abe0a5000, 173329)          = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 12), ...}) = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f0abe0cf000
write(1, "Hello, HITR2NLDB!\n", 18Hello, HITR2NLDB!
)      = 18
kill(0, SIGKILLKilled
```

ltrace

```
$ ltrace ./easy
__libc_start_main(0x400640, 1, 0x7fff18e9a368, 0x400690, 0x400720 <unfin
puts("Hello, HITR2NLDB!"Hello, HITR2NLDB!
)                                                     = 18
kill(0, 9Killed
```

GDB

```
$ gdb ./easy
gdb$ b main
Breakpoint 1 at 0x400644
gdb$ r
Breakpoint 1, 0x0000000000400644 in main ()
gdb$ set haxxor = 1
gdb$ c
```

Now scriptable with Python and gdb scripts. Check out gdbinit v8.0 by reverse.put.as, and the CERT Linux Triage Tools

procfs

```
$ ls /proc/$PID
attr             cwd       maps        oom_score
autogroup        environ   mem         oom_score_adj
auxv             exe       mountinfo   pagemap
cgroup           fd        mounts      personality
clear_refs       fdinfo    mountstats  root
cmdline          io        net         sched
comm             latency   ns          schedstat
coredump_filter  limits    numa_maps   sessionid
cpuset           loginuid  oom_adj     smaps
```

procfs

```
$ ls -l /proc/self/fd
total 0
lrwx------ 1 aczid aczid 64 Sep  8 17:30 0 -> /dev/pts/9
lrwx------ 1 aczid aczid 64 Sep  8 17:30 1 -> /dev/pts/9
lrwx------ 1 aczid aczid 64 Sep  8 17:30 2 -> /dev/pts/9
lr-x------ 1 aczid aczid 64 Sep  8 17:30 3 -> /proc/8631/fd
$ ls -l /proc/self/exe
lrwxrwxrwx 1 aczid aczid 0 Sep  8 17:30 /proc/self/exe -> /bin/ls
$ cat /proc/self/maps
00400000-0040b000 r-xp 00000000 08:05 167548            /bin/cat
0060a000-0060b000 r--p 0000a000 08:05 167548            /bin/cat
0060b000-0060c000 rw-p 0000b000 08:05 167548            /bin/cat
01d9e000-01dbf000 rw-p 00000000 00:00 0                 [heap]
7fa756187000-7fa756453000 r--p 00000000 08:05 1844412   /usr/lib/locale/locale-archive
7fa756453000-7fa756606000 r-xp 00000000 08:05 398229    /lib/x86_64-linux-gnu/libc-2.15.so
7fa756606000-7fa756805000 ---p 001b3000 08:05 398229    /lib/x86_64-linux-gnu/libc-2.15.so
7fa756805000-7fa756809000 r--p 001b2000 08:05 398229    /lib/x86_64-linux-gnu/libc-2.15.so
7fa756809000-7fa75680b000 rw-p 001b6000 08:05 398229    /lib/x86_64-linux-gnu/libc-2.15.so
7fa75680b000-7fa756810000 rw-p 00000000 00:00 0
7fa756810000-7fa756832000 r-xp 00000000 08:05 399715    /lib/x86_64-linux-gnu/ld-2.15.so
7fa756a02000-7fa756a05000 rw-p 00000000 00:00 0
7fa756a30000-7fa756a32000 rw-p 00000000 00:00 0
7fa756a32000-7fa756a33000 r--p 00022000 08:05 399715    /lib/x86_64-linux-gnu/ld-2.15.so
7fa756a33000-7fa756a35000 rw-p 00023000 08:05 399715    /lib/x86_64-linux-gnu/ld-2.15.so
7fffc5643000-7fffc5665000 rw-p 00000000 00:00 0         [stack]
7fffc5722000-7fffc5723000 r-xp 00000000 00:00 0         [vdso]
ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0 [vsyscall]
```

GCC anti-debugging “features”

-g	Produce debugging information in the operating system’s native format (stabs, COFF, XCOFF, or DWARF 2). GDB can work with this debugging information.
-static	On systems that support dynamic linking, this prevents linking with the shared libraries. On other systems, this option has no effect.
-fomit-frame-pointer	Don’t keep the frame pointer in a register for functions that don’t need one. This avoids the instructions to save, set up and restore frame pointers; it also makes an extra register available in many functions. It also makes debugging impossible on some machines.
-funroll-loops	Unroll loops whose number of iterations can be determined at compile time or upon entry to the loop. -funroll-loops implies -frerun-cse-after-loop, -fweb and -frename-registers. It also turns on complete loop peeling (i.e. complete removal of loops with small constant number of iterations). This option makes code larger, and may or may not make it run faster.

```
$ man gcc
/debugging impossible
```

strip & sstrip

```
$ gcc -o easy easy.c
$ strings -a easy | wc -l
95
$ readelf -h easy | grep 'Number of'
  Number of program headers:         9
  Number of section headers:         31
$ strip easy
$ strings -a easy | wc -l
47
$ readelf -h easy | grep 'Number of'
  Number of program headers:         9
  Number of section headers:         29
$ sstrip easy
$ strings -a easy | wc -l
19
$ readelf -h easy | grep 'Number of'
  Number of program headers:         9
  Number of section headers:         0
```

ELF obfuscation

```
$ ./troll easy
$ readelf -a easy
readelf: Error: Section headers are not available!
ELF Header:
  Magic:   7f 45 4c 46 53 41 52 45 55 42 45 52 31 33 33 37
  Class:                             <unknown: 53>
  Data:                              <unknown: 41>
  Version:                           82 <unknown: %lx>
  OS/ABI:                            <unknown: 45>
  ABI Version:                       85
  Type:                              EXEC (Executable file)
  Machine:                           Advanced Micro Devices X86-64
  Version:                           0x7a69
  Entry point address:               0x4004f0
  Start of program headers:          0 (bytes into file)
  Start of section headers:          64 (bytes into file)
  Flags:                             0x0
  Size of this header:               4480 (bytes)
  Size of program headers:           0 (bytes)
  Number of program headers:         0
  Size of section headers:           0 (bytes)
  Number of section headers:         31337
  Section header string table index: 0

There are no program headers in this file.
```

Packers/crypters

- Parse ELF segments to make a memory map and compress / encrypt it
- Decode and correctly load the memory map from a 'stub' binary
- Jump to the relative entrypoint

Simple inline crypter stub

```
extern void* __libc_csu_init;
static void dummy();
#define round_down(to_round,page_size) ((((unsigned long) to_round)/page_size)*page_size)
#define round_up(to_round,page_size) (round_down(to_round, page_size) + page_size)
static void __attribute__((constructor)) stub(void){
        void* start = &dummy;
        void* end = &__libc_csu_init;
        size_t code_size = (((unsigned long) end)) - ((unsigned long) start);
        size_t page_size = getpagesize();
        mprotect((void*) round_down(start, page_size), round_up(code_size, page_size),
                  PROT_READ | PROT_WRITE | PROT_EXEC);
        memfrob(&dummy, code_size);
        mprotect((void*) round_down(start, page_size), round_up(code_size, page_size),
                  PROT_READ | PROT_EXEC);
}
static void dummy(){}
```

Code to be decrypted after modifying file

```
0000000000400ac5 <main>:
  400ac5:       7f 62                   jg     400b29 <__libc_csu_init+0x9>
  400ac7:       a3 cf 62 a9 c6 0a a3    movabs %eax,0xd657a30ac6a962cf
  400ace:       57 d6
  400ad0:       62                      (bad)
  400ad1:       a3 5f da 62 a3 7f c2    movabs %eax,0x94c2c27fa362da5f
  400ad8:       c2 94
  400ada:       d5                      (bad)
  400adb:       d5                      (bad)
  400adc:       d5                      (bad)
  400add:       af                      scas   %es:(%rdi),%eax
  400ade:       ea                      (bad)
  400adf:       5f                      pop    %rdi
  400ae0:       33 c2                   xor    %edx,%eax
  400ae2:       25 d4 d5 d5 af          and    $0xafd5d5d4,%eax
  400ae7:       ea                      (bad)
  400ae8:       5f                      pop    %rdi
  400ae9:       3a 62 a1                cmp    -0x5f(%rdx),%ah
  400aec:       6f                      outsl  %ds:(%rsi),(%dx)
  400aed:       c2 62 a3                retq   $0xa362
  400af0:       ed                      in     (%dx),%eax
  400af1:       c2 68 d5                retq   $0xd568
  400af4:       d5                      (bad)
  400af5:       d5                      (bad)
  400af6:       af                      scas   %es:(%rdi),%eax
  400af7:       ea                      (bad)
  400af8:       5e                      pop    %rsi
  400af9:       3b 95 26 26 6a 2a       cmp    0x2a6a2626(%rbp),%edx
  400aff:       c2 56 d1                retq   $0xd156
  400b02:       d5                      (bad)
  400b03:       d5                      (bad)
```

Disable coredumps

```
prctl(PR_SET_DUMPABLE, 0);
```

Avoid ptrace

Needs a bit extra for modern `ptrace_scope`.

```
int detect_ptrace(void) {
        pid_t pid;
        pid_t ppid;
        int status;
        int success = 1;
        int res = 0;
        pid = fork();
        if(pid == 0){
                /* wait for signal from parent */
                wait(NULL);
                ppid = getppid();
                res = ptrace(PTRACE_ATTACH, ppid, 0, 0);
                if(res == 0){
                        /* wait for parent to be ready for ptrace */
                        waitpid(ppid, NULL, 0);
                        ptrace(PTRACE_CONT, ppid, 0, 0);
                        ptrace(PTRACE_DETACH, ppid, 0, 0);
                }
                exit(res);
        } else if(pid > 0) {
                /* prepare to be ptraced */
                prctl(PR_SET_PTRACER, pid);
                prctl(PR_SET_DUMPABLE, 1);
                /* instruct child to continue */
                kill(pid, SIGCONT);
                /* wait for child to exit */
                waitpid(pid, &status, 0);
                prctl(PR_SET_DUMPABLE, 0);
                success = WEXITSTATUS(status);
        }
        return success;
}
```

LD_{PRELOAD,LIBRARY_PATH}

```
long ptrace(int req, int pid, void *adr, void *dat){
        return 0;
}

$ gcc -shared -fPIC -o fakeptrace.so fakeptrace.c
$ LD_PRELOAD=./fakeptrace.so ./anti_ptrace
```

Avoid LD overrides

```
int detect_ld_env(char **envp){
        char **p;
        for (p = envp; *p != NULL; p++) {
                if ((*p)[0] == 'L' && (*p)[1] == 'D' && (*p)[2] == '_')
                        return 1;
        }
        return 0;
}
```

We have to go deeper

```
#include <stdlib.h>
void __attribute__((constructor)) cleanup(void){
        unsetenv("LD_PRELOAD");
        unsetenv("LD_LIBRARY_PATH");
}

long ptrace(int req, int pid, void *adr, void *dat){
        return 0;
}
```

Hide and seek - beating procfs

Regularly do something like:

```
pid_t pid;
kill(0, SIGSEGV);
pid = fork();
prctl(PR_SET_NAME, random_string());
if(pid > 0){
        exit(rand());
} else {
        setsid();
}
```

```
$ ./nothere
Try to find me!
$ kill $!
bash: kill: (6241) - No such process
$ killall nothere
nothere: no process found
$ ps aux | grep nothere
aczid    30703  0.0  0.0   9392   940 pts/11   S+   20:02   0:00 grep --color=auto nothere
$ You can't find me, huh? I was hiding in pid 6342 with processname 2cePjafX :)
```

Profit

```
$ file serious
serious: ELF, unknown class 83
$ objdump -d serious
objdump: serious: File format not recognized
$ size serious
size: serious: File format not recognized
$ ldd serious
not a dynamic executable
$ strip serious
strip:serious: File format not recognized
$ sstrip serious
sstrip: serious: not a valid ELF file
$ strace -f ./serious
< ... >
write(1, "Nope!\n", 6Nope!
$ ltrace -f ./serious
ltrace: Can’t open ELF file "./serious"
$ gdb ./serious
"/path/to/serious": not in executable format: File format not recognized
$ LD_FOO=bar ./serious
Nope!
$ ./serious
Okay.
```

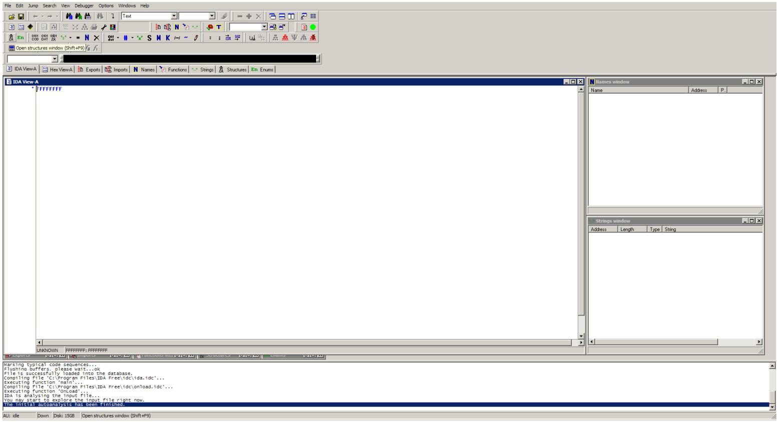
Profit

```
$ unstrip -f hexedited_elf
[../src/Object-elf.C][5130]WARNING: .shstrtab section not found in ELF b
[../src/Object-elf.C][362]WARNING: .shstrtab section not found in ELF bi
[../src/Object-elf.C][442]WARNING: .shstrtab section not found in ELF bi
Segmentation fault (core dumped)

$ readelf -S readelf_crash
There are 30 section headers, starting at offset 0x1140:

Section Headers:
  [Nr] Name              Type             Address           Offset
       Size              EntSize          Flags  Link  Info  Align
  [ 0] <no-name>         NULL             0000000000000000  00000000
Segmentation fault (core dumped)
```

Profit



```
FFFFFFFF
```

```
UNKNOWN    FFFFFFFF: FFFFFFFF
```

```
Marking typical code sequences...
Flushing buffers, please wait...ok
File is successfully loaded into the database.
Compiling file 'C:\Program Files\IDA Free\idc\ida.idc'...
Executing function 'main'...
Compiling file 'C:\Program Files\IDA Free\idc\onload.idc'...
Executing function 'OnLoad'...
IDA is analysing the input file...
You may start to explore the input file right now.
The initial autoanalysis has been finished.
```

```
AU: idle    Down    Disk: 15GB    Open structures window (Shift+F9)
```

Systemtap

Get a debug kernel running and install systemtap

```
global syscalls
probe syscall.* {
  syscalls[pid()]++
}
probe end {
  printf("%-10s %-s\n", "#SysCalls", "PID")
  foreach (pid in syscalls-)
    printf("%-10d %-d\n", syscalls[pid], pid)
}

$ stap syscalls.stp
^C
#SysCalls PID
      100 1337
```

Ideas / TODO

- A better packer
- Fake disassembly
- Using code caves
- Nanomites
- Virtualizing
- Inspect the ld.so at run-time for overrides

EOF

Thanks, questions, food?

`http://www.hackintherandom2600nldatabox.nl/archive/slides/2012/antidebugging.tgz`

`HTTP error 451: Support VXHeavens`

`http://vx.netlux.org/`