Reverse Engineering Linux x86 Binaries

Outline

- What is Reverse Engineering?
- Technical Background
- Reverse Engineering Techniques
- Case Study
- Documentation
- Legalities

What is Reverse Engineering?

"Oh, you can't get out backwards. You've got to go forwards to go back, better press on." - Willy Wonka

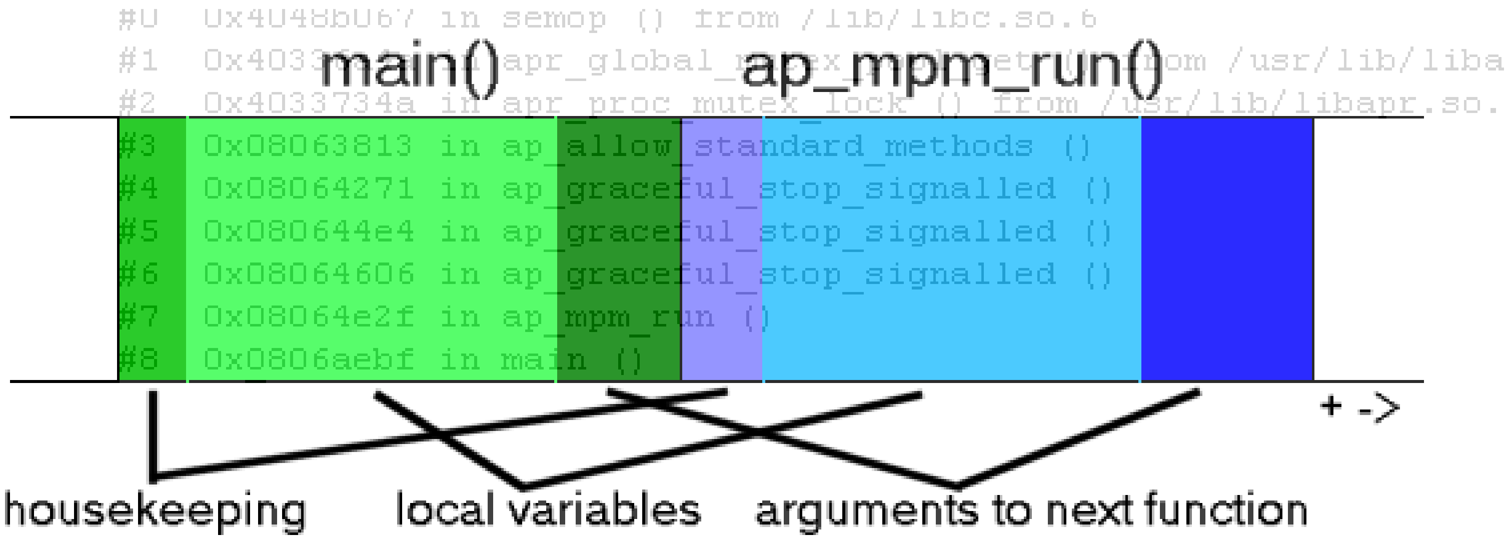
- The process of examining and probing a program to determining the original design
- The documentation from this process can be used to
  - Document the characteristics of an unknown program
  - Clarify published interfaces for interacting with the program
  - Implement another program to interact with a proprietry program
  - Modify the features or behaviour of a program

Technical Background

- Network Protocols/File Formats
- Function Calls
- Libraries
- Kernel Functions
- Execution Tracing
- The ELF File Format
- The /proc Filesystem

Function Calls

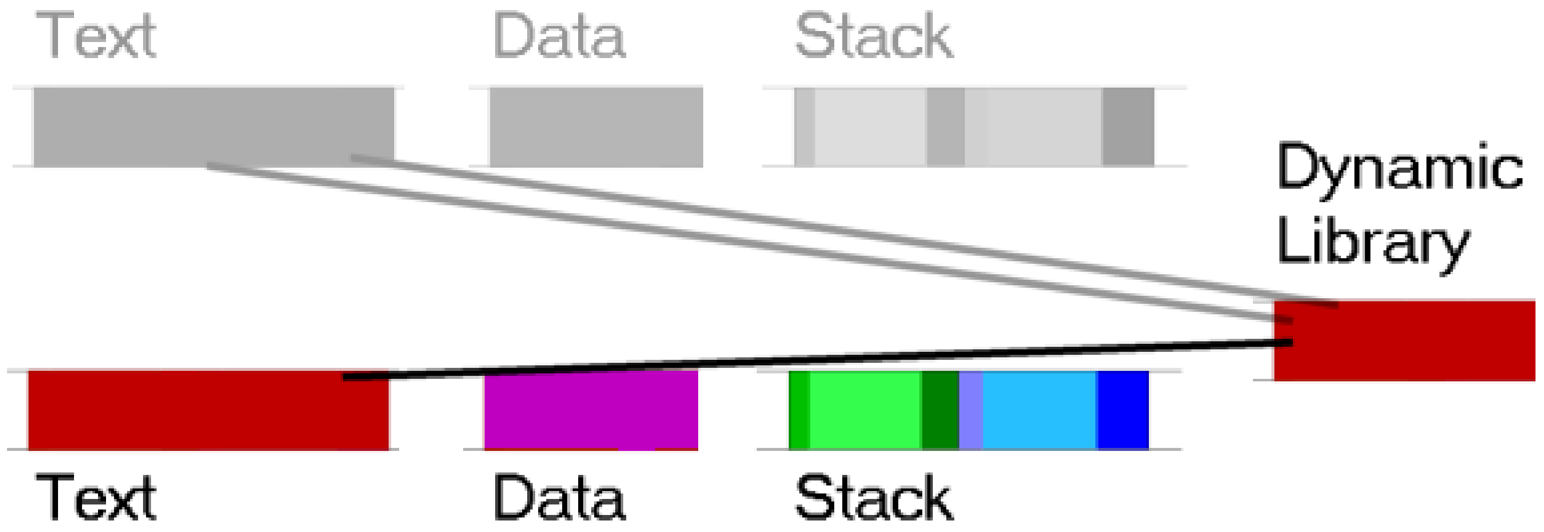
- When a program is written, it is logically divided into functions, which call each other to perform tasks.
- Every process has a scratch area referred to as the stack, on which values are stored in a first in last out fashion.



```
#0  0x4048b067 in semop () from /lib/libc.so.6
#1  0x4033[illegible] apr_global_[illegible] () from /usr/lib/liba
#2  0x4033734a in apr_proc_mutex_lock () from /usr/lib/libapr.so.
#3  0x08063813 in ap_allow_standard_methods ()
#4  0x08064271 in ap_graceful_stop_signalled ()
#5  0x080644e4 in ap_graceful_stop_signalled ()
#6  0x08064606 in ap_graceful_stop_signalled ()
#7  0x08064e2f in ap_mpm_run ()
#8  0x0806aebf in main ()
```

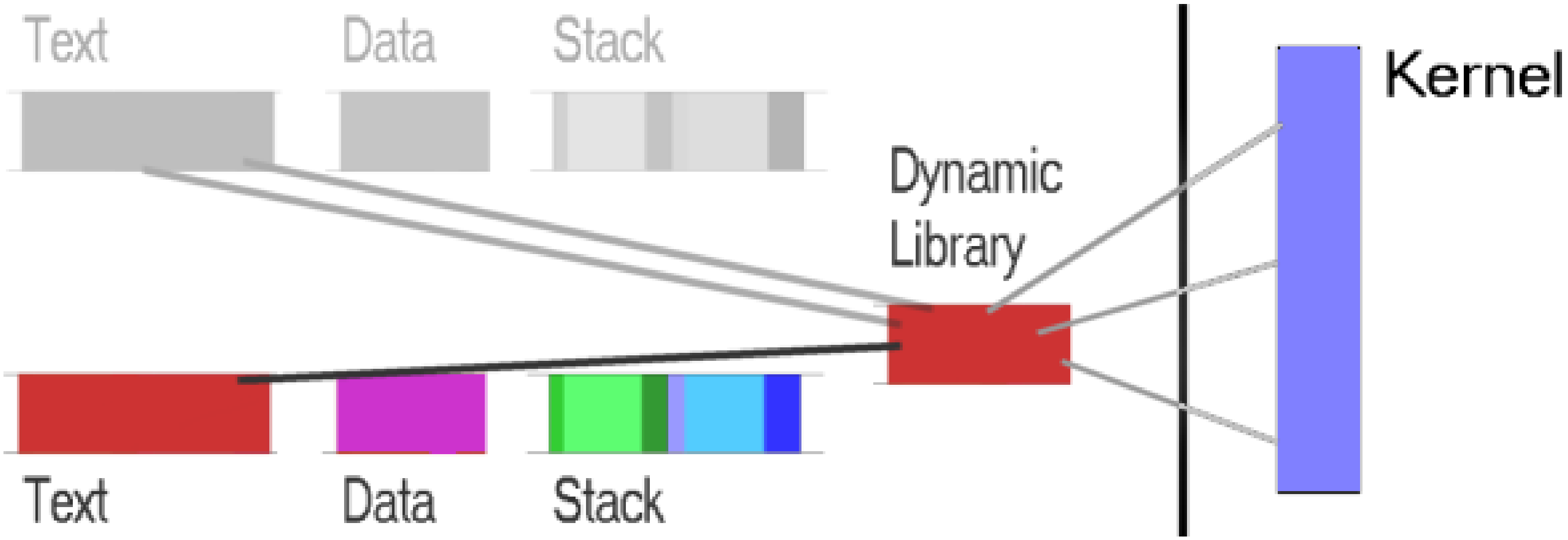
Libraries

- Libraries contain functions that can be used by many programs, eg printf()
  - static executables
  - dynamic executables
  - runtime linking



Kernel Functions

- Some library functions need the kernels help, for example to perform IO.



Execution Tracing

- Execute program in a debugger, monitor or alter live data values or program behaviour
- xtrace (functions), ltrace (library functions) and strace (system calls) use the kernels ptrace functionality to intercept and display calls.

```
# strace -iq /usr/sbin/dhcpd
...
[400f5508] open("/var/lib/dhcp/dhcpd.leases", O_RDONLY) = 4
[400f57c8] lseek(4, 0, SEEK_END)              = 467
[400f57c8] lseek(4, 0, SEEK_SET)              = 0
[400f56c8] read(4, "# All times in this file are in "..., 467)
                                                      = 467
[400f5641] close(4)                           = 0
[400f5508] open("/var/lib/dhcp/dhcpd.leases",
                O_WRONLY|O_APPEND|O_CREAT, 0666)
                = -1 EACCES (Permission denied)
[400b6fbd] time([1070352265])                 = 1070352265
[40103666] send(3, "Dec  2 18:34:25 dhcpd: "..., 76, 0) = 76
```

The ELF File Format

- A format for executable object files (eg. executables, relocatable files, shared object files).
- Divides the file into headers and sections.
- Interesting sections include:

```
.text:                     The programs executable instructions
.symtab:                   The symbol table (imported and exported)
.rel*:                     Relocation entries (where symbols used)
.data and .data1:          Initialised data
.rodata and .rodata1:      Initialised read only data
.debug:                    Symbolic debugging information
```

The /proc Filesystem

- The proc filesystem is a representation of kernel structures
- It contains much useful information about running processes, including:
  - cmdline: command line process was invoked with
  - maps: memory map of process and libraries
  - status: process state, privileges, granular memory usage, signal handling and capabilities
  - fd: file descriptors in use by process
  - cwd: link to processes current directory
  - root: link to processes root directory
  - mounts: mount table visible to process

A world of tools

- Low Detail, Easy to Learn
  - strings
  - ldd
  - ptools, /proc
  - strace, ltrace, xtrace
  - ngrep
- Medium Detail, Some Knowledge Required
  - Fenris, Ragnarok
  - REC
  - Ethereal
- High Detail, Most Knowledge Required
  - objdump
  - GDB
  - TCPDump

Reverse Engineering Techniques

- Mix and Match to suit your objective:
  - Deadlisting and Disassembly
  - Live Debugging and Tracing
  - Enumeration
  - Library Replacement/Interception

Examining an ELF binary with binutils

- ldd, nm and objdump can dump and disassemble interesting sections

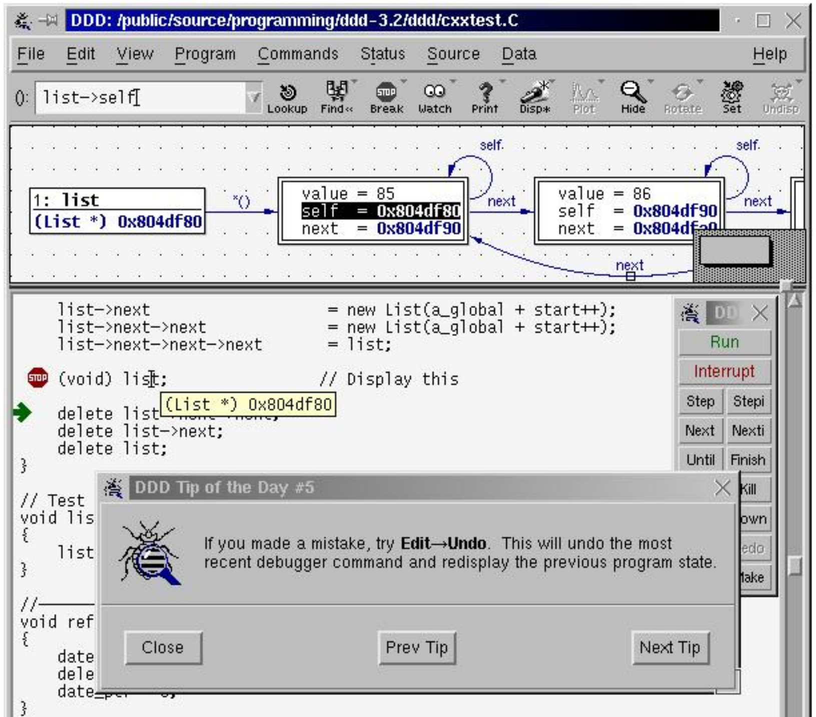
```
strings:        strings /usr/bin/who
dependencies:   ldd /usr/bin/yes
symbols:        nm -D -l -S /usr/bin/yes
sections:       objdump -h /usr/bin/who
data:           objdump -s -j .rodata /usr/bin/who
code:           objdump -d -r -j .text /usr/bin/who
```

Examining an ELF binary with REC

- REC - the Reverse Engineers Compiler
- Recognises "signature" assembly code that compilers generate for program structures such as conditionals (if, else, switch) and loops (for, do, while).
- Generates C like code.
- Works with Linux and MS Windows executables.

Live Debugging with GDB

- GNU Debugger
- Supports many CPU architectures, several high level languages, threading.



```
1: list
(List *) 0x804df80

value = 85
self  = 0x804df80
next  = 0x804df90

value = 86
self  = 0x804df90
next  = 0x804dfa0
```

```
    list->next                  = new List(a_global + start++);
    list->next->next            = new List(a_global + start++);
    list->next->next->next      = list;

    (void) list;               // Display this
(List *) 0x804df80
    delete list
    delete list->next;
    delete list;
}

// Test
void lis
{
    list
}

//
void ref
{
    date
    dele
    date
}
```

DDD Tip of the Day #5

If you made a mistake, try **Edit→Undo**. This will undo the most recent debugger command and redisplay the previous program state.

Live Debugging with strace

- strace, ltrace and xtrace can attach to processes
- these tools show calls as they happen

```
# ltrace -e fwrite_unlocked ls
fwrite_unlocked("init.d", 1, 6, 0x401585c0)      = 6
fwrite_unlocked("rc0.d", 1, 5, 0x401585c0)       = 5
fwrite_unlocked("rc2.d", 1, 5, 0x401585c0)       = 5
fwrite_unlocked("rc4.d", 1, 5, 0x401585c0)       = 5
fwrite_unlocked("rc6.d", 1, 5, 0x401585c0)       = 5
fwrite_unlocked("rc.sysinit", 1, 10, 0x401585c0) = 10
init.d  rc0.d  rc2.d  rc4.d  rc6.d     rc.sysinit
fwrite_unlocked("rc", 1, 2, 0x401585c0)          = 2
fwrite_unlocked("rc1.d", 1, 5, 0x401585c0)       = 5
fwrite_unlocked("rc3.d", 1, 5, 0x401585c0)       = 5
fwrite_unlocked("rc5.d", 1, 5, 0x401585c0)       = 5
fwrite_unlocked("rc.local", 1, 8, 0x401585c0)    = 8
rc      rc1.d  rc3.d  rc5.d  rc.local
```

Live Debugging with Fenris

- Fenris is a set of tools; Fenris, Aegir, Ragnarok, Dress
- A bit like having strace, ltrace, gdb, objdump and REC rolled into one
- Fenris: backend, tracer
- Aegir: client, interactive debugger
- Ragnarok: trace to HTML
- Dress: rebuilds stripped symbol table

```
[0804b158] 02   local fnct_11 (g/805ace7 "fenris.h")
[0804b158] 02   + 805ace7 = 805ace7:9  (first seen in F fnct_13:fnct_14)
[4009ee3e] 03    L strlen (805ace7 "fenris.h") = 8
[4009ee3e] 03    + 805ace7 = 805ace7:9  (first seen in F fnct_13:fnct_14)
[0804eb74] 03    local fnct_6 (9) (Click here for trace of this function)
[0804eaf3] 03    ...return from function = ""
[4009e8cf] 03    L strcpy (805bc58, 805ace7 "fenris.h") = 805bc58
[4009e8cf] 03    + 805ace7 = 805ace7:9  (first seen in F fnct_13:fnct_14)
[4009e8cf] 03    + 805bc58 = 805bc58:9  (first seen in L fnct_14:malloc)
[4009e8cf] 03    \ buffer 805bc58 modified.
[4009e8cf] 03    \ data migration: 805ace7 to 805bc58
[0804eb84] 02   ...return from function = ""
```

Network Enumeration

- Given reasonable knowledge of the protocol, alter one value over its range and observe results.
  - May trigger informative error messages.
- Libnet/LibPCap can be used to quickly build custom packets.

Enumeration with Plugins

- A variation on "poke and fsck"
- Stick values into the API and see what happens.
- eg. Determining data available to Netscape Directory Server plugins:
  - plugins are shared libraries
  - parameters passed in "parameter block"
  - slapi_pblock_get(pb, SLAPI_*, &value);
  - SLAPI_* is actually a number
    - Calling for SLAPI_* 1 to 300 reveals all parameters
    - Interpreting the return values is another challenge
    - Some values actually crash slapd
    - Some values cause warnings in logs

Library Replacement/Interception

- LD_PRELOAD instructs the linker to preload shared libraries.
- This can be used to replace standard library functions.
  - eg. replace time(2) to return a different date to test for Y3k bugs.
  - or replace socket(), connect(), read(), write() for network or file interception

Case Study: an MPEG library

- Via produce a great little (22cm x 19cm) motherboard.
- They only produce binary drivers for the on board MPEG decoder, limitting its use to certain kernels and Linux distributions.
- Drivers include a shared library and kernel modules.
- "GPL plus additional rights".
- Ivor Hewitt reverse engineered the MPEG library.
- Reversing of the rest of the driver is apparently underway.

Methodology

- By examining a disassembly, write similar C code.
- Examine function calls to determine arguments.
- Analyse function call tree to determine structure.

Tools

- objdump
- header files
- IDA-Pro

Results

- Reproduced source code for library.
- Source code reveals how to talk to kernel module.
- Some patches have been made to Video 4 Linux and XFree86 to support the MPEG hardware as a result of this project.
- Definately not and example of clean room reverse engineering.

Documentation

- Make useful insight available
- Provide a history of your discoveries
- Document progress to prevent repetition

The Call Tree

- Shows what functions are called where, as a tree.
- Useful for getting a feel for the program flow.
- Can identify the purpose of a function.
- Fenris' Ragnarok can generate a similar tree from an execution trace.

```
.- main
| .- setlocale
| | brk
| | brk
| | brk
| | open
| | fstat64
| | mmap
| | read
| | brk
| | read
| | close
| | munmap
```

Writing API Documentation

- For each callable function, document the purpose and parameters.
- Almost a prerequisite for interacting with an API.
- See Unix's section 2 and 3 man pages for excellent examples of API documentation.
    - Name
    - Synopsis
    - Description
    - Parameters
    - Return Value
    - See Also

Writing Pseudocode

- For each interesting function, or even the entire program, write a pseudocode representation of the alogrithm.
- Pseudocode seperates the code from the concepts and ideas.
  - eg. bubblesort:

```
Assume the list is not sorted (sorted is set to false)
while(!sorted)
{
   Assume the list is sorted (sorted is set to true)
   Set i to loop through list from first to last - 1
   {
      if list[i] and list[i + 1] are not in the right order
      {
         switch them (called a swap)
         set sorted to false
      }
   }
}
```

example from http://www.rrcc-online.com/~julies/csc160/bubble.htm

Network Protocols

- RFCs contain many examples of how to document network protocols.
- Eg, from RFC 1885: ICMPv6:

```
The ICMPv6 messages have the following general format:

       0                   1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |     Type      |     Code      |          Checksum             |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                                                               |
      +                         Message Body                          +
      |                                                               |

   The type field indicates the type of the message. Its value
   determines the format of the remaining data.

   The code field depends on the message type. It is used to create an
   additional level of message granularity.

   The checksum field is used to detect data corruption in the ICMPv6
   message and parts of the IPv6 header.
```

File Formats

- Hex editors are great if you can spot a pattern.
- If not, you can use strace to examine file access patterns, or Fenris' aegir to analyse memory buffer accesses
- Failing that, you may have to disassemble the file access routines.
- Once you have the file format, documentation should be similar to that for network protocols

Legalities

- If in doubt, seek professional legal advice
  - Copyright Amendment (Digital Agenda) Act 2000
  - Trade Secrets
  - Software Patents

Questions?

Further References

- Introduction to Reverse Engineering Software
  - http://www.acm.uiuc.edu/sigmil/RevEng/
- Executable and Linking Format (ELF)
  - http://www.skyfree.org/linux/references/ELF_Format.pdf
- Andrew Tridgell's Network Analysis Techniques presentation
  - http://us1.samba.org/samba/ftp/slides/net_analysis.pdf
- REC: The Reverse Engineers Compiler
  - http://www.backerstreet.com/rec/rec.htm
- Fenris: debugger, tracer, decompiler
  - http://razor.bindview.com/tools/fenris/
- Ivor's Via Mini-ITX MPEG library page
  - http://www.ivor.it/cle266/

Further References - 2

- Legalities
  - Electronic Frontiers Australia
    - http://www.efa.org.au
  - Australian Digital Alliance
    - http://www.digital.org.au
  - ipcr 2000: Review of intellectual propery legislation under the Competition Principles Agreement
    - Intellectual Property and Competition Review Committee
    - http://www.ipcr.gov.au/
  - Chilling Effects Clearinghouse
    - http://www.ipcr.gov.au/