radare

LaCon2008

--pancake

radare

Block based command line hexadecimal editor
 - Multiple IO backends
 - Debugger support
 - Configurable hashtable ('-e' flag or 'e' cmd)
 - All commands are single letter (? for help)
 - Flexible command syntax
   > 3pd 20 @@ sym_* > file
- IO plugins also hooks io_system()
   > !contsc write

Screen filtering
 - Output in ascii, ansi, w32 console, html
 - OTF string replacements

Networking

Remoting
- All IO can be wrapped and URIs can be nested to use radare remotely. Non-standard IO cmds is based on string parsing.

```
$ radare connect://10.0.3.22:9898/dbg:///bin/ls
```

IO backend for socket connections
- Handles a socket as a growing file

Hexadecimal editor

Multiple IO backends as plugins
- posix, ptrace, tcp, haret, w32, ewf, wine, ..

Block based editor
- Command line and visual interface
- Zoom out/in for global views
- p% bar showing info of functions, data, code..

Print modes
- Different radix bases, timestamps, endian, C structs/code, assembly

Undo history
- For seeks and writes

Search engine

Advanced search engine
- Strings (char, wchar), bytes
- Multiple keyword definition
- Binary masks for each
- Ranged searchs

Pattern searching
- Look for byte patterns from a pattern length

Grepping for opcodes
- pd 0xfff | grep call eax

Expanded AES key search
- Victor muñoz algorithm used for the Wii

Disassembler

Multiple architectures (asm.arch)
- x86 (16,32,64), arm, mips, sparc, powerpc, m68k, java, msil, csr ..

Syntax flavours (asm.syntax)
- intel, at&t, olly, pseudocode

Basic flow analysis
- ascii-art jump lines

Metadata
- Comments, data types, execution traces, symbols, flags, easily scriptable

Assembler

$ rasm

Multiple arch cmdline assembler/disassembler
 - Allow to define the base address
 - Multiple syntaxis support
 - rsc backend (using NASM or GAS)
 - Pseudo-opcodes for fast patching
 - Raw assembler from files

```
$ rasm 'mov eax, 33'
b8 21 00 00 00

$ rasm -d 'b8 21 00 00 00'
mov eax,33
```

Code analysis

Function analysis
- Identify function sizes, local variables, stack size, data references.

Basic blocks
- Uses graph.jmpblocks, .callblocks, ...
- GUI for graphs

Opcode
- Jump information, and basic data access
- Initial work on code emulation (pseudocode)

Data analysis
- Find string, registers, function pointers

Binary diff

$ radiff /bin/true /bin/false

Raw file byte-level diffing
 - byte-per-byte memory comparision
 - Support for delta diff (erg0t, gnu diff)

Code graph differences
 - From imternal graph analysis
 - Import data from IDA
 - Identifies new paths, blocks and local vars

Checksumming

$ rahash -s “hello” -a md5

Multiple hash algorithms
 - crc16,32, md4-5, sha1-512, xor parity, mod

Entropy calculation
 - Entropy, energy, hamming distance

Block based checksumming
 - Partial hash for big disk images. (f.ex)
 - Configurable block size
 - Define range of bytes (from, to, length)

Rabin

Identify file types
- Support for ELF, PE, CLASS, MACH-O,..

Extract information
- Architecture (intel, arm, ..)
- Imports/exports
- Sections
- Linked libraries
- Strings in .data section

Debugger

Ported to multiple OS/arches:
- GNU/Linux – x86-32,64, mips, arm
- Net/Free/OpenBSD – x86-32,64
- MacOSX – x86-32*, powerpc*
- Solaris – x86-32*, sparc*

Other backends:
- GDB, GDB remote, WineDBG, GxEmul

Extreme development
- Needs some refactoring
- Raw and handy interface

*** = work in progress**

Debugger (2)

CPU control
- Get/set drx,gp,fp,mm registers and flags

Breakpoints
- Software/Hardware support
- Watchpoint expressions

Memory control
- Alloc/free/mprotect/mmap
- Dump/restore memory pages

Signal handling
- Edit event and signal handlers

File descriptors
- Open, dup, close, seek, socket-connect

Debugger (3)

Stepping
- step, step over, stepbp (mips)
- skip N opcodes (!jmp eip+x)

Continuations
- continue until address, fork or event
- !contsc: syscall tracing

Threads and processes
- Send events, attach/detach, status

Touch tracing
- Swap memory filled with traps
- Trace information available for processing

Shellcodes

$ rasc -N 30 -i x86.linux.binsh -c > sc.c

Small database of common shellcodes
 - Multiple output forms
 - Pad generators (A, nops, traps, 1234..)

Support for syscall proxying
 - Also radare with an IO plugin

Scripting

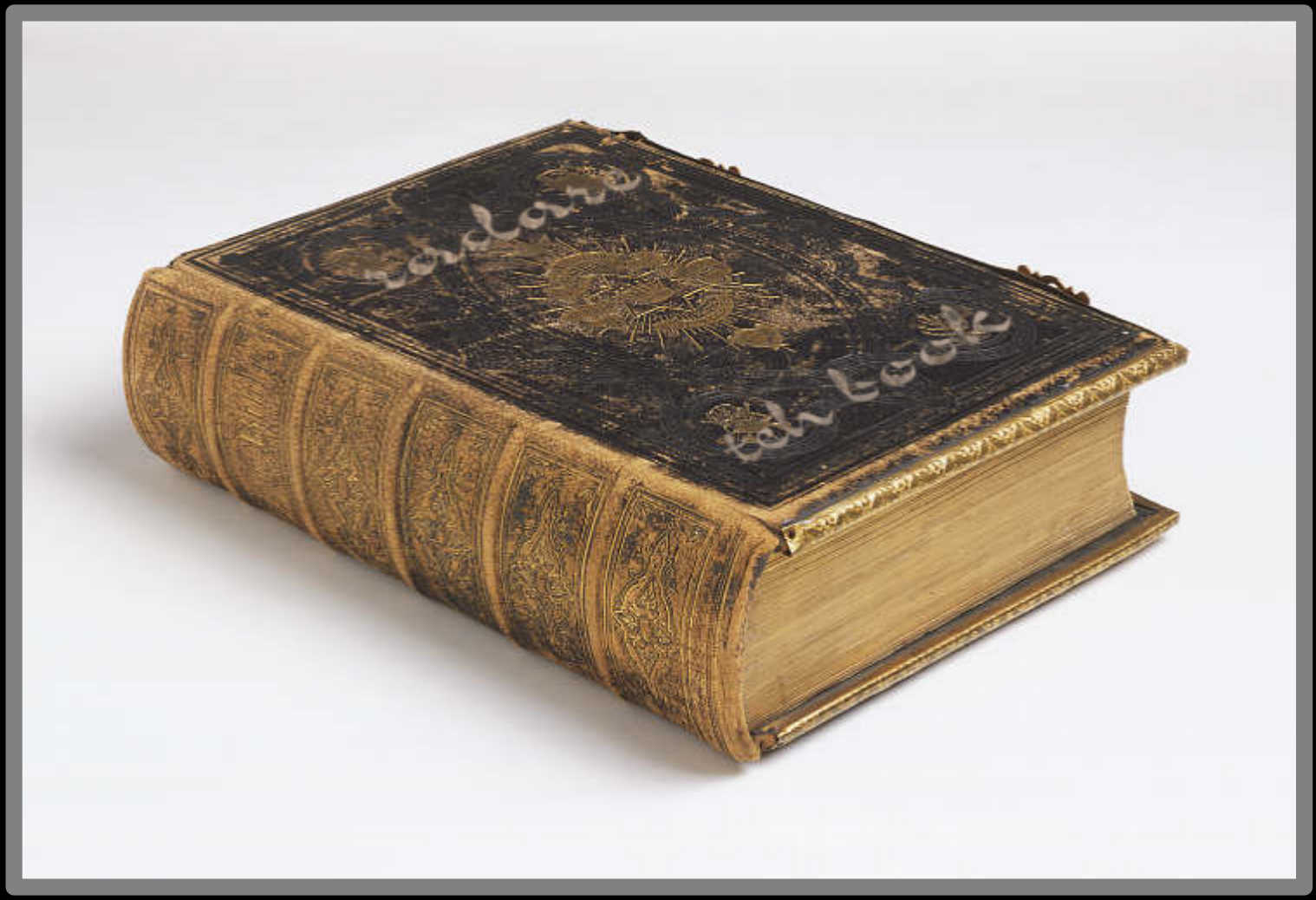
$ radare -i unpack.py -d ./target

Multiple language bindings
 - Python, Perl, LUA

API and helpers
 - Code analysis
 - Search
 - Flags, symbol management
 - Debugger access
 - Full control over radare thru
   str=r.cmd(str)

The book

http://radare.nopcode.org/get/radare.pdf



The human-radare interface

The end

Enjoy :)

http://radare.nopcode.org/

pancake@nopcode.org