My Methodology of Software Reverse Engineering

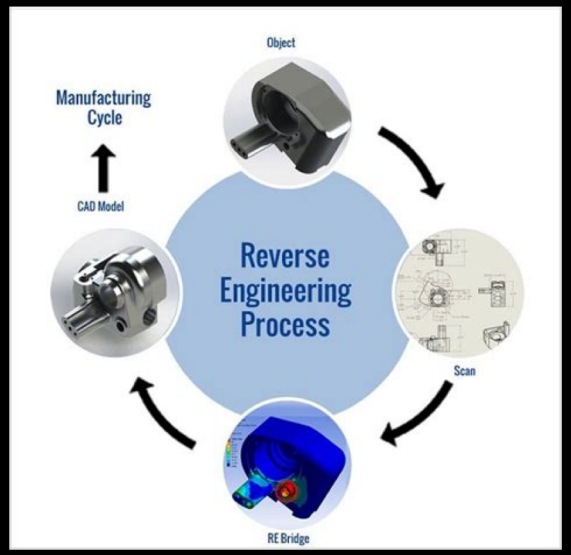
Atum

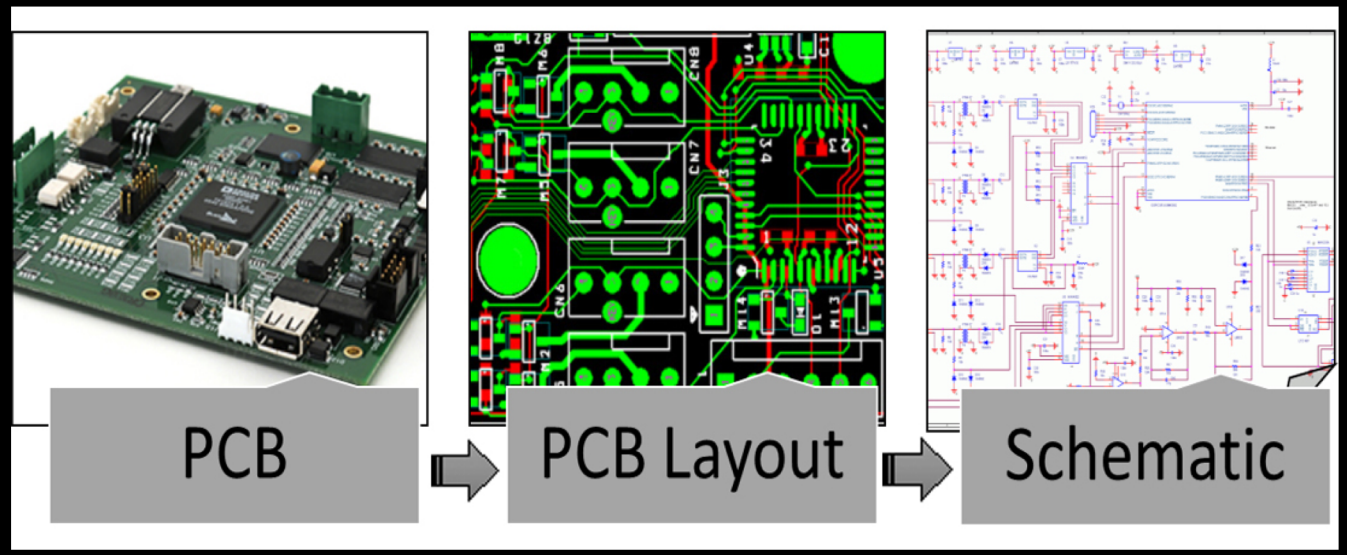
About me

- Atum
  - CTF Player @Eur3kA @blue-lotus @Tea Deliverers
  - Master candidate @ICST SECLAB, Peking University
  - http://atum.li lgcpku@gmail.com
- Software Security
  - Focus on Vulnerability Discovering & Exploiting & Defense
  - Active in CTF, PWN/Reverse

Reverse Engineering

- The process by which a man-made object is deconstructed to reveal its designs, architecture, or to extract knowledge from the object.



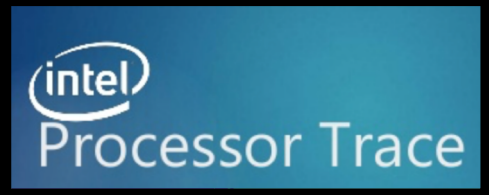


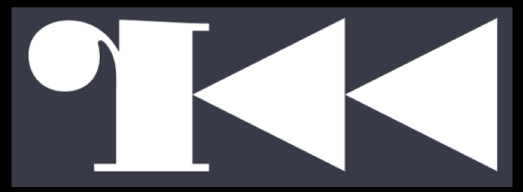
Software Reverse Engineering

- The practice of analyzing a software system, either in whole or in part, to extract design and implementation information
- How?
  - Open IDA -> auditing -> debugging -> auditing -> got the idea
  - or..
  - anything else?
- The key is to figure out what is going on.

Tools

- Disassembler
- Tracer
- Debugger
- Decompiler
- Emulator
- Symbolic Execution







Warmup Challenge Examples

- XMAN 2015 re0,re1,re2
- HCTF 2017 Evr
- QWB CTF 2018 simplecheck,hide

Regular Reverse Engineering

- ELF/PE/Mach-O file with x86/x64/arm ISA etc.
- How?
- Open IDA : audit -> debug -> audit -> mission accomplished?
- auditing is actually not the first step!

Information gathering

- strings/file/binwalk/IDA etc. -> google/github
- Examples:
  - CISCN RE Challenge
  - Kugou Music

Locate ”the crucial code”

- Some ways
  - control flow
  - data cross-reference(data xref)
  - code cross-reference(code xref)
  - memory searching + r/w breakpoint
  - tracing
  - anything else that can help you
- Example
  - Youku Client Reversing

REAL Reverse Engineering

- Some tips we have to remember
  - source code is written by programmer
  - binary is generated by compiler
  - reusing opensource code is common
  - binary is executable
  - be patient
  - 70% guessing + 30 reversing
- what do they mean?

Regular Pattern of Code

- 区分人写的代码跟编译器自己加上去的代码
- 区分库代码和程序代码
- 不需要仔细逆向 code add by compiler

Regular Pattern of Binary

- binary layout pattern
  - |executable|lib1|lib2|lib3|
- identify the code optimized by compiler

Identify the open source code

- string xref
- code style
- 开发自动化识别工具 (on going)

Dynamic Analysis

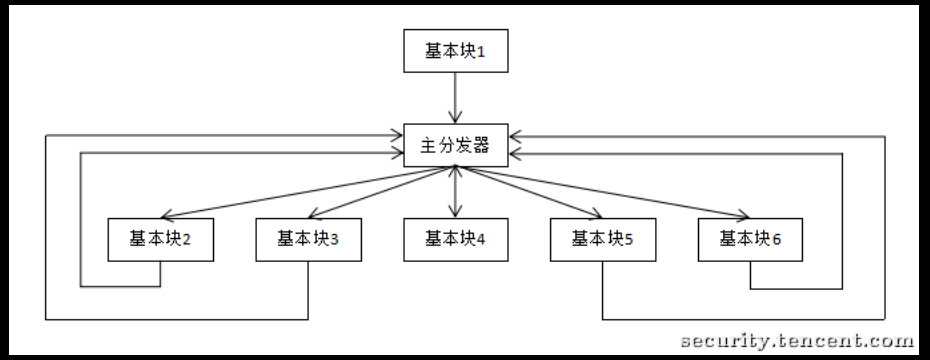
- Identify the key code, verify the guessing
- Debugging
- Tracing
- Symbolic Execution
- Taint Analysis

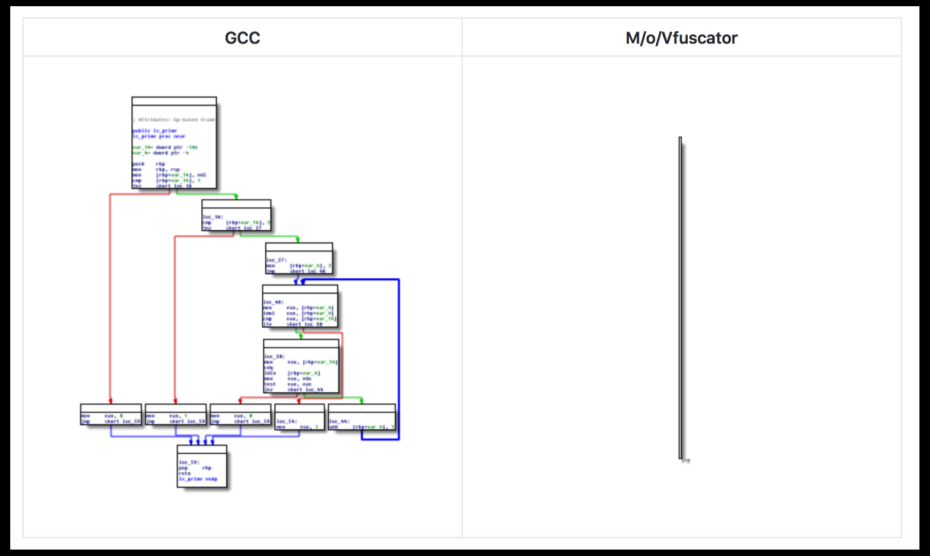
Reverse code block by block

- common algorithm identification
  - Tea/XTea/XXTea/IDEA/RC4/RC5/RC6/AES/DES/IDEA/MD5/SHA256/SHA1 etc.
  - 大数加减乘除/最短路等传统算法
- common data structure identification
  - graph/tree/hash table etc.
- common designed pattern identification
  - proxy stub etc.

Obfuscation

- Obfuscation Techniques Examples
  - ollvm, control flow flatten
  - movfuscator
  - push rax, ret
  - vm/self modify code
- Deobfuscation
  - the key is to recover the control flow
  - simulation execution/ symbolic execution





Packer

- various types of packers
  - unpack -> execute
  - unpack -> execute -> unpack -> execute ..
  - unpack -> [decoder | encoded code] -> decode -> execute
  - run the virtual machine
- case by case
- esp law
- read /proc/[pid]/mem

Name	Latest stable	Software license	x86-64 support
.netshrink	2.7 (July 2, 2016)[1]	Proprietary	Yes
Armadillo	9.62 (June 7, 2013)	Proprietary	Yes
ASPack	2.40 (November 2, 2016)	Proprietary	Yes
ASPR (ASProtect)	2.76 (November 2, 2016)	Proprietary	Yes
BoxedApp Packer	3.3 (July 26, 2015)	Proprietary	Yes
CExe	1.0b (July 20, 2001)	GPL	No
dotBundle	1.3 (April 4, 2013)[2]	Proprietary	Yes
Enigma Protector	6.10 (January 24, 2018)[3]	Proprietary	Yes
EXE Bundle	3.11 (January 7, 2011)[4]	Proprietary	?
EXE Stealth	4.14 (June 29, 2011)[5]	Proprietary	?
eXPressor	1.8.0.1 (January 14, 2010)	Proprietary	?
FSG	2.0 (May 24, 2004)[6]	Freeware	No
kkrunchy src	0.23a4 (Unknown)	BSD	No
MEW	1.1 (Unknown)	Freeware	No
MPRESS	2.19 (January 2, 2012)	Freeware	Yes
Obsidium	1.6 (April 11, 2017)[7]	Proprietary	Yes
PELock	2.06 (August 15, 2016)[8]	Proprietary	No
PESpin	1.33 (May 3, 2011)	Freeware	Yes
Petite	2.4 (September 22, 2016)	Freeware	No
RLPack Basic	1.21 (October 31, 2008)	GPL	No
Smart Packer Pro	1.9.9 (July 5, 2016)	Proprietary	Yes
Themida	2.4.6 (February 17, 2017)	Proprietary	Yes
UPX	3.94 (May 12, 2017)	GPL	experimental
VMProtect	3.1.1 (April 24, 2017)	Proprietary	Yes
XComp/XPack	0.98 (February 18, 2007)	Freeware	No

Anti-debugging

- Debugger Detection
  - API call, isDebuggerPresent()
  - try {int3}, catch{}
  - timestamp
  - etc..
- Debugger Interfering
  - DebugPort Overwrite
  - Self Debugging
  - etc..

Regular RE Challenge Examples

- N1CTF 2018 baby_neural_network
- TCTF 2018 Quals udp
- TCTF 2018 Final vtp
- DEFCON 26 CTF Qual preview
- RCTF 2018 magic
- Google CTF 2018 keygenme

Irregular Reverse Engineering

- Any format file with any architecture
  - lua/python/java/lua-jit/haskell/applescript/js/solidity/webassembly/etc..
  - firmware/raw bin/etc..
  - chip8/avr/clemency/risc-v/etc..
- Open IDA -> binaryfile
- How?

Find Tools

- Binary Parser
- Disassembler
  - disassembler is necessary
- Tracer
  - disassembler + tracer = debugger
  - trace replay
- Debugger
- Decompiler
  - 得之我幸，失之我命

Preparation

- read docs
- toolchains?
- tutorial?
- Example:
  - haskell

Find Binary Parser

- google大法好
- parse the unknown binary format
- firmware
  - rebase the binary
  - recover the symbol table
- other unknown binary formats
  - use strings/binwalk etc.
  - find any clues you can
  - use IDA pro/radare2/binary.ninja interface. e.g. IDA loader

Find Disassembler

- google “xxx disassembler/xxx IDA etc.”
  - AVR IDA
- human disassembler (笑)
- IDA Pro/radare2/binary.ninja interface
  - eg. IDA processor

Find tracer and debugger

- google 大法好 again
- tracer
  - try official tracer?
- debugger
  - gdb-multiarch
  - qemu
  - emulator
  - trace replay
- Example
  - solidary

Start regular reverse engineering

- Information gathering
- locate “the crucial code”
- REAL Reverse Engineer
- again: be patient
- again: 70% guessing + 30 reversing

How to audit assemble code

- find code pattern
  - loop, if else, etc.
- read assemble block by block

Irregular RE Challenge Examples

- Plaid CTF 2018 apl
- SECCON CTF 2017 printf_machine
- CodeGate CTF 2017 easy_serials
- *CTF 2018 wasm
- Nuit du hack CTF Quals 2018 AssembllyMe
- QWB CTF qual 2018 re
- N1CTF 2018 patient

Thank You

Questions are welcome