esil - universal il

ESIL - Intermediate Language for radare2 toolset

Anton Kochkov (@akochkov)

April 6, 2016

ZeroNights 11-2015

anton kochkov

- Moscow, Russia
- Love Reverse Engineering, foreign languages and travel
- Member of R2 crew, radare2 evangelist
- Security Code Ltd.

intermediate languages

what is intermediate language?

- *Intermediate language is the language of an abstract machine designed to aid in the analysis of computer programs. Intermediate Language - Wikipedia 2015*
- Heavily used for academic research and real world tools
- Vital for decompilation process
- Base for various kind of applications - SMT, AEG, AEP, etc

reil[1]

- Invented by Zynamics company
- BinNavi, BinDiff were both based on top of REIL
- x86, ARM, PowerPC architectures supported
- VM - Infinite memory
- VM - Infinite range of registers
- Missing Floating Point
- Written in Java

[1] Sebastian Porst Thomas Dullien (2009). *REIL: A platform independent intermediate representation of disassembled code for static code analysis*. In:

reil

- 17 instructions
- Register aliases (eax, ebx, r0, . . .)[2]

[2] *REIL description - Zynamics* (2005).

bap

- BAP - Binary Analysis Platform[3]
- IL itself called BIL
- Well-maintained framework, various tools included
- Integration with another tools like: TEMU, libVEX, IDA Pro, Qira, ...
- Targeted for x86, ARM
- Missing Floating Point

[3] Edward J. Schwartz David Brumley Ivan Jager and Spencer Whitman (2014). *The BAP Handbook*. In:

bitblaze (vineil/vex)

- BitBlaze[4] - also a platform, like BAP
- Uses two intermediate languages
- Vine IL - "low-level" language
- VEX IL (libVEX from valgrind) - "high-level" language
- Written in OCaml + C++

[4] Heng Yin Dawn Song David Brumley, Juan Caballero, and Ivan Jager (2008). *BitBlaze: A New Approach to Computer Security via Binary Analysis*. In:

vine il[5][6]

- Implicitly require description of all side-effects
- Quite similar to ESIL
- Very useful for direct code emulation
- Too much unused information

[5] BitBlaze Team (2009). *Vine Installation and User Manual*. In:
[6] David Brumley (2008). *Analysis and Defense of Vulnerabilities in Binary Code*. In:

vex il

- Infinite memory
- Infinite range of registers
- Support types
- “Variable scope” support
- Used in Valgrind and a few other programs
- Well-tested and actively maintained

rreil, openreil, mail

- RREIL[7] - modern and flexible alternative to REIL
- RREIL - supports types
- RREIL - unique conceipt of “domains”
- MAIL - IL, constructed specifically for Malware analysis
- MAIL - the only IL, which allows polymorph programs
- RREIL and MAIL - both lacks Floating Point support

[7] Bigdan Mihaila Alexander Sepp and Axel Simon (2011). *Precise Static Analysis of Binaries by Extracting Relational Information*. In:

rreil, openreil, mail

- OpenREIL[8] - reinvention and attempt to spruce up REIL
- OpenREIL - self-contained and ready to use framework, like BAP
- OpenREIL has major differencies from the original REIL
- Based on libVEX and has embedded support of SMT-solving

[8] Dmytro Oleksiuk (2015). *OpenREIL GitHub repository*. https://github.com/Cr4sh/openreil.

esil - what's different and what's common

short description

- Evaluable Strings Intermediate Language[9]
- Based on RPN (Reverse Polish Notation)(for the sake of speed)
- Designed for evaluation and emulation, not human-reading
- Low-level, pretty much alike Vine IL
- Small set of the instructions
- Implicit specification of all side-effects for each instruction

[9] Radare2 Team (2015a). *ESIL description*.

short description

- Designed with a wide range of different architectures in mind
- Infinite memory
- Infinite set of registers
- Register aliases (“native” names, like “eax” or “cpsr”)
- Ability to call external functions (+syscalls)
- Ability to implement “custom ops” easily
- Without Floating Point support (planned, though)

esil operands

Table 1: ESIL Operands[10]

ESIL Opcode	Operands	Name	Desription
$	src	Syscall	syscall
$$	src	Instruction address	Get address of current instruction
==	src,dst	Compare	v = dst - src ; update_eflags(v)
<	src,dst	Smaller	stack = (dst <src)
<=	src,dst	Smaller or Equal	stack = (dst <= src)
>	src,dst	Bigger	stack = (dst >src)
|=	src,reg	OR eq	reg = reg | src

[10]Radare2 Team (2015b). *ESIL Instruction Set*.

practical usage

radare

[11]

[11] *Radare advertisement in Berlin's U-Bahn* (2015).

radare2 tools

- rax2
- rabin2
- rasm2
- radiff2
- rafind2
- rahash2
- radare2
- r2pm
- rarun2/ragg2/ragg2-cc

1 command <—>1 reverse-engineering'notion

1. Every character of the command has some meaning (w = write, p = print)
2. Usually they're simple abbreviations pdf = p <->print d <->disassemble f <->function
3. Short usage message for each command can be printed with **cmd?**, e.g. pdf?,?, ???, ???, ?$?, ?@?

radare2 — important commands of cli-mode

1. **r2 -A** или **r2** + **aaa** : Анализ
2. **s** : seek to the address or flag
3. **pdf** : print disassembly for function
4. **af?** : perform function analysis
5. **ax?** : do analysis for XREF
6. **/?** : various kinds of search
7. **ps?** : print string
8. **C?** : comments
9. **w?** : write bytes (hex, assembly, etc)

radare2 — visual mode

radare2 — important commands of visual mode

1. **V?** or just **?** : Hotkeys help
2. **p/P** : circle between various visual modes
3. hjkl/arrows navigation
4. **o** : go to the offset/address
5. **e** : show all config variables
6. **v** : show the functions list
7. **_** : HUD
8. **V** : ASCII Graph
9. **0-9** : jump to the corresponding function
10. **u** : Undo

emulation using esil

- **ae*** - evaluate and show r2 commands
- **aei** - VM initialization
- **aeim** - VM memory/stack setup
- **aeip** - to set IP (Instruction Pointer) for VM
- **aes** - step in ESIL emulation
- **aec[u]** - continue [until]
- **aef** - emulate whole function by name

emulation using esil

- DEMO

embedded controller - 8051 - esil vm[12]

- **r2 -a 8051 ite_it8502.rom**
- **. ite_it8502.r2**
- **e io.cache=true** to cache IO while emulating
- run **aei**
- run **aeim**
- run **aeip** to start from current IP
- **aecu [addr]** to emulate until IP = [addr]

[12] *ESIL emulation in radare2* (2014).

full fledged emulation in vm

- Allows to start ESIL VM with predefined properties
- Allows to run the code (e.g. unpack) in this VM
- Allows to set callbacks on some opcodes
- Allows to translate syscalls into real ones (optional)
- Good example - unpacking Baleful code[13]

[13]Skuater (2015). *Reverse Engineering Baleful Virtual Machine with radare2*. In:

using esil emulation in analysis

- Using emulation of the code to find indirect jumps
- Using emulation to find some strings addresses
- Started by **aae**
- **aae** is a part of **aaaa** command

using background emulation to improve disassembly output

- Shows possible registers and memory values in comments
- Run ESIL VM with default properties in background
- Show “likely/unlikely” for conditional jumps
- **e asm.emu=true**

using background emulation to improve disassembly output

- DEMO

converting into another ils - openreil

- OpenREIL - actively maintained framework
- Ability to perform symbolic execution using SMT solver
- Radare2 has a special command to translate ESIL into OpenREIL
- **aetr**

converting into another ils - openreil

- DEMO

embedded controller - 8051 - esil2reil

- **r2 -a 8051 ite_it8502.rom**
- **. ite_it8502.r2**
- run **pae 36** to show ESIL of the function 'set_SMBus_frequency'
- run aetr `pae 36` to translate ESIL into REIL[14]
- Save this output into the file/pipe and send as OpenREIL input
- Could be easily automated using r2pipe script

[14] Dmytro Oleksiuk (2015). *OpenREIL GitHub repository*. https://github.com/Cr4sh/openreil.

radeco il and radeco decompiler

esil -> radeco[16]

- Uses ESIL as input
- Request more metadata from radare2 (xrefs, functions, etc)
- Using r2pipe to talk to radare2
- Written in pure Rust
- Biggest part of it written during GSoC 2015
- Authors are: Sushant Dinesh and David Kreuter[15]
- GSoC 2015 was done under the Openwall's project umbrella

---

[15] *Radeco GSoC 2015 report* (2015).

[16] Radare2 Team (2015c). *Radare2 GitHub repository*. `https://github.com/radare/radeco`.

why radeco?

- Existent FOSS decompilers are old and not use modern research
- Interesting methods to decompile rarely implemented in FOSS tools
- Radare2 as a RE suite need a good decompiler
- Challenging still viable task for Google Summer of Code

radeco il description

- Based purely on graphs
- Based on some concepts from RREIL and MAIL
- Simplification to SSA form while lifting ESIL -> Radeco IL
- Automatically performed DCE (Dead Code Elimination)
- Types inference[17]

[17]Thanassis Avgerinos JongHyup Lee and David Brumley (2011). *TIE: Princpled Reverse Engineering of Types in Binary Programs*. In:

radeco demo

- DEMO

future improvements

supported architectures

- Currently supported: x86, ARM, GameBoy, 8051, etc
- Goal is to add support for all architectures in radare2
- CPU/SoC/chip profiles for a slight differences between them

instruction sets

- Floating point (LLVM/McSema)[18]
- SIMD instructions (SSE, AVX, Neon, etc)
- VLIW and parallel execution (for DSP architectures emulation)

[18] *StackOverflow: floating point in ILs* (2014).

visual debugging and tracing

- General UI improvements
- Simple representation of diffs between emulation and native execution
- Auto removing dead ways/blocks from ASCII graphs
- Integration with WebUI and Bokken[19]

[19] *Bokken* (2015).

radeco improvements

- pseudo-C code emission
- Wider support for various: native and custom types
- Recalculating results on the fly, depending from debugging results
- Types inference and function/classes autorecognition[2021]

[20]Thanassis Avgerinos JongHyup Lee and David Brumley (2011). *TIE: Princpled Reverse Engineering of Types in Binary Programs*. In:

[21]Wei Huang Xue Lei Wenqing Fan, Yixian Yand, and Zhongxian Li (2015). *IL Optimization: Detecting and Eliminating Redundant Eflags by Flag Relevant Chain*. In:

references

references

- Alexander Sepp, Bigdan Mihaila and Axel Simon (2011). *Precise Static Analysis of Binaries by Extracting Relational Information*. In:
- *Bokken* (2015).
- Brumley, David (2008). *Analysis and Defense of Vulnerabilities in Binary Code*. In:
- David Brumley Ivan Jager, Edward J. Schwartz and Spencer Whitman (2014). *The BAP Handbook*. In:
- Dawn Song David Brumley, Heng Yin, Juan Caballero, and Ivan Jager (2008). *BitBlaze: A New Approach to Computer Security via Binary Analysis*. In:
- *ESIL emulation in radare2* (2014).
- *Intermediate Language - Wikipedia* (2015).

a lot of them II

- JongHyup Lee, Thanassis Avgerinos and David Brumley (2011). *TIE: Princpled Reverse Engineering of Types in Binary Programs*. In:
- Oleksiuk, Dmytro (2015). *OpenREIL GitHub repository*. https://github.com/Cr4sh/openreil.
- *Radare advertisement in Berlin's U-Bahn* (2015).
- *Radeco GSoC 2015 report* (2015).
- *REIL description - Zynamics* (2005).
- Skuater (2015). *Reverse Engineering Baleful Virtual Machine with radare2*. In:
- *StackOverflow: floating point in ILs* (2014).
- Team, BitBlaze (2009). *Vine Installation and User Manual*. In:
- Team, Radare2 (2015a). *ESIL description*.
- — (2015b). *ESIL Instruction Set*.

a lot of them III

- Team, Radare2 (2015c). *Radare2 GitHub repository*. https://github.com/radare/radeco.
- Thomas Dullien, Sebastian Porst (2009). *REIL: A platform independent intermediate representation of disassembled code for static code analysis*. In:
- Xue Lei Wenqing Fan, Wei Huang, Yixian Yand, and Zhongxian Li (2015). *IL Optimization: Detecting and Eliminating Redundant Eflags by Flag Relevant Chain*. In: