

esil - универсальный il

ESIL - Intermediate Language для radare2

---

Anton Kochkov (@akochkov)

26 ноября 2015 г.

ZeroNights 11-2015

- Москва, Россия
- Хобби - реверс инжиниринг, языки и путешествия
- Участник R2 crew и евангелист radare2
- ООО Код Безопасности

краткий обзор intermediate languages

---

# что такое intermediate language

- *Intermediate language is the language of an abstract machine designed to aid in the analysis of computer programs. Intermediate Language - Wikipedia 2015*
- Используется как в теории (и практике) компиляции
- Аналогично незаменим и для декомпиляции
- Огромное количество разных академических и практических воплощений
- Основа для высокоуровневого анализа - SMT, AEG, AEP, etc

- Изобретен компанией Zynamics
- Использовался в продуктах BinNavi, BinDiff
- Поддерживает архитектуры x86, ARM, PowerPC
- Бесконечная память VM
- Бесконечное количество регистров VM
- Без Floating Point
- Оригинальные утилиты написаны на Java

---

<sup>1</sup>Sebastian Porst Thomas Dullien (2009). *REIL: A platform independent intermediate representation of disassembled code for static code analysis*. B:

- 17 инструкций
- Алиасы для реальных регистров (eax, ebx, r0, ...) <sup>2</sup>

---

<sup>2</sup>REIL description - Zynamics (2005).

- BAP - Binary Analysis Platform<sup>3</sup>
- Настоящее имя IL - BIL
- Развитый фреймворк
- Интеграция с другими утилитами - TEMU, libVEX, IDA Pro, Qira, ...
- Ориентирован на x86, ARM
- Без Floating Point

---

<sup>3</sup>Edward J. Schwartz David Brumley Ivan Jager и Spencer Whitman (2014). *The BAP Handbook*. В:

# bitblaze (vineil/vex)

- BitBlaze<sup>4</sup> - платформа, аналогичная BAP
- Имеет несколько промежуточных языков
- VEX IL (libVEX из valgrind) - “нижний” уровень
- Vine IL - “верхний” уровень
- Написан на OCaml + C++

---

<sup>4</sup>Heng Yin Dawn Song David Brumley, Juan Caballero и Ivan Jager (2008). *BitBlaze: A New Approach to Computer Security via Binary Analysis*. B:



- Явное указание всех side-эффектов для команд
- Ближе всего к ESIL
- Оттестирован и используется в Valgrind
- Хорошо подходит для эмуляции кода
- Избыточен

- Бесконечная память
- Бесконечное количество регистров
- Поддержка типов
- Поддержка “variable scope”

---

<sup>5</sup>BitBlaze Team (2009). *Vine Installation and User Manual*. В:

<sup>6</sup>David Brumley (2008). *Analysis and Defense of Vulnerabilities in Binary Code*. В:

- RREIL<sup>7</sup> - гибкий язык, замена REIL
- RREIL - поддержка типов
- RREIL - интересная концепция “доменов”
- MAIL - IL, созданный для анализа Malware
- MAIL - позволяет программе перезаписывать себя саму
- RREIL и MAIL - опять отсутствие Floating Point

---

<sup>7</sup>Bigdan Mihaila Alexander Sepp и Axel Simon (2011). *Precise Static Analysis of Binaries by Extracting Relational Information*. B:

- OpenREIL<sup>8</sup> - проект, созданный для использования REIL в современных реалиях
- OpenREIL - полноценный фреймворк, как и BAP
- OpenREIL отличается от оригинального REIL
- Использует libVEX и имеет поддержку SMT-solving

---

<sup>8</sup>Dmytro Oleksiuk (2015). <https://github.com/Cr4sh/openreil>.

esil - сходства и отличия

---

- Evaluable Strings Intermediate Language<sup>9</sup>
- Использует обратную польскую нотацию (для скорости)
- Не предназначен для чтения человеком
- По “уровню” приближен к VEX
- Небольшое количество инструкций
- Полный учёт side-эффектов

---

<sup>9</sup>Radare2 Team (2015a). *ESIL description*.

- Сроектирован для большого количества архитектур
- Бесконечная память
- Бесконечные регистры
- Алиасы (использование “нативных” имен)
- Есть возможность вызывать куски нативного кода (+syscall)
- Возможность добавления “custom ops”
- Нет Floating Point (будет в следующей версии)

Таблица 1: ESIL Operands<sup>10</sup>

ESIL Opcode	Operands	Name	Description
\$	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

---

<sup>10</sup>Radare2 Team (2015b). *ESIL Instruction Set*.



практическое применение

---



11

---

<sup>11</sup>*Radare advertisement in Berlin's U-Bahn (2015).*

# radare2 утилиты

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

# 1 command $\longleftrightarrow$ 1 reverse-engineering' notion

1. Каждый символ команды что-то значит (w = write, p = print)
2. Обычно команды - это аббревиатуры действий pdf = p  $\longleftrightarrow$  print  
d  $\longleftrightarrow$  disassemble f  $\longleftrightarrow$  function
3. Доступна короткая справка для каждой команды cmd?,  
например pdf?, ?, ???, ???, ?\$, ?@?

# radare2 — основные команды cli-режима

1. `r2 -A` или `r2 + aaa` : Анализ
2. `s` : Переход по указанному адресу
3. `pdf` : Дизассемблирование функции
4. `af?` : Анализ функции
5. `ax?` : Анализ XREF
6. `/?` : Поиск
7. `ps?` : Напечатать строку (print string)
8. `C?` : Комментарии
9. `w?` : Запись (hex, опкодов, etc)

radare2 — visual mode

---

# radare2 — основные команды визуального режима

1. **V?** или просто **?** : Помощь по командам
2. **p/P** : переключение между разными визуальными представлениями
3. Навигация с помощью стрелок/hjkl
4. **o** : переместиться по адресу
5. **e** : визуальный режим настроек
6. **v** : список функций
7. **\_** : HUD
8. **V** : ASCII Graph
9. **0-9** : Прыжок на функцию
10. **u** : Undo

# эмуляция участков кода

- **ae\*** набор инструкций
- **aei** - инициализация ESIL VM
- **aeim** - инициализация стека/памяти VM
- **aeip** - установка IP (Instruction Pointer)
- **aes** - step в режиме эмуляции ESIL
- **aec[u]** - continue [until]
- **aef** - эмуляция функции



- DEMO

- `r2 -a 8051 ite_it8502.rom`
- `. ite_it8502.r2`
- `'e io.cache=true'` для использования кеширования IO
- запустим `'aei'`
- запустим `'aeim'`
- запустим `'aeip'` для старта с момента указания команды
- `'aescu [addr]'` для эмуляции, пока не достигнем `IP = [addr]`

---

<sup>12</sup>*ESIL emulation in radare2* (2014).

- Использование “подсказок” ESIL при визуальной отладке
- DEMO

- Позволяет выполнить распаковку или выполнение в VM
- Хороший пример - использование ESIL для распаковки Baleful

# автоматическое отображение результатов эмуляции в дизассемблере

- Отображает в комментариях значения регистров и памяти
- Использует тот же механизм эмуляции кода ESIL VM
- Показывает likely/unlikely для условных переходов
- e asm.emu=true

# автоматическое отображение результатов эмуляции в дизассемблере

- DEMO

# конвертация в другие языки - openreil

- OpenREIL - развитый фреймворк
- Есть возможность использования SMT
- Добавлена возможность конвертации ESIL в OpenREIL
- Команда 'aetr'

- DEMO



- `r2 -a 8051 ite_it8502.rom`
- `. ite_it8502.r2`
- `run 'pae 36'` для показа ESIL представления функции `'set_SMBus_frequency'`
- `run 'aetr `pae 36`'` для конвертации строки ESIL в REIL<sup>13</sup>
- Сохранить вывод в файл и передать управление в OpenREIL
- Можно проделать всё вышеперечисленное с помощью `r2pipe` скрипта

---

<sup>13</sup>Dmytro Oleksiuk (2015). <https://github.com/Cr4sh/openreil>.

radeco il и radeco decompiler

---

- Использует ESIL в качестве входных данных
- Использует другие метаданные из radare2
- Соединяется с radare2 через r2pipe
- Написан на Rust

# причины появления декомпилятора

- Существующие FOSS декомпиляторы не учитывают последние исследования
- Академические (но интересные) идеи не имеют полноценной реализации
- Radare2 нуждается в декомпиляторе
- Хорошее и интересное задание для Google Summer of Code

- Графовое представление
- Взяты идеи из RREIL и MAIL
- Использование SSA на этапе лифтинга ESIL -> Radeco IL
- Встроенная поддержка DCE (Dead Code Elimination)
- Базовая возможность вывода типов<sup>14</sup>

---

<sup>14</sup>Thanassis Avgerinos JongHyup Lee и David Brumley (2011). *TIE: Principled Reverse Engineering of Types in Binary Programs*. В:

- DEMO

пути будущего развития

---

# поддерживаемые архитектуры

- Сейчас лучше всего поддерживаются x86, ARM, GameBoy, 8051, etc
- Глобальная цель - поддержка ESIL для всех архитектур в radare2
- Поддержка профилей для выбранных модификаций/вариаций процессоров



# поддерживаемые наборы инструкций

- Floating point (LLVM/McSema)<sup>15</sup>
- Векторные инструкции (SSE, AVX, Neon, etc)
- VLIW инструкции (для эмуляции кода DSP)

---

<sup>15</sup>*REIL description - Zynamics (2014).*

- Улучшение UI
- Возможность визуального сравнения эмуляции и нативного выполнения
- Устранение “мертвого” кода из ASCII графов на лету
- Интеграция в WebUI и Bokken

- Генерация C кода
- Поддержка нативных типов
- Синхронизация с отладкой
- Автовывод типов/распознавание объектов и классов<sup>1617</sup>

---

<sup>16</sup>Thanassis Avgerinos JongHyup Lee и David Brumley (2011). *TIE: Principled Reverse Engineering of Types in Binary Programs*. В:

<sup>17</sup>**il-optimization.**

references

---

## список литературы

---



Alexander Sepp, Bigdan Mihaila и Axel Simon (2011). *Precise Static Analysis of Binaries by Extracting Relational Information*. B:



Brumley, David (2008). *Analysis and Defense of Vulnerabilities in Binary Code*. B:



David Brumley Ivan Jager, Edward J. Schwartz и Spencer Whitman (2014). *The BAP Handbook*. B:



Dawn Song David Brumley, Heng Yin, Juan Caballero и Ivan Jager (2008). *BitBlaze: A New Approach to Computer Security via Binary Analysis*. B:



*ESIL emulation in radare2* (2014).



*Intermediate Language - Wikipedia* (2015).

## a lot of them II



JongHyup Lee, Thanassis Avgerinos and David Brumley (2011). *TIE: Principled Reverse Engineering of Types in Binary Programs*. B:



Oleksiuk, Dmytro (2015).

<https://github.com/Cr4sh/openreil>.



Radare advertisement in Berlin's U-Bahn (2015).



REIL description - Zynamics (2005).



REIL description - Zynamics (2014).



Team, BitBlaze (2009). *Vine Installation and User Manual*. B:



Team, Radare2 (2015a). *ESIL description*.



— (2015b). *ESIL Instruction Set*.



Thomas Dullien, Sebastian Porst (2009). *REIL: A platform independent intermediate representation of disassembled code for static code analysis*. B: