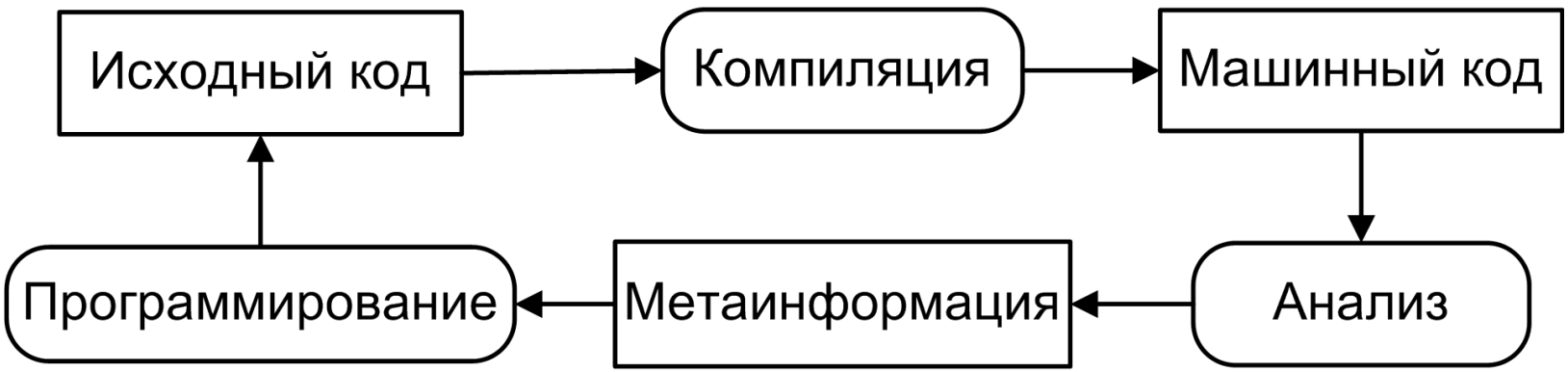
# Исследование методов генерации защищенного машинного кода

Диссертация на соискание степени магистра

Выполнил студент гр. 63601/2: А.Б. Терентьев
Научный руководитель: В.Ю. Аранов

26 июня 2014

# Проблема обратного проектирования программ




Метаинформация из скомпилированного кода может быть
Извлечена $\rightarrow$ Проанализирована $\rightarrow$ Модифицирована

# Критерии оценки методов

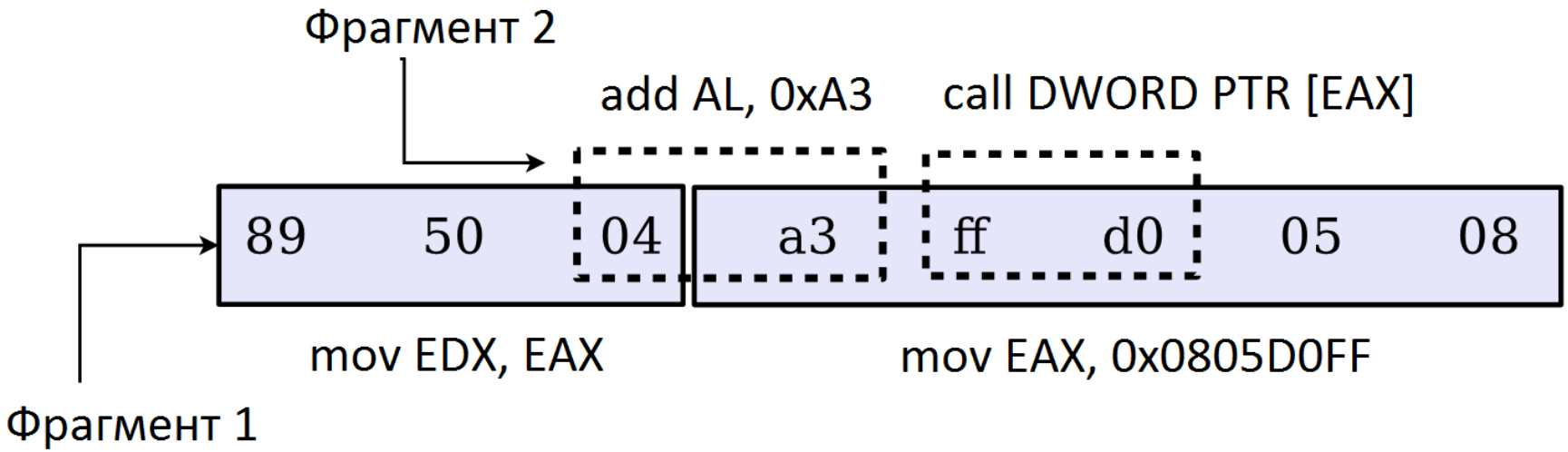
«Взломоустойчивость — число условных «человеко-часов», которые необходимо потратить для извлечения метаинформации достаточной для организации атаки[1]»

- Устойчивость к различным методам обратного проектирования программы
- Степень потери производительности программы по времени выполнения и использованной динамической памяти
- Временная трудоемкость метода построения защиты

[1] C. Collberg, *A Taxonomy of Obfuscating Transformations*, 1997

# Код с перекрывающимися инструкциями

«Перспективным и заманчивым представляется также создание перекрывающихся инструкций[2]»




Байты по одному и тому же адресу используются в нескольких инструкциях

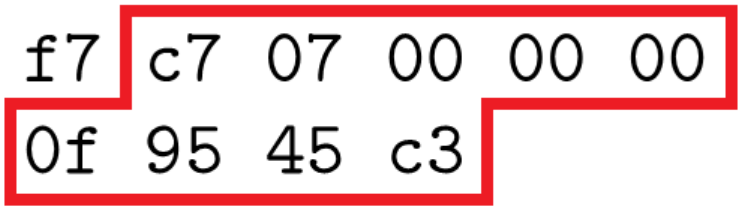
[2] С,И. Алейников, *Защита программ от дизассемблирования*, 2006

# Постановка задачи

- Разработка подходов к генерации кода с перекрывающимися инструкциями
- Разработка критерия оценки качества кода, сгенерированного с помощью этих подходов
- Реализация подходов для платформы x86
- Оценка качества генерируемого защищенного кода

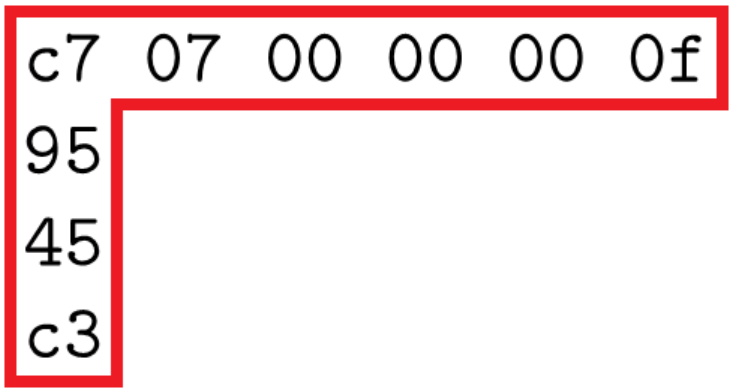
# Возвратно-ориентированное программирование

**Гаджет** — последовательность инструкций, содержащая ровно одну инструкцию передачи управления, находящуюся в конце




test EDI, 0x00000007
setnz EBP

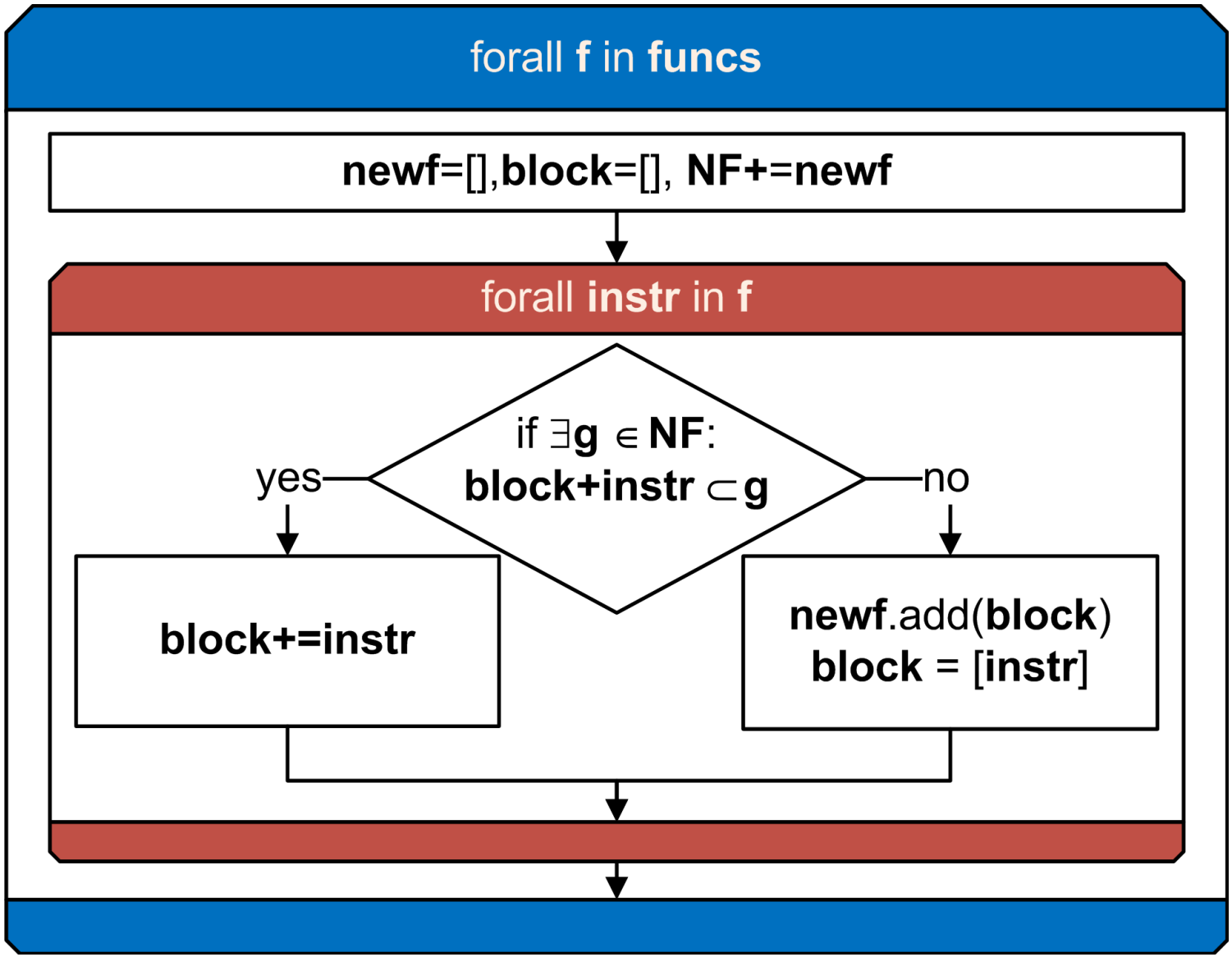
---



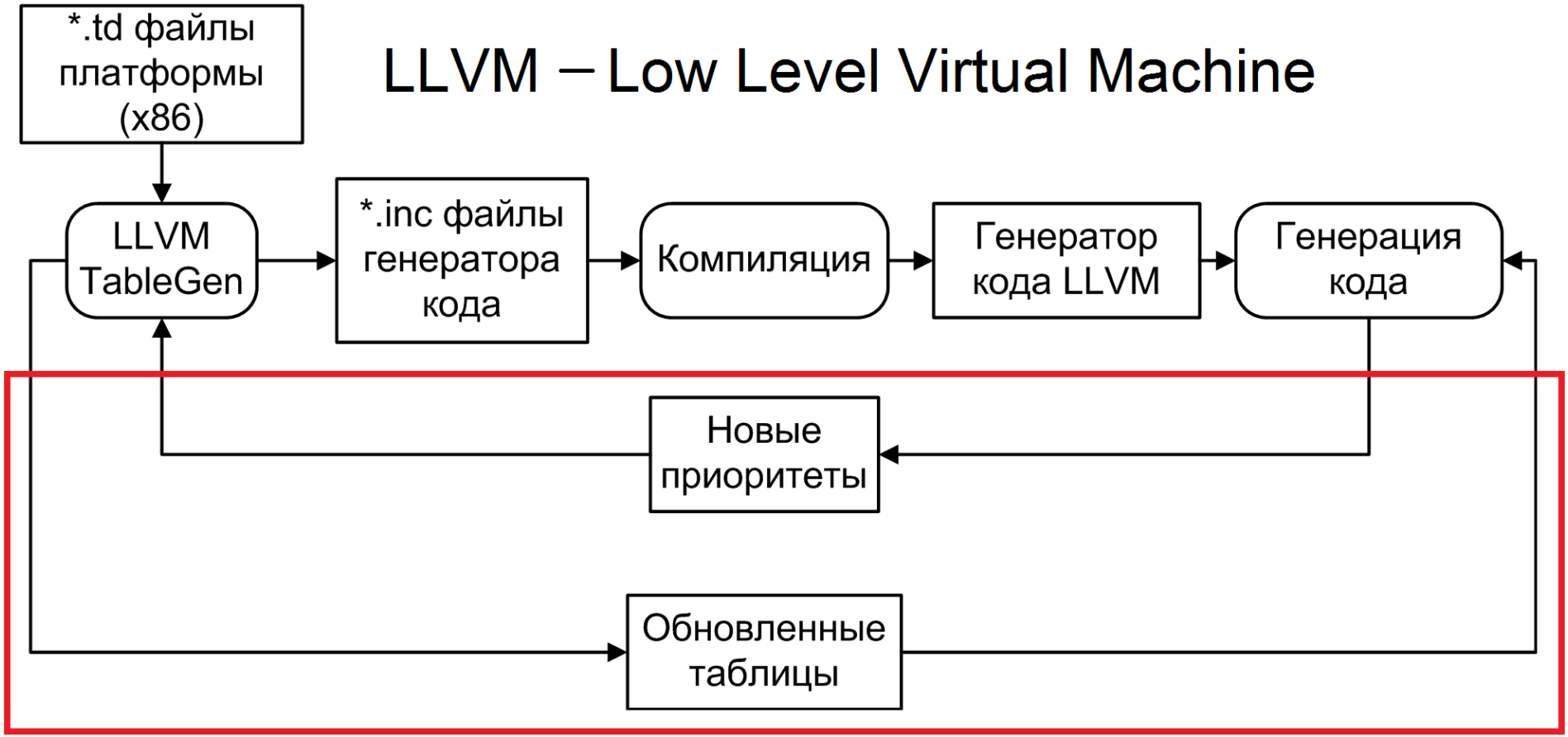

mov EDI, 0x0F000000
xchg EBP, EAX
inc EBP
ret

# Алгоритм

**Вход:**
funcs
**Выход:**
NF



# Увеличение количества гаджетов




**Временная сложность** в худшем случае — $O(n^2)$

$n$ - размер программы в байтах

# Критерий оценки. Требования

- Дано:
  - Код программы $P$(немодифицированный генератор кода LLVM3.3), размер $n$
  - Код программы $Q$(модифицированный генератор кода LLVM), размер $m$, $x_i, i = 1..m$ - значения перекрываемости байт(количество инструкций, элементом которых является $i$-ый байт)
  - $W_P(Q)$ - искомая функция
- Требования:
  - $W_P(P) \equiv 1$
  - Учитывать изменение размера:
    - $\forall Q : m = n, x_i = 1, i = 1 \ldots m : W_P(Q) \equiv 1$;
    - $\forall Q : m < n, x_i = 1, i = 1 \ldots m : W_P(Q) > 1$;
    - $\forall Q : m > n, x_i = 1, i = 1 \ldots m : W_P(Q) < 1$
  - $\forall Q : m = n, x_i = N, i = 1 \ldots m : W_P(Q) \equiv N$
- Цель: $W_P(Q) > 1$

# Критерий оценки

- Средняя перекрываемость $A\{x_1..x_k\}$: $V(A) = \frac{1}{k}\sum_{i=1}^{k} x_i$
- Удельная средняя перекрываемость:

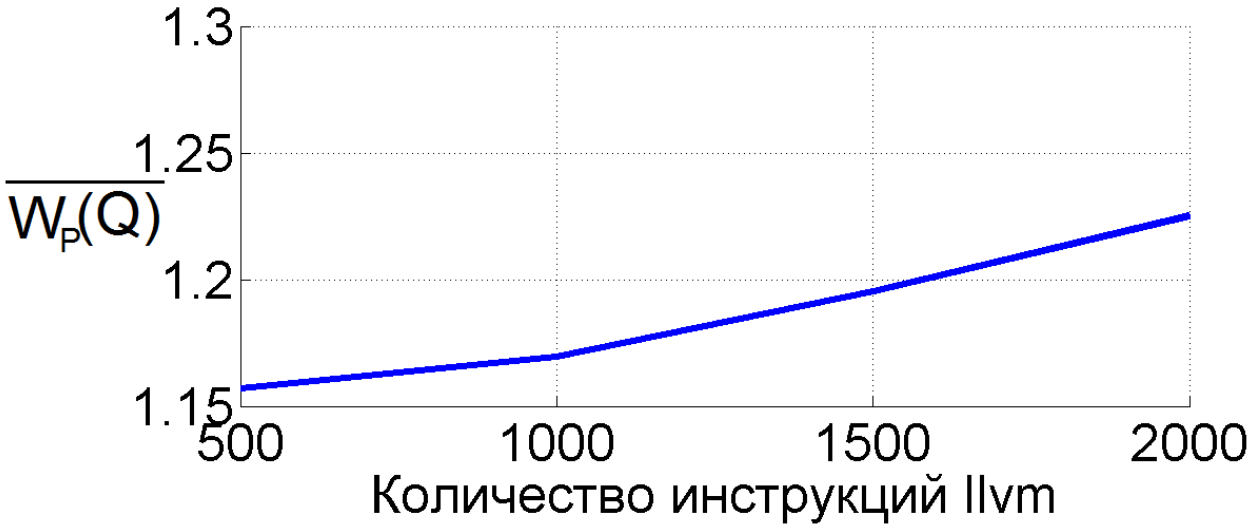
$$V_{spec}(A) = \frac{V(A)}{k} = \frac{1}{k}\sum_{i=1}^{k} x_i \Big/ k$$

- $f$ для программ $A, B$: $f(A,B) = \frac{V_{spec}(A)}{V_{spec}(B)}$
- Для программы $P$: $x_i \equiv 1$, $V_{spec}(P) = \frac{1}{n}$:

$$\boxed{W_P(Q) = f(Q,P) = \frac{\left(\sum_{i=1}^{m} x_i\right) * n}{m^2}}$$

# Оценка качества кодогенерации

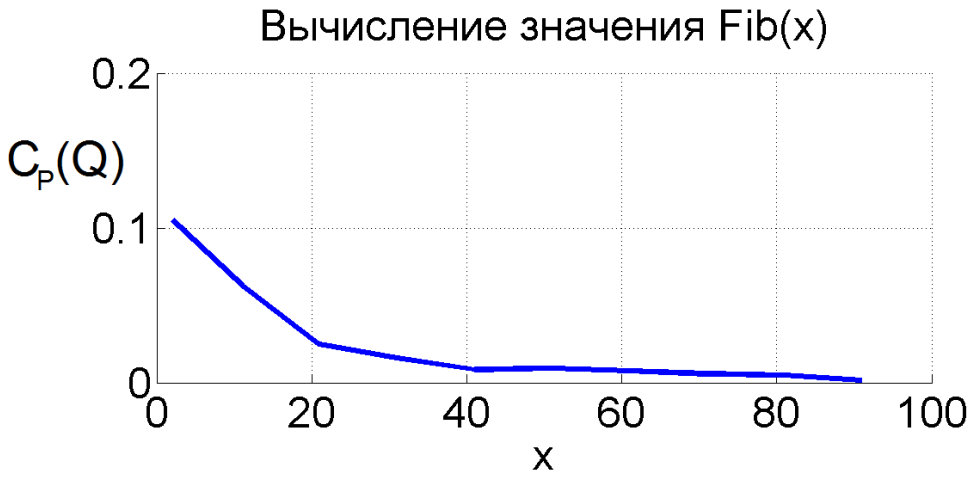
**llvm-stress** — генерация случайных .ll файлов.
50 файлов на тест, 2000 инструкций $\sim$ 6КБ(размер секций .text)




| Размер | $\overline{W_P(Q)}$ | $\sigma$ | MIN | MAX |
|---|---|---|---|---|
| 500 | 1.116 | 0.0156 | 1.017 | 1.753 |
| 1000 | 1.169 | 0.0113 | 1.006 | 1.614 |
| 1500 | 1.195 | 0.0046 | 1.084 | 1.376 |
| 2000 | 1.226 | 0.0085 | 1.113 | 1.450 |

# Исследование производительности

- $C_P(Q) = \frac{C(Q)-C(P)}{C(P)}$, $C(A)$ — количество тактов процессора, необходимое для запуска $A$
- $T_P(Q) = \frac{T(Q)-T(P)}{T(P)}$, $T(A)$ — время, необходимое для запуска $A$
- 100000 вызовов каждой функции, CPU — Core i7 2600K




| Тест | $T(P)$, с | $T(Q)$, с | $T_P(Q) * 100,\%$ | $C_P(Q) * 100,\%$ |
|---|---|---|---|---|
| Sin | 36.149 | 36.044 | -0.29% | +0.07% |
| Fact | 3.334 | 3.477 | +4.2% | +4.5% |
| Fib | 3.211 | 3.301 | +2.8% | +2.7% |

# Результаты анализа свойств полученного кода

- Затрудняет анализ и модификацию машинного кода
- Затрудняет использование стандартных программ для анализа — IDA, Hex-Rays, OllyDbg
- Может использоваться в комбинации с другими методами защиты машинного кода
- Иногда:
  - Модифицированная программа меньше оригинальной
  - Модифицированная программа быстрее оригинальной

# Результаты

- Разработаны новые подходы к генерации машинного кода с перекрывающимися инструкциями
- Подходы реализованы на основе генератора кода LLVM
- Разработана методика оценки результатов работы
- На основе предложенного критерия и результатов исследований производительности защищенного кода сделаны выводы о сравнительной эффективности разработанных алгоритмов

Доклад + статья на Spring/Summer Young Researchers' Colloquium on Software Engineering, 2014.

# Направление дальнейших исследований

- Алгоритм Витерби

| | ASSEMBLY | OPCODE | ASCII |
|---|---|---|---|
| 1 | push %esp<br>push $20657265<br>imul %esi,20(%ebx),$616D2061<br>push $6F<br>*jb short $22* | 54<br>68 65726520<br>6973 20 61206D61<br>6A 6F<br>*72 20* | There is a major |
| 2 | push $20736120<br>push %ebx<br>je short $63<br>*jb short $22* | 68 20617320<br>53<br>74 61<br>*72 20* | h as Star |

| 1 | Skip | 2 | Skip |
|---|---|---|---|
| **There is a major** | center of economic activity, suc | **h as Star** | Trek, including The Ed |

| Skip | 3 | Skip |
|---|---|---|
| Sullivan | **Show. The form** | er Soviet Union. International organization participation |

# Дополнительные измерения

Средняя перекрываемость байт

| Размер | $\overline{V(Q)}$ | MAX | MIN |
|---|---|---|---|
| 500 | 1.24 | 1.31 | 1.18 |
| 1000 | 1.28 | 1.41 | 1.21 |
| 1500 | 1.30 | 1.41 | 1.24 |
| 2000 | 1.33 | 1.49 | 1.25 |

Процент байт с перекрываемостью $> 1$

| Размер | AVG | MAX | MIN |
|---|---|---|---|
| 500 | 12.9 | 17.2 | 8.9 |
| 1000 | 13.6 | 16.5 | 9.8 |
| 1500 | 12.8 | 15.7 | 10.2 |
| 2000 | 13.1 | 15.6 | 11.6 |

## Low Level Virtual Machine (LLVM)

Универсальная система анализа, трансформации и оптимизации программ, реализующая виртуальную машину