ИСП РАН

# Применение диверсифицирующих и обфусцирующих преобразований для изменения сигнатуры программного кода

Нурмухаметов Алексей Раисович

1 декабря 2016 г.

Институт Системного Программирования

# Введение

**Обфускация** - приведение исполняемого кода программы к виду, сохраняющему ее функциональность, но затрудняющему анализ, понимание алгоритмов работы и модификацию.

**Диверсификация** - изменение программного кода с целью получения большого количества разных программ с одинаковой функциональностью.

# ИСП Обфускатор

# ИСП Обфускатор

- Основан на открытой компиляторной инфраструктуре LLVM**.
- Поддержка нескольких языков программирования (C, C++, Objective C, Objective C++).
- Поддержка множества целевых архитектур (ARM, x86/x86_64, Microblaze, MIPS, PowerPC, SPARC).
- Реализовано большое количество обфусцирующих преобразований.

** Часть преобразований реализована также в компиляторе GCC.

# Реализованные преобразования

- Приведение графа потока управления к плоскому виду.
- Перемещение локальных переменных в глобальную область видимости.
- Размножение тел функций.
- Переплетение нескольких функций в одну.
- Сокрытие вызовов функций.
- Создание несводимых участков в графе потока управления.
- Шифрование константных строк, используемых программой.
- Вставка в код фиктивных циклов из одной итерации.
- Разбиение целочисленных констант.
- Изменение структуры стека.
- Перестановка функций в модуле местами.

# Применение обфускации

- Затруднение обнаружения функциональности.
- Препятствование обратной разработке.
- Затруднение идентификации используемых компонентов с открытым исходным кодом.
- Расстановка водяных знаков на версиях программ для разных клиентов.
- Усложнение идентификации автора кода.
- Затруднение генерации эксплойтов на основе анализа патчей, закрывающих уязвимость.

# Затруднение идентификации используемых компонентов

# Постановка задачи

Затруднить идентификацию функций из библиотек с открытым исходным кодом в бинарном коде, не содержащем информации о символах.

# Актуальность

Существуют инструменты автоматического обнаружения, использованных функций.

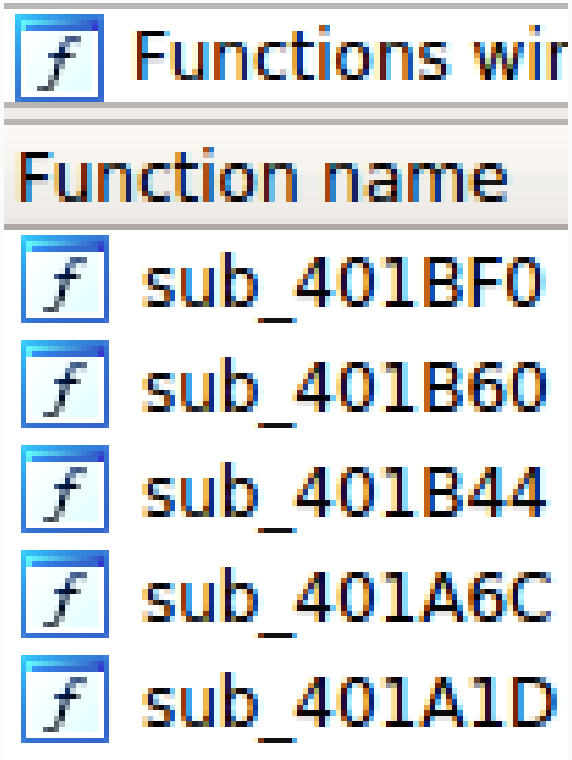
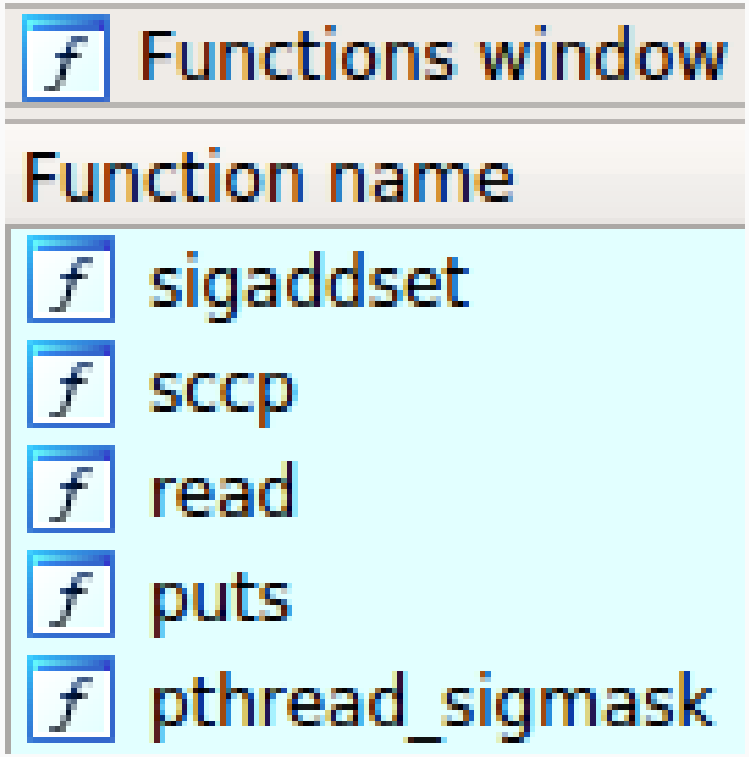
- Ida Pro FLAIR/FLIRT
- https://github.com/pzread/unstrip
- DynInst unstrip tool

# Ida Pro FLAIR/FLIRT

FLAIR (Fast Library Acquisition for Identification and Recognition) - набор утилит (pelf, sigmake) для генерации FLIRT сигнатур.

FLIRT (Fast Library Identification and Recognition Technology) - технология идентификации сгенерированных компилятором и статически связанных функций в программе по сигнатурам, сгенерированным FLAIR.

# Ida Pro FLAIR/FLIRT

| До | После |
| --- | --- |
|  |  |

# Применение ИСП Обфускатор для затрудения идентификации функций

# Тестовый пример

```
$ cat test.c
#include<stdio.h>
#include<stdlib.h>
int main(){
    puts("Hello world\n");
    system("ls");
    return 0;
}
```

# Сборка тестового примера

- Стандартной библиотекой Си (musl).
- Файл связан с двумя версиями стандартной библиотеки:
  - обычная,
  - обфусцированная ИСП Обфускатором.
- Из связанного файла убрана информация о символах.

# Результаты идентификации функций

| | оригинальный | обфусцированный* |
|---|---|---|
| всего функций | 71 | 229 |
| Ida Pro | 61 | 6 |
| pzread/unstrip | 19 | 0 |

* все обфускации включены

# Производительность

| | размер, МБ | libc_bench, мс |
|---|---|---|
| оригинальный | 2,5 | 6.5 |
| обфусцированный | 140,0 | 443.6 |
| разница в разах | 56 | 68 |

libc-bench - набор тестов для измерения производительности разных реализаций стандартной библиотеки Си

# Тесты при различной степени обфускации

| обфусцирующее преобразование | 1 | 2 | 3 |
|---|---|---|---|
| разбиение констант | + | + | - |
| вынос локальных перем. в глоб. обл. | + | + | - |
| вставка фиктивных циклов | + | + | - |
| приведение ГПУ к плоскому виду | + | - | - |
| перестановка функций в модуле | + | + | + |
| изменение структуры стека | + | + | + |
| переплетение функций | + | - | - |
| сокрытие вызовов функций | - | - | - |
| создание несводимых участков в ГПУ | + | + | - |

# Производительность при различной степени обфускации

| | размер | libc_bench | Ida Pro фун. |
|---|---|---|---|
| оригинал | 1 | 1 | 61/71 |
| все | 56 | 68 | 6/229 |
| 1 | 12.4 | 23 | 6/71 |
| 2 | 1.08 | 2.2 | 14/71 |
| 3 | 1 | ~1 | 57/71 |

Изменение производительности измеряется отношением к производительности оригинальной программы.

# Разработка и реализация специализированного преобразования

# Вставка "мертвых" инструкций

1. Добавляется глобальная переменная в каждый модуль.
2. В каждую функцию этого модуля добавляется арифметическая операция, меняющая значение этой глобальной переменной.

# Изменения промежуточного представления и ассемблера

| llvm-ir | asm |
| --- | --- |
| ; ModuleID = `test.c' | foo: |
| @GLOB = constant i32 7 | movl $GLOB+8, %ecx |
| define i32 @foo() { | movq %rcx, GLOB |
| %BADD = add i32* @GLOB, i32 8 | ... |
| store i32* %BADD, i32* @GLOB | .data: |
| ... | GLOB: .long 7 |
| } | .size GLOB, 4 |

# Производительность

| | размер | libc_bench | Ida Pro фун. |
|---|---|---|---|
| оригинал | 1 | 1 | 61/71 |
| все | 56 | 68 | 6/229 |
| 1 | 12.4 | 23 | 6/71 |
| 2 | 1.08 | 2.2 | 14/71 |
| 3 | 1 | ~1 | 57/71 |
| вставка "мертвого" кода | 1.06 | ~1 | 8/71 |

Изменение производительности измеряется отношением к производительности оригинальной программы.

# Выводы

- ИСП Обфускатор предоставляет большой набор полезных преобразований.
- Может быть расширен преобразованиями, ориентированными на конкретную задачу.
- Успешно показано на примере затруднения идентификации используемых компонентов с открытым исходным кодом.

# Спасибо за внимание!