Обфусцирующий компилятор

Курмангалеев Ш. Ф. [1] kursh@ispras.ru
Нурмухаметов А. Р. [1] oleshka@ispras.ru
Харченко Н. А. [1] [2] kharchenko@ispras.ru

[1]ИСП РАН, Москва [2]МФТИ (ГУ), Москва

2 декабря 2015 г.

Содержание:

ИСП РАН

Обфускация

ИСП РАН

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

Цели обфускации

ИСП РАН

- Затруднить обнаружение функциональности.
- Препятствовать обратной разработке.

Применение обфускации

ИСП РАН

- запутывание вирусов;
- сокрытие закладок в коде;
- расстановка водяных знаков на версиях программ для разных клиентов;
- затруднение идентификации используемых компонентов с открытым исходным кодом;
- усложнение идентификации автора кода;
- затруднение генерации эксплойтов на основе анализа патчей, закрывающих уязвимость;
- предотвращение эксплуатации известной уязвимости.

Подход к реализации

ИСП РАН

Для реализации преобразований выбрана компиляторная инфраструктура LLVM:

- открытый исходный код;
- модульная и расширяемая инфраструктура;
- поддержка нескольких компиляторов переднего плана (C, C++, Objective-C);
- поддержка множества целевых архитектур (ARM, Alpha, Intel x86, Microblaze, MIPS, PowerPC, SPARC).

Все запутывающие преобразования реализованы как отдельные компиляторные проходы над промежуточным представлением LLVM на машинно-независимом уровне.

Поддерживаемые преобразования

ИСП РАН

- Перемещение локальных переменных в глобальную область видимости.
- Приведение графа потока управления к плоскому виду.
- 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.

Приведение CFG к плоскому виду

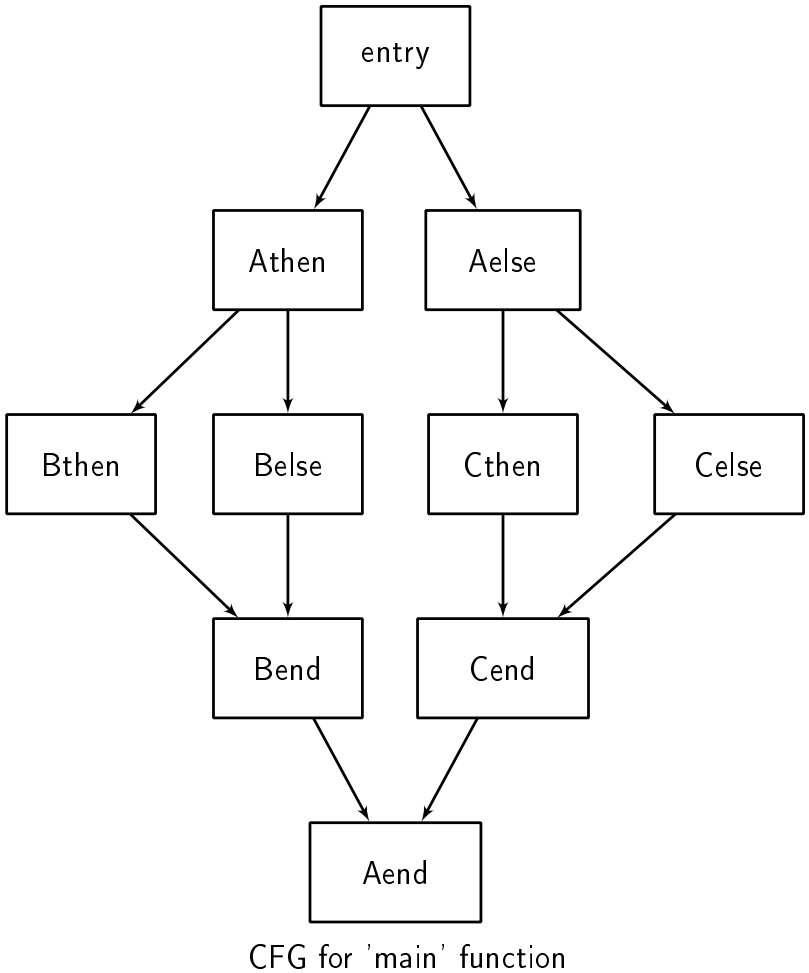

Рис. : Оригинальный CFG.

Приведение CFG к плоскому виду

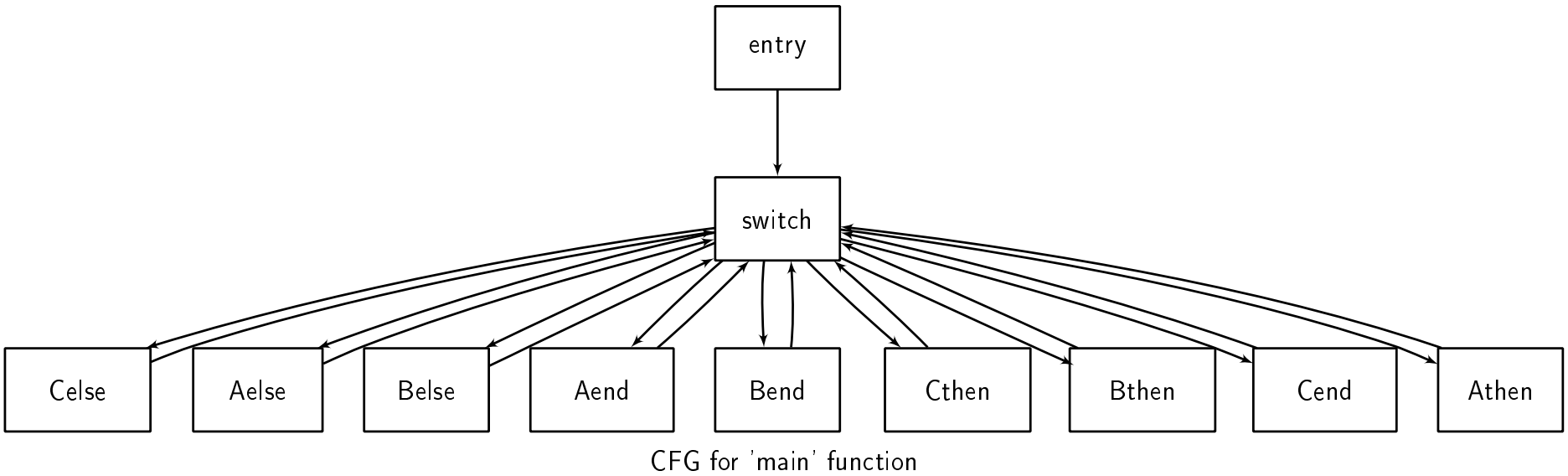

Рис. : CFG преобразованный к плоскому виду.

Приведение CFG к плоскому виду

ИСП РАН

- Переменная диспетчера сцеплена с живыми переменными блоков.
- Значение переменной диспетчеризации скрыто.
- Задача восстановления порядка следования базовых блоков является NP-трудной задачей.

Усложнение динамического анализа

ИСП РАН

Реализована схема запутывания, увеличивающая количество путей для увеличения сложности программы. Для этого:

- вставляются предикаты, зависящие от живых переменных и реализующие различные пути исполнения;
- часть кода после предиката клонируется и добавляется в качестве одной из возможных веток условного перехода;
- клонированный код диверсифицируется.

Используемые предикаты

ИСП РАН

Используются предикаты похожие на те, которые чаще всего встречаются в больших проектах:

- x != 0;
- x != y;
- x == y;
- x != f().

На проекте Firefox вместе составляют около 51 %.

Диверсификация CFG

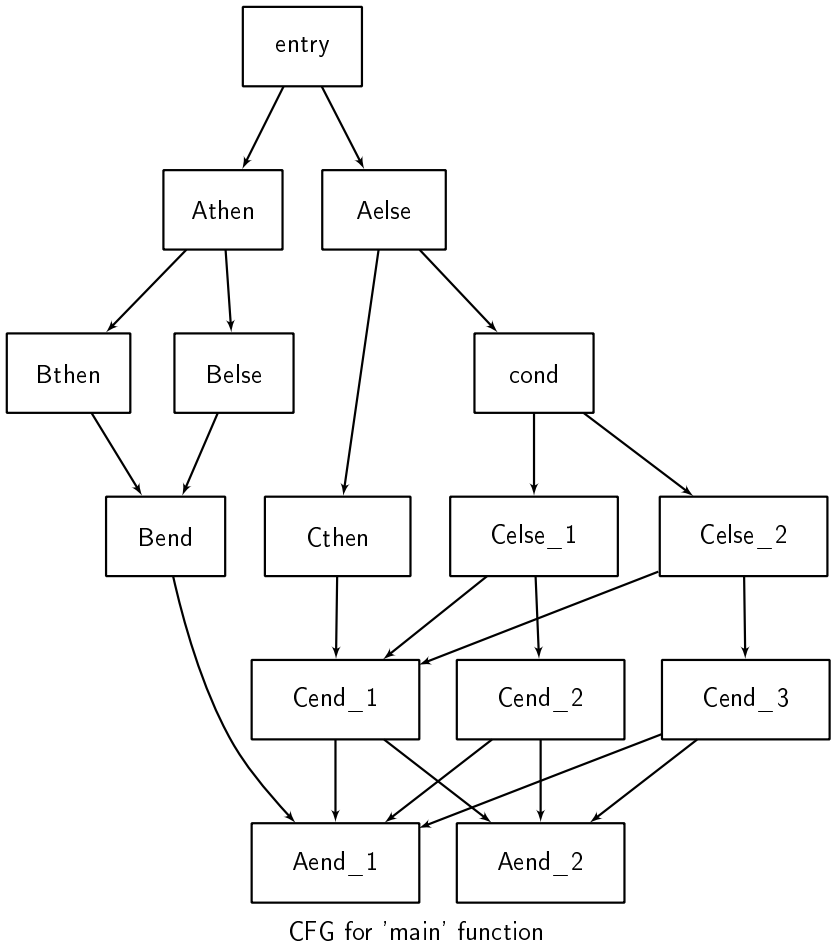

Рис. : CFG после преобразования

Дальнейшее направление работ

ИСП РАН

1. Доработка имеющегося преобразования по диверсификации CFG:
   - реализовать предикаты, зависящие от входных данных;
   - улучшить механизм диверсификации кода;
   - модифицировать алгоритм диверсификации таким образом, чтобы один конкретный путь был корректен только для некоторого подмножества входных данных.
2. Разработка метода запутывания, усложняющего анализ помеченных данных.

Противодействие эксплуатации уязвимостей

ИСП РАН

Предположим:

1. имеется программа с эксплуатируемой уязвимостью, например — с переполнением буфера на стеке;
2. злоумышленник, имея в своем распоряжении бинарную версию этого приложения, создает каким–либо образом эксплойт.

Задача — воспрепятствовать или усложнить эксплуатацию этого эксплойта у других пользователей данного приложения.

Противодействие эксплуатации уязвимостей

ИСП РАН

Предлагается генерировать диверсифицированную популяцию исполняемых файлов компилируемого приложения с помощью следующих преобразований:

1. перестановка функций;
2. добавление локальных переменных;
3. перемешивание переменных на стеке.

Оценка защищенности на примере переполнения буфера

ИСП РАН

Для адаптации эксплойта под другую копию приложения нужно подобрать константу, характеризующую масштаб изменений между кадрами стека.

1. Эта константа линейно зависит от размера кадра стека незащищённого приложения и количества добавленных локальных переменных.
2. Константа подбирается простым перебором, при этом на каждой итерации перебора требуется запуск целевого приложения.
3. В лучшем (для атакующего) случае потребуется около десятка попыток.

Возвратно–ориентированное программирование (ROP)

ИСП РАН

Возвратно–ориентированное программирование (англ. Return oriented programming) — метод эксплуатации уязвимостей при котором:

- атакующий получает контроль над стеком вызовов и выполняет тщательно подобранные последовательности инструкций, называемые «гаджетами»;
- гаджет заканчивается инструкцией возврата и располагается в коде программы или разделяемой библиотеки;
- связанные в цепочку инструкциями возврата, гаджеты позволяют атакующему выполнить произвольные операции.

Оценка защищенности на примере ROP–атаки

ИСП РАН

- Возможно построение цепочки ROP–гаджетов для одной копии приложения.
- Для другой копии этот эксплойт работать не будет из-за другого расположения функций.
- Подбор адресов — затратная по времени задача.
- Каждая неудачная попытка приводит к падению приложения.

Оценка производительности

ИСП РАН

Запутывающее преобразование	Увеличение размера	Падение производительности
Перемещение локальных переменных в глобальную область видимости	1.26	0.91
Размножение тел функций	2.17	0.96
Добавление и перемешивание переменных на стеке	1.00	1.00
Перестановка функций	1.00	1.00
Переплетение нескольких функций в одну	1.06	1.01
Формирование непрозрачных предикатов	1.00	1.02
Разбиение констант	1.04	1.06
Создание несводимых участков в графе потока управления	1.17	1.27
Вставка в код фиктивных циклов	1.17	1.32
Приведение графа потока управления к плоскому виду	4.58	6.99
Сокрытие вызовов функций	2.73	11.84
Диверсификация CFG	1.35	91.44
Диверсификация CFG + Приведение CFG к плоскому виду	4.83	96.31
Диверсификация CFG + Сокрытие вызовов функций + Приведение CFG к плоскому виду	8.35	1768.60

Таблица : Производительность запутывающих преобразований

Спасибо за внимание!

ИСП РАН

Вопросы?