Анализ и генерация байт-кода языка Python

Егор Шитов, 16.Б09-мм

Научный руководитель
проф. каф. СП, д.ф.-м.н., проф. А.Н. Терехов

Консультант
ст. преп. каф. СП М.В. Баклановский

Введение

1. Популярность языков программирования с промежуточным представлением.
2. Обфускация на уровне промежуточного представления.
3. Большинство исследований в области Java.
4. Важность незначительного увеличения времени работы программы для обфускации.

Постановка задачи

Цель работы — исследовать влияние подходов обфускации на время работы программы в контексте байт-кода виртуальной машины Python. Задачи, поставленные для решения в рамках данной работы:

- сделать обзор виртуальной машины языка Python
- разработать инструментарий для оперирования линейными блоками байт-кода Python
- сконструировать подход для корректного измерения времени работы программы
- провести эксперименты о влиянии подходов обфускации на время работы программы

Обфускация

- удаление информации
  - удаление отладочной информации
  - удаление имени или переименование идентификаторов
- запутывание потока выполнения
  - подстановка функций
  - изменение порядка выполнения инструкций и функций
  - вставка дополнительного кода
- запутывание потока данных
  - изменение области видимости идентификатора

Виртуальная машина Python

Стандартная виртуальная машина Python стековая

- ограниченные инструкции
- CodeObject
- оптимизации

Измерительные инструменты

- микробенчмаркинг
- профайлеры
- функция time

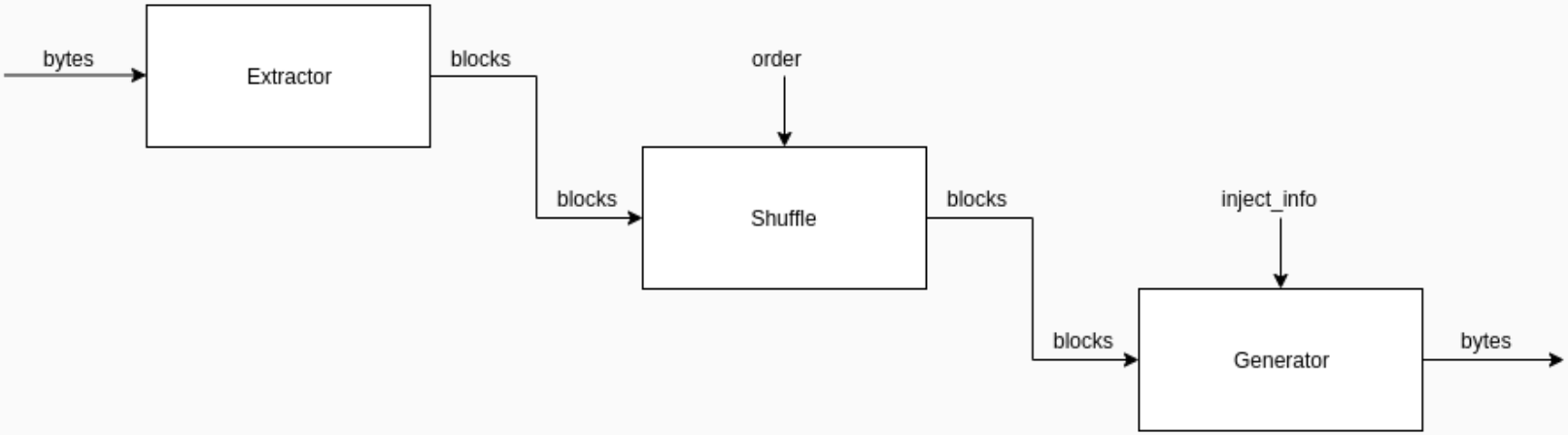
Оперирование линейными блоками

Задачи инструментария оперирования линейными блоками:

- фиксирование размера линейного блока
- изменение порядка линейных блоков
- вставка произвольного кода

Архитектура

В ходе работы выкристаллизовалась архитектура, похожая на pipes-and-filters



Программы для тестирования

Требования:

- отсутствие сторонних библиотек
- различный размер программ

Размер входных данных был подобран так, чтобы программы работали несколько секунд

Особенности проведения измерений

Особенности:

- CPython 3.5
- минимизация сторонних процессов
- фиксирование частоты (1500 МГц)
- прогрев
- исключение времени запуска процессов в ОС, виртуальной машины Python, IO операций

Окружение проведения измерений

Все измерения проводились на ноутбуке Lenovo Ideapad 720s 14

- процессор — Intel(R) Core(TM) i7-8550U
- ОС Linux — Elementary OS 0.4.1 Loki, kernel 4.15.0-88-generic
- ОС Windows — Windows 10
- RAM — 16 Гб

Фиксирование размера линейного блока

Размер	пуз.	быстр.	extr.
16	4.36 (6.18)	3.77 (5.42)	3.92 (6.22)
17	4.35 (6.18)	3.76 (5.56)	3.90 (6.07)
18	4.35 (6.17)	3.77 (5.54)	3.96 (6.06)
19	4.34 (6.23)	3.77 (5.52)	3.96 (6.11)
20	4.34 (6.18)	3.73 (5.32)	3.91 (6.00)
21	4.35 (6.30)	3.73 (5.34)	3.90 (6.06)
22	4.32 (6.11)	3.73 (5.34)	3.91 (6.11)
23	4.32 (6.11)	3.72 (5.28)	3.90 (6.08)
24	4.32 (6.11)	3.71 (5.29)	3.90 (6.02)
25	4.33 (6.23)	3.71 (5.37)	3.89 (5.99)
26	4.34 (6.23)	3.71 (5.28)	3.90 (6.02)
27	4.33 (6.23)	3.71 (5.27)	3.89 (6.05)
28	4.29 (6.01)	3.72 (5.27)	3.89 (5.95)
29	4.28 (6.01)	3.72 (5.27)	3.87 (5.92)
30	4.29 (6.01)	3.72 (5.27)	3.88 (5.97)
31	4.29 (6.01)	3.71 (5.28)	3.88 (5.96)

Tabelle 1: Фиксирование размера блока, Linux (Windows)

Изменение порядка линейных блоков

Прог., группировка	Linux	Windows
Пузырек, станд.	4.27	5.98
Пузырек, груп. 1	4.27	6.09
Пузырек, груп. 2	4.28	5.97
Быстрая, станд.	3.69	5.21
Быстрая, груп. 1	3.69	5.26
Быстрая, груп. 2	3.69	5.26
Extr., станд.	3.84	5.98
Extr., груп. 1	3.85	5.95
Extr., груп. 2	3.84	5.95

Tabelle 2: Перестановка линейных блоков

Вынесение идентификаторов и подстановка функций

Подход	Linux	Windows
Без изменений	3.69	5.21
Вынесение лок. перем.	4.35	6.00
Вынесение лок. перем. и подстановка	4.30	5.92
Подстановка функций	3.65	5.17

Tabelle 3: Вынесение идентификаторов и подстановка функций, быстрая сортировка, сек

Подход	Linux	Windows
Без изменений	3.84	5.98
Вынесение лок. перем.	6.34	8.73

Tabelle 4: Вынесение идентификаторов, модуль Extractor, сек

Вывод

1. Выполнить подстановку функций.
2. Зафиксировать небольшой размер линейных блоков.
3. Выполнить перестановку линейных блоков.

Итоги

В ходе данной работы исследовано влияние подходов обфускации на время работы программы в контексте байт-кода виртуальной машины Python, в частности:

- сделан обзор виртуальной машины языка Python
- разработан инструментарий для оперирования линейными блоками байт-кода Python
- разработан подход для измерения времени работы программы
- проведены эксперименты о влиянии подходов обфускации на время работы программы