Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 22.М07-мм

Статистический анализ машинных инструкций

Афонина Ольга Андреевна

Отчёт по учебной практике

Научный руководитель:
ст. преп. каф. СП, Я. А. Кириленко

Санкт-Петербург
2023

Оглавление

Введение

В настоящее время процессорные архитектуры активно развиваются: появляются новые поколения, расширения. Особый интерес представляет архитектура RISC-V[1], являющаяся открытой и свободной, то есть она может использоваться кем угодно как для открытых, так и коммерческих реализаций. Сообщество активно занимается его развитием, например, регулярно добавляются новые расширения [13] такие, как векторные или расширения битовых манипуляций.

При добавлении новых инструкций перед разработчиками встаёт ряд проблем. При разработке процессоров необходимо как можно более оптимально аппаратно реализовать инструкции, для чего нужны содержащие их тесты производительности. Также нужна информация о том, какие инструкции следует оптимизировать в первую очередь. Разработчикам компиляторов необходимо встраивать оптимизации, использующие новые возможности процессоров. Для тестирования этих оптимизаций могут подойти утилиты, в которых подобные инструкции встраиваются компилятором на другой платформе. Кроме того, при разработке библиотек информация о добавленных вручную инструкций подобного расширения другой платформы подскажет, где такие же оптимизации нужны для новых инструкций.

В решении упомянутых проблем может помочь статистика использования машинных инструкций. Её можно собирать динамически, что отобразит, какие инструкции реально используются. Так, например, можно управлять политикой вытеснения кэша [11] или сокращения циклов микроопераций [8]. Однако такой подход требует создания окружения, подбора сценариев использования, при этом не гарантирует покрытия всей функциональности и является довольно трудоёмким. Статический же анализ позволит быстро собирать статистику для большого объема данных, что открывает возможности для поиска интересных особенностей и формулировки гипотез на их основе. Например, позволит быстро делать выборку утилит для дальнейшего динамического

[1] https://riscv.org/

анализа.

Таким образом, актуальная статистика использования инструкций может быть применена для решения многих проблем. Однако нет собранного набора данных, который позволил бы оценивать и сравнивать использование инструкций. Отсюда видно, что нужен инструмент для сбора этой статистики, а также в обновляемом наборе собираемых им данных, находящихся в открытом доступе.

1. Постановка задачи

Целью работы является создание инструмента для статистического анализа машинных инструкций и создания набора собранных им данных. Для её выполнения на осенний семестр была поставлена следующая задача:

- провести обзор методов применения статистического анализа инструкций.

2. Обзор

В данной главе рассмотрены работы, использующие в своих исследованиях статистические данные о машинных инструкциях, которые были собраны с помощью статического анализа. Цель обзора — выяснить, какую статистику в них используют и как её собирают.

2.1. Выявление и классификация вредоносного кода

Opcode Frequency Based Malware Detection Using Hybrid Classifiers [2]. В данной работе частоты операций используются в качестве признаков для обучения модели, определяющей, является ли исполняемый файл вредоносным. Подсчет интересующих для анализа инструкций выполнялся для 3000 переносимых исполняемых файлов с помощью инструмента `objdump`[2], используемого для их дизассемблирования. Полученный набор данных о частоте инструкций использовался для обучения моделей на основе метода случайного леса, AdaBoost, XGBoost и Voting Classifier. Последний классификатор показал лучшие результаты с долей правильных ответов 95.77, точностью 0.95; полнотой 0.98 и F-мерой 0.96. Стоит отметить, что автор исследовал только 32-битовые файлы размером не более 35 KB.

Control flow-based opcode behavior analysis for Malware detection [4]. В работе исследуются методы обнаружения вредоносных программ, основанных на поведении машинных инструкций. Авторы предлагают метод для извлечения последовательностей инструкций на основе дерева управления. Инструкции извлекаются из исполняемых файлов Windows PE с помощью коммерческого инструмента IDA Pro[3]. Декомпилированные исполняемые файлы анализируются не только как текстовые: для них строятся всевозможные пути исполнения программы, которые конкатенируются для получения потока операций. Для полученного из дерева потока инструкций и их текстовой последователь-

[2] https://sourceware.org/binutils/docs/binutils/objdump.html
[3] https://hex-rays.com/ida-pro/

ности в декомпилированном файле были собраны статистики частот 3-грамм инструкций. Поскольку последовательностей трёх подряд идущих инструкций слишком много для использования их в качестве признаков для классификаторов, было взято 400 последовательностей с наибольшим показателем прироста информации. Полученные классификаторами (k-ближайших соседей, дерево решений и метод опорных векторов) результаты оказались лучше для признаков, полученных на основе дерева управления: доля правильных ответов выше, а доли ложных положительных и ложных отрицательных срабатываний ниже.

Malware Classification using Instruction Frequencies [6].
Авторы используют распределение частот использования инструкций для быстрой классификации и обнаружения вредоносных программ с низкими вычислительными затратами. Для этого в работе подсчитываются частоты шестнадцатеричных кодов инструкций в исполняемых файлах. Для бинарных файлов ELF строятся последовательности инструкций, отсортированные по частоте использования, и подсчитывается число пересечений при соединении рёбрами одних и тех же инструкций в полученных последовательностях. В зависимости от диапазона, в котором лежит число пересечений, можно предположить, является ли программа вредоносной и принадлежит ли она рассматриваемому семейству вирусов. Полученные в исследовании результаты весьма позитивны, однако они не были проверены на большом наборе данных и не были представлены метрики надёжности предложенной классификации.

Opcodes Histogram for Classifying Metamorphic Portable Executables Malware [12].
Авторы данной работы показывают, как гистограммы частот инструкций могут помочь в классификации семейства метаморфных вирусов. Набор таких гистограмм был собран для семейства вирусов NGVCK и построена для него усредненная гистограмма. Частоты получали, подсчитывая операционные коды инструкций в дизассемблированных ис-

полняемых файлах MS-DOS (PE, COFF). Классификация основана на вычислении расстояния Минковского для гистограмм. Оценка предложенного метода проводилась для тестовых данных всего из 100 файлов, также было рассмотрено всего одно семейство вирусов, из-за чего нельзя утверждать про его эффективность.

Opcode Graph Similarity and Metamorphic Detection [14].
В работе предложили оценку сходства исполняемых файлов на основе графов инструкций для определения вредоносности файла. Мера сходства определяется для матричного представления графов инструкций: a_{ij} — частота, с которой пара инструкций (i, j) присутствует в последовательности дизассемблированных инструкций. Затем строится граф, где вес ребра между инструкциями — вероятность того, что они следуют друг за другом. Показатель сходства считается для матриц следующим образом: $score(A, B) = \frac{1}{N^2}(\sum_{i,j=0}^{N-1} |a_{ij} - b_{ij}|)^2$. Данный подход показал свою эффективность — при некоторых правдоподобных сценариях он превзошёл метод, основанный на скрытой марковской модели.

Opcodes as predictor for malware [3].
В работе сравниваются распределения инструкций для вредоносного и невредоносного ПО, которые также исследуются в качестве средства обнаружения и классификации вирусов. Данные о частоте инструкций были собраны с использованием IDA Pro и расширения InstructionCounter для выборки вредоносных и невредоносных исполняемых файлов. Для проанализированных файлов оценили взаимосвязь нормализованной статистики самых частых и самых редких инструкций и классов вредоносного ПО. Мера V Крамера показала, что связь с часто встречающимися инструкциями слабая (мера варьируется от 0.5 до 0.15), а наиболее редкими она умеренная — от 0.12 до 0.63. Данные были собраны для небольшого числа файлов, что позволяет только выдвинуть гипотезу.

Detecting unknown malicious code by applying classification techniques on OpCode patterns [5].

В данной работе шаблоны N-грамм инструкций используются в качестве признаков для процесса классификации. Авторы проводят эксперименты для выявления представления N-грамм, размеров N-грамм, способов их выбора для использования в качестве признаков и классификатора, которые бы показали себя лучше всего. Сбор данных о частоте инструкций в исполняемых файлах был выполнен с помощью инструмента IDA Pro. В результате поставленных экспериментов было выявлено, что наилучший результат получается при представлении TF (normalized term frequency) — частота слова в файле делится на частоту наиболее популярного в нём. Поскольку размеры словарей для различных N-грамм могут быть слишком велики для использования в качестве признаков классификации, авторы статьи выяснили, что лучшим из рассмотренных ими способов для отбора признаков является DF (document frequency), то есть берутся слова, встречающиеся в наибольшем числе файлов.

2.2. Другие применения

An Analysis of x86-64 Instruction Set for Optimization of System Softwares [1].

В работе проанализирован набор инструкций x86-64 в OS Windows 7 для 32- и 64-битовых приложений и собрана для них разнообразная статистика. Статистические данные генерируются на основе дизассемблированных с помощью инструмента `udis86`[4] инструкций. Для инструкций собиралась информация о её типе, аргументах, длине, режиме адресации. На основе этих данных авторы составили статистику о частоте инструкций, использовании регистров, количестве использования различных типов адресации. Авторы считают, что эта информация полезна при разработке промежуточных языков (например, байт-код Java)

[4] https://github.com/vmt/udis86

для минимального потребления памяти.

Instruction Set Usage Analysis for Application-Specific Systems Design [10]. Исследуют влияние увеличения размера набора команд процессора на аппаратное обеспечение, насколько эффективно используются аппаратные ресурсы четырех процессорных архитектур (Intel x86, Intel x86-64, MIPS64 и PowerPC), измеряя использование набора команд группой приложений тестов производительности SPEC CPU 2006 [7]. Приложения были скомпилированы с помощью GCC и PCC с опцией генерации ассемблерных файлов вместо бинарных на Ubuntu 10.04 LTS. Результаты исследования показали, что среднее использование ресурсов процессоров составляет от 5 до 20 процентов.

x86-64 Instruction Usage among C/C++ Applications [16].
Авторы данной работы исследуют относительную важность инструкций на основе частоты, с которой они встречаются в пакетах, и популярности этих пакетов. Исследование проводилось для дистрибутива Ubuntu 16.04 и архитектуры x86-64. С использованием утилит `readelf`[5] и `objdump` для каждого пакета рекурсивно считается частота встречающихся инструкций в бинарных файлах и в вызываемых библиотеках. Также для инструкций собирается информация об их размере, префиксах, методу адресации. Авторы предлагают использование собранных ими данных для измерения полноты бинарных инструментов, поскольку для многих приложений нет необходимости в поддержке всех инструкций; также они утверждают, что для проверки бинарных инструментов достаточно 55 пакетов для покрытия всех инструкций. Следует отметить, что в открытом доступе есть инструмент визуализации собранных в работе данных, а также разработанный для анализа инструмент.

SHRINK: Reducing the ISA Complexity Via Instruction Recycling [15]. Выполняют статический анализ образов дисков виртуальных машин.

[5] https://man7.org/linux/man-pages/man1/readelf.1.html

Реализовали обходчик инструкций, который просматривает каждый образ диска и подсчитывает, сколько раз каждый UIS (Unique Instruction Signature) появляется в любом исполняемом файле. В качестве библиотек используется два инструмента дизассемблирования: конвертер объектных файлов objconv[6]) и библиотека дизассемблера виртуальной машины Bochs[7]. Помимо статического анализа выполняют динамический анализ на модифицированной виртуальной машине (собирающей гистограммы исполненных инструкций), поскольку во время выполнения могут быть сгенерированы и выполнены новые инструкции. Анализ проводили для 32-битовых машин, разных версий Ubuntu и Windows, вышедших с 1995 по 2012 год. Было выявлено, какие инструкции не использовались или перестали использоваться со временем. На основе собранных данных разработали инструмент удаления старых инструкций без ущерба для обратной совместимости. В открытом доступе данный инструмент найти не удалось

Opcode statistics for detecting compiler settings [9].
Исследовательский проект в рамках магистерской, для изучения корреляции между частотой машинных инструкций (и пар последовательных инструкций) и версией GCC, а также флагами компилятора. Данные об инструкциях из скомпилированных на Ubuntu 16.04.3 LTS файлов извлекались с использованием инструмента `objdump`. На основе оценок V-Крамера и Z-score было выявлено, что с версией связь между частотой инструкций и версией компилятора совсем слабая, то есть для использования в классификаторах не подходит, с флагами же умеренная. Однако исследование проводилось только для 20 приложений (достаточно популярных и разнообразных по словам автора), что не позволило проверить гипотезу с использованием машинного обучения из-за отсутствия готового набора данных.

[6] https://github.com/gitGNU/objconv
[7] https://bochs.sourceforge.io/

2.3. Вывод

Проведенный обзор показывает, что статистика использования инструкций может применяться для разных целей, поэтому целесообразно собрать большой открытый набор данных для разных платформ. Поскольку в работах анализировалась частота как единичных инструкций, так и их N-грамм (последовательностей N подряд идущих инструкций) не только в текстовом представлении потока инструкций, но и в графовом (с учётом условных и безусловных переходов), есть смысл хранить в собранном наборе статистических данных исходные результаты дизассемблирования. Наиболее популярными инструментами для извлечения инструкций оказались IDA Pro и `objdump`, а так как первый является коммерческим, для прототипа разрабатываемого инструмента сбора статистики следует обратить внимание на `objdump`.

Заключение

В рамках учебной (ознакомительной) практики осеннего семестра была выполнена следующая задача:

- проведен обзор методов применения статистического анализа машинных инструкций и выявлены подходящие для прототипирования решения.

В следующем семестре планируется разработать прототип инструмента, воспроизведя результаты, полученные в работе [16], и расширить собранные данные на другие дистрибутивы и архитектуры.

Список литературы

[1] An Analysis of x86-64 Instruction Set for Optimization of System Softwares / Amr H. Ibrahim, M. B. Abdelhalim, Hanady Hussein, Ahmed Fahmy. — 2011. — 10. — Vol. 1, no. 4. — P. 152–162.

[2] Arun Manoharan Kollara. Opcode Frequency Based Malware Detection Using Hybrid Classifiers. — 2020.

[3] Bilar Daniel. Opcodes as predictor for malware // International journal of electronic security and digital forensics. — 2007. — Vol. 1, no. 2. — P. 156–168.

[4] Control flow-based opcode behavior analysis for Malware detection / Yuxin Ding, Wei Dai, Shengli Yan, Yumei Zhang // Computers Security. — 2014. — Vol. 44. — P. 65–74. — URL: https://www.sciencedirect.com/science/article/pii/S0167404814000558.

[5] Detecting unknown malicious code by applying classification techniques on opcode patterns / Asaf Shabtai, Robert Moskovitch, Clint Feher et al. // Security Informatics. — 2012. — Vol. 1, no. 1. — P. 1–22.

[6] Han Kyoung Soo, Kang Boojoong, Im Eul Gyu. Malware Classification Using Instruction Frequencies // Proceedings of the 2011 ACM Symposium on Research in Applied Computation. — RACS '11. — New York, NY, USA : Association for Computing Machinery, 2011. — P. 298–300. — URL: https://doi.org/10.1145/2103380.2103441.

[7] Henning John L. SPEC CPU2006 benchmark descriptions // ACM SIGARCH Computer Architecture News. — 2006. — Vol. 34, no. 4. — P. 1–17.

[8] Huang Ing-Jer, Peng Tzu-Chin. Analysis of /spl times/86 instruction set usage for DOS/Windows applications and its implication on superscalar design // Proceedings International Conference on Computer

Design. VLSI in Computers and Processors (Cat. No.98CB36273). — 1998. — P. 566–573.

[9] Kenneth van Rijsbergen. Opcode statistics for detecting compiler settings. — 2018.

[10] Mutigwe Charles, Kinyua Johnson, Aghdasi Farhad. Instruction set usage analysis for application-specific systems design // Int'l Journal of Information Technology and Computer Science. — 2013. — Vol. 7, no. 2.

[11] Petoumenos Pavlos, Keramidas Georgios, Kaxiras Stefanos. Instruction-based reuse-distance prediction for effective cache management // 2009 International Symposium on Systems, Architectures, Modeling, and Simulation. — 2009. — P. 49–58.

[12] Rad Babak Bashari, Masrom Maslin, Ibrahim Suahimi. Opcodes histogram for classifying metamorphic portable executables malware // 2012 International Conference on E-Learning and E-Technologies in Education (ICEEE). — 2012. — P. 209–213.

[13] Recently Ratified Extensions. — https://wiki.riscv.org/display/HOME/Recently+Ratified+Extensions.

[14] Runwal Neha, Low Richard, Stamp Mark. OpCode graph similarity and metamorphic detection // Journal in Computer Virology. — 2012. — 05. — Vol. 8.

[15] SHRINK: Reducing the ISA Complexity via Instruction Recycling / Bruno Cardoso Lopes, Rafael Auler, Luiz Ramos et al. // SIGARCH Comput. Archit. News. — 2015. — jun. — Vol. 43, no. 3S. — P. 311–322. — URL: https://doi.org/10.1145/2872887.2750391.

[16] X86-64 Instruction Usage among C/C++ Applications / Amogh Akshintala, Bhushan Jain, Chia-Che Tsai et al. // Proceedings of the

12th ACM International Conference on Systems and Storage. — SYSTOR '19. — New York, NY, USA : Association for Computing Machinery, 2019. — P. 68–79. — URL: https://doi.org/10.1145/3319647.3325833.