

Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 22.Б15-мм

Замена встроенных типов Ghidra стандартными типами C

Ефремов Алексей Александрович

Отчёт по производственной практике
в форме «Производственное задание»

Научный руководитель:
старший преподаватель кафедры системного программирования Я. А. Кириленко

Консультант:
эксперт по разработке ПО, ООО «Ядро Центр Программных Разработок»
А. Ю. Стариковский

Санкт-Петербург
2025

Оглавление

Введение	3
1. Постановка задачи	4
2. Обзор	5
2.1. Скрипты	5
2.2. Расширения	5
2.3. Модификация исходного кода	6
2.4. Выбор способа модификации	7
3. Реализация	9
4. Тестирование	10
Заключение	11
Список литературы	12

Введение

Программное обеспечение является неотъемлемой частью современного мира, оказывая влияние на все аспекты человеческой деятельности, от решения повседневных задач до функционирования сложных технологических систем. Значительная часть программного обеспечения распространяется в скомпилированном виде, представляя собой машинный код, готовый к исполнению процессором. Однако, скомпилированный код, будучи оптимизированным для исполнения машиной, зачастую обладает сложной внутренней структурой, что может затруднять его понимание и анализ. В различных областях, таких как разработка программного обеспечения, оптимизация производительности [3, 4] и поиск уязвимостей [1, 2], возникает потребность в изучении структуры и логики работы скомпилированного кода.

Для анализа скомпилированного кода широко применяются инструменты статического анализа, которые позволяют более удобно работать с машинным кодом. Такие инструменты предоставляют информацию о функциях, алгоритмах и структурах данных, используемых в программе. Одним из наиболее известных инструментов является Ghidra [6] — декомпилятор с открытым исходным кодом, созданный для анализа программ и предоставляющий широкий набор функций для их исследования.

Хотя Ghidra поддерживает стандартные типы данных языка C, она преимущественно использует собственные встроенные типы для представления данных. Для повышения удобства работы и обеспечения согласованности с исходным кодом на языке C было необходимо минимизировать использование типов, не соответствующих стандартным типам языка C.

В рамках данной работы встроенные типы данных Ghidra были адаптированы в соответствии со стандартными типами языка C. Для этого были изучены архитектура Ghidra, механизмы её расширения и модификации, а также реализованы требуемые изменения.

1. Постановка задачи

Целью работы является модификация Ghidra для минимизации использования типов, не соответствующих стандартным типам языка C.

1. выполнить обзор механизмов расширения функциональности Ghidra;
2. реализовать модификацию Ghidra, направленную на увеличение использования стандартных типов данных языка C;
3. провести тестирование инструмента после модификации.

2. Обзор

Ghidra, являясь мощным инструментом для анализа бинарных файлов, предлагает разработчикам ряд методов для модификации и расширения своей функциональности. Эти методы обеспечивают адаптацию инструмента к специфическим задачам, позволяя добавлять новые возможности или изменять существующие. К основным подходам модификации Ghidra относятся: применение скриптов, разработка расширений, а также прямое изменение исходного кода.

2.1. Скрипты

В Ghidra реализована поддержка создания скриптов на языках программирования Java и Python. Посредством Ghidra API скрипты могут взаимодействовать с различными компонентами и сущностями инструмента. Это позволяет эффективно автоматизировать выполнение рутинных задач, значительно сокращая время, затрачиваемое на их выполнение.

Разработка скриптов в Ghidra облегчается за счёт подробной документации архитектуры API. Однако, несмотря на широкие возможности Ghidra API, его функциональность не всегда позволяет удовлетворить все потребности пользователя, особенно в случаях, требующих нестандартных решений или углубленного взаимодействия с внутренними механизмами инструмента

2.2. Расширения

В отличие от скриптов, расширения в Ghidra представляют собой более сложные и глубоко интегрированные решения, реализуемые в виде проектов на языке Java. Основное отличие расширений заключается в их тесной интеграции с внутренними механизмами Ghidra, что позволяет расширить функциональность инструмента на фундаментальном уровне.

Существует пять основных типов расширений, которые могут быть созданы в Ghidra:

- **Analyzer** — расширения, предназначенные для автоматического анализа кода и данных;
- **Plugin** — расширения, добавляющие новые функции в пользовательский интерфейс, такие как дополнительные панели или инструменты;
- **Loader** — расширения, отвечающие за загрузку и обработку новых типов файлов или форматов;
- **FileSystem** — расширения, обеспечивающие доступ к данным в различных форматах хранения (образы, архивы и т.д.);
- **Exporter** — расширения, позволяющие экспортировать данные в различные форматы.

Несмотря на то, что перечисленные типы расширений являются наиболее распространёнными, архитектура Ghidra допускает создание дополнительных видов расширений. Для этого используется специальный класс `ExtensionPoint` [5], а точнее его потомки, которые позволяют разработчикам реализовывать собственные типы расширений. Однако как отмечает команда разработчиков Ghidra, использование данного подхода не рекомендуется [7], так как он не является хорошо поддерживаемым. Разработчики советуют по возможности ограничиваться стандартными типами расширений, чтобы обеспечить совместимость и стабильность работы.

2.3. Модификация исходного кода

Открытость исходного кода Ghidra предоставляет возможность непосредственной модификации её исходного кода, представляя собой наиболее радикальный подход к кастомизации. Данный метод обеспечивает разработчикам большую гибкость, позволяя реализовывать

функциональность, не ограничиваясь возможностями, доступными через скрипты или расширения. Тем не менее этот подход требует глубокого понимания архитектуры инструмента и повышенной осторожности при внесении изменений, поскольку некорректные модификации исходного кода сложной системы могут привести к нарушению её стабильности и функциональности.

2.4. Выбор способа модификации

На начальном этапе работы рассматривалась возможность разработки скрипта, который проводил замену всех вхождений выбранных встроенных типов на стандартные типы языка C в функциях и данных. Однако данный подход был признан неэффективным из-за необходимости повторного запуска скрипта при образовании нового вхождения встроенного типа в процессе анализа бинарного файла.

В качестве альтернативы было предложено использование расширения типа Analyzer (анализатор). При создании анализатора требуется задать его дополнительный тип, определяющий события, на которые будет подписан анализатор (например, изменение или создание функции, создание новой переменной). Благодаря подписке на события, анализатору не требуется перезапуск, в отличие от скрипта. Однако ограничение на один тип для каждого анализатора потребует создания нескольких отдельных анализаторов, отдельно для обработки функций и отдельно для данных. Для минимизации дублируемого кода между двумя анализаторами можно выделить общий родительский класс, содержащий общую функциональность. Таким образом, разработка расширения для изменения типов изначально представлялась оптимальным решением.

Но при разработке расширения стало известно, что декомпилятор в Ghidra может самостоятельно выводить типы, не синхронизируя их с основной базой данных, где производится замена типов. Это приводило к отображению в окне декомпиляции встроенных типов, не соответствующих стандартным типам языка C. В результате анализа было принято

решение о необходимости внесения прямых изменений в исходный код Ghidra для обеспечения корректной работы системы.

3. Реализация

Для повышения соответствия встроенных типов Ghidra стандартным типам языка C, были модифицированы идентификаторы 31 встроенного типа. Изменения включают два основных вида преобразований:

- **Приведение размерности к битовой:** Размерность типа, ранее указанная в байтах, преобразована в биты. Также добавлен суффикс «_t» к типам, представляющим целые числа фиксированной размерности, для соответствия стандарту C99. Например, тип `int8` преобразован в `int64_t`. Тип `float2` (специфичный для Ghidra) преобразован в `float16`.
- **Расшифровка сокращений:** Сокращения вида `u<type>`, обозначающие беззнаковый тип данных, заменены на полную форму `unsigned <type>`. Например, тип `uchar` преобразован в `unsigned char`.

Второй вид преобразований обусловлен архитектурным ограничением Ghidra, которое не допускает наличия встроенных типов с идентичными именами. В то же время, реализация объединения типов представляет собой значительную техническую задачу, целесообразность и необходимость которой требуют тщательного анализа, учитывая основания для их первоначального разделения в рамках системы.

4. Тестирование

Тестирование внесённых изменений проводилось с использованием тестовой базы Ghidra. В исходной версии Ghidra без изменений 525 из 18250 тестов не проходили успешно. После внедрения изменений количество неуспешных тестов увеличилось до 1004. Таким образом, из-за внесённых изменений не прошли 479 дополнительных тестов. Анализ этих тестов показал, что функциональность инструмента не нарушена, а неудачи связаны с тем, что тесты ожидают использования старых названий встроенных типов. Для устранения проблемы требовалось обновление тестов с учётом новых имён типов.

Обновление тестовой базы включало следующие действия:

- **Замена имён в строковых литералах:** В 475 тестах произведена замена всех вхождений устаревших наименований встроенных типов в строковых литералах на новые.
- **Коррекция индексов:** В двух тестах скорректированы индексы, определяющие позицию переменных в списке, отсортированном по имени типа. Это изменение обусловлено изменением порядка сортировки после переименования типов.
- **Переименование вводимого типа:** В одном тесте имя вводимого типа изменено с `int64_t` на `new_int64_t` для избежания конфликта имён с изменёнными встроенными типами.
- **Обновление позиций курсора:** В одном тесте скорректированы позиции курсора в окне декомпиляции, поскольку изменение длины названий типов привело к сдвигу позиций кода.

Заключение

В ходе этой работы были выполнены все задачи и получены следующие результаты:

1. выполнен обзор механизмов расширения функциональности Ghidra;
2. произведено изменение встроенных типов данных Ghidra в соответствии со стандартными типами языка C;
3. проведено тестирование модифицированной версии Ghidra и обновлены наборы тестов с учётом изменённых базовых типов.

Версия Ghidra, разрабатываемая компанией «YADRO», является проектом с закрытым исходным кодом, в связи с чем доступ к исходному коду не предоставляется.

Список литературы

- [1] Binary-level Directed Fuzzing for Use-After-Free Vulnerabilities / Manh-Dung Nguyen, Sébastien Bardin, Richard Bonichon et al. // 23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020). — San Sebastian : USENIX Association, 2020. — . — P. 47–62. — URL: <https://www.usenix.org/conference/raid2020/presentation/nguyen> (дата обращения: 27 декабря 2024 г.).
- [2] The Convergence of Source Code and Binary Vulnerability Discovery—A Case Study / Alessandro Mantovani, Luca Compagna, Yan Shoshitaishvili, Davide Balzarotti // Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security. — 2022. — P. 602–615.
- [3] Cooper Keith, Dasgupta Anshuman, Kennedy Ken. Vizer: A system to vectorize intel x86 binaries // Intl. Symp. on Computer Architecture, Santa Fe, NM. — Vol. 64. — 2002.
- [4] Drucker Nir, Gueron Shay. Software Optimization of Rijndael for Modern x86-64 Platforms // ITNG 2022 19th International Conference on Information Technology-New Generations / Springer. — 2022. — P. 147–153.
- [5] ExtensionPoint. — Ghidra documentation. — URL: https://ghidra.re/ghidra_docs/api/ghidra/util/classfinder/ExtensionPoint.html (дата обращения: 27 декабря 2024 г.).
- [6] Ghidra. — Github. — URL: <https://github.com/NationalSecurityAgency/ghidra/> (дата обращения: 27 декабря 2024 г.).
- [7] Mike Bell: Extending Ghidra: from Script to Plugins and Beyond. — Открытое выступление. — URL: <https://theycyberwire.com/stories/90782540555d4f49a4b00700ae4619fb/>

[mike-bell-extending-ghidra-from-script-to-plugins-and-beyond](#)
(дата обращения: 27 декабря 2024 г.).