Автономная некоммерческая организация высшего образования
«Университет Иннополис»

АННОТАЦИЯ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
(БАКАЛАВРСКУЮ РАБОТУ)
ПО НАПРАВЛЕНИЮ ПОДГОТОВКИ
09.03.01 – «ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА»

НАПРАВЛЕННОСТЬ (ПРОФИЛЬ) ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ
«ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА»

Тема **Метаморфическое и комбинаторное тестирование в сфере компьютерной безопасности**

Выполнил **Абрамов Матвей Евгеньевич**

подпись

Иннополис, Innopolis, 2024

Оглавление

Аннотация

Стремительное развитие программного обеспечения и его растущая интеграция в различные аспекты нашей повседневной жизни привели к появлению проблем в обеспечении его безопасности. В результате слияния не ИТ и ИТ-индустрий увеличилось количество ошибок и уязвимостей в программном обеспечении, которые влекут за собой значительные риски для отдельных людей, организаций и общества в целом. Несмотря на постоянные усилия по улучшению кибербезопасности, вездесущность киберугроз и отсутствие единого надежного метода обеспечения корректности программного обеспечения по-прежнему остаются актуальными проблемами.

Данный дипломный проект направлен на решение критической потребности в повышении кибербезопасности путем изучения потенциала метаморфического тестирования и комбинаторного тестирования как эффективных методов выявления и устранения уязвимостей программного обеспечения. Цель исследования - оценить совместное использование этих методов и оценить их эффективность, действенность и всесторонний охват при обнаружении дефектов безопасности.

Исследовательский проект позволил получить значительные результаты, продемонстрировав, что интеграция методов метаморфического и комбинаторного тестирования значительно повышает эффективность и качество тестирования безопасности программного обеспечения.

Глава 1

Введение

1.1 Мотивация

В последние годы быстрая эволюция разработки программного обеспечения привела к значительному росту числа ошибок и уязвимостей в программном обеспечении. Растущее число общих уязвимостей и уязвимых мест (CVE), о которых сообщается каждый год, отражает растущие проблемы, возникающие при обеспечении безопасности и надежности программных систем [1].

Традиционные подходы к тестированию программного обеспечения часто оказываются недостаточными для эффективного обнаружения и устранения уязвимостей. С принятием практики непрерывной интеграции в качестве стандарта в отрасли возрос спрос на автоматизированные методы тестирования, которые могут легко интегрироваться в рабочий процесс разработки. Главная цель - выявить и устранить ошибки и уязвимости на самых ранних этапах жизненного цикла разработки программного обеспечения.

1.2 Предлагаемый подход

В данном дипломе предлагается методология, объединяющая две различные техники тестирования: Метаморфическое тестирование и Комбинаторное тестирование. В предлагаемом подходе используется полуавтоматизированная методология генерации тестовых примеров. Используя принципы метаморфического и комбинаторного тестирования, мы стремимся упростить процесс тестирования, обеспечивая при этом адекватное покрытие тестируемой системы. Этот гибридный подход позволяет найти баланс между ручным вмешательством и автоматизированным тестированием, оптимизируя эффективность и результативность процесса тестирования.

Глава 2

Обзор литературы

Угрозы кибербезопасности требуют инновационных методов тестирования для обеспечения надежности и безопасности программного обеспечения. В данном обзоре литературы рассматриваются современные достижения в области комбинаторного тестирования (КТ) и метаморфического тестирования (МТ) для кибербезопасности, изучается интеграция КТ и МТ, оценивается эффективность совместной работы этих методов и их потенциальное применение в кибербезопасности.

2.1 Комбинаторное тестирование в кибербезопасности

В области кибербезопасности комбинаторное тестирование было использовано, в частности, для создания тестовых примеров, направленных на выявление таких уязвимостей, как SQL-инъекции [2] и XSS-инъекции [3]. В обоих исследованиях основное внимание уделялось одному типу уязвимости. Эти исследования показали, что комбинаторное тестирование быстро и

эффективно выявляет уязвимости. Однако важно отметить, что рамки этих исследований несколько ограничены с точки зрения спектра оцениваемых технологий.

2.2 Метаморфическое тестирование в кибербезопасности

Литература, посвященная метаморфическому тестированию (MT) в кибербезопасности, подробно рассматривает его применение и эффективность. Segura *и др.* выделяют ключевую роль MT в решении проблемы оракула при тестировании программного обеспечения, когда правильность результата неочевидна. [4]. Это особенно важно в кибербезопасности, где точные результаты критически важны, но зачастую их трудно определить.

Исследуя возможности применения MT в кибербезопасности, исследователи продемонстрировали его эффективность в обнаружении скрытых ошибок в жизненно важных приложениях. Ярким примером является тестирование компиляторов и обфускаторов кода, где традиционные методы тестирования могут оказаться неэффективными [4] [5]. Использование MT в таких сценариях подчеркивает его способность работать со сложными и критическими программными системами.

Разработка и внедрение фреймворка METRIC [6] ознаменовали собой значительный прогресс в области MT. Этот подход систематически определяет метаморфические отношения (MO) с помощью системы выбора категорий, упрощая процесс идентификации MO, который ранее был ручным и несистематическим. Этот систематический подход особенно полезен для ки-

бербезопасности, где создание надежных тестовых оракулов является сложной задачей. Якушева и др. показали, что выведение MR можно еще больше упростить, используя специфические для данной области рекомендации [7].

2.3 Интеграция комбинаторного и метаморфического тестирования

Сочетание комбинаторного тестирования (КТ) и метаморфического тестирования (МТ) рассматривается в работе Niu *et al.* [8] в инновационном подходе к тестированию программного обеспечения. В их исследовании рассматривается проблема, связанная с созданием автоматизированных тестовых оракулов, которая часто приводит к ручному и подверженному ошибкам процессу тестирования.

Niu *и др.* представляют новый метод под названием COMER, который сочетает МТ с СТ. Этот метод фокусируется на формировании пар тестовых случаев, известных как метаморфические группы (МГ), на основе определенных метаморфических отношений (МО). Эти MG позволяют автоматически проверять результаты тестирования, проверяя, соответствуют ли выходные данные MR.

Глава 3

Методология

Это исследование использует методологию COMER для генерации тестов. Чтобы оценить эффективность этой методологии и провести сравнительный анализ с другими существующими методологиями, я взял на себя задачу реализовать её с нуля с использованием языка программирования Python.

3.1 Методология COMER

Методология COMER предназначена для генерации тестов, используя как традиционные методы комбинаторного тестирования (КТ), так и добавляя метаморфные связи в процессе генерации. После получения входных данных, включая набор метаморфных связей, набор ограничений и абстрактные входные параметры, COMER генерирует тесты, которые соответствуют указанным ограничениям и удовлетворяют определённым метаморфным связям (МС).

3.2 Ограничения

Несмотря на достижения в первоначальной реализации COMER, были выявлены определённые ограничения. Во-первых, использование жадного подхода для генерации тестов КТ являлось заметным недостатком. Хотя этот метод может повысить скорость процесса генерации, он может привести к избыточному количеству тестов, что потенциально снижает эффективность и использование ресурсов. Во-вторых, ориентация реализации исключительно на метаморфные связи с одним входным и одним последующим тестом ограничивала её применимость к сценариям с более сложными МС. В-третьих, отсутствие возможности генерировать тесты для функций, не поддерживаемых в реализации, представляло значительное ограничение.

В моей реализации были предприняты усилия по смягчению некоторых из этих ограничений. Чтобы преодолеть ограничение поддержки только ограниченного числа функций, я работал над улучшением методологии, позволяя генерировать тесты для любой функции. Кроме того, в ответ на обеспокоенность по поводу эффективности жадного подхода были предусмотрены альтернативные алгоритмы для генерации тестов КТ, предоставляя пользователям возможность выбрать наиболее подходящий подход. Однако важно отметить, что ограничение, касающееся МС с множественными входами и последующими тестами, остаётся неустранённым в моей реализации, представляя собой область для потенциальных будущих исследований и улучшений.

Глава 4

Реализация

Основная функция, отвечающая за генерацию тестовых примеров, выглядит следующим образом:

```
def __next__(self) -> OrderedDict:
    r = random.random()
    if r > self._mr_probability and self.generated_cases:
        case_pre = random.choice(self.generated_cases)
        solution = self.csp_solver(case_pre)
        if solution:
            self.add_testcase_to_tested(solution)
            return solution

    return OrderedDict(super().__next__())
```

Функция придерживается основополагающих принципов методологии COMER. В частности, на основе заданной вероятности, self._mr_probability, функция выбирает подход, основанный на комбинаторном тестировании

(КТ) или метаморфическом тестировании (МТ). В КТ для итеративной генерации тестовых примеров используется решатель задачи удовлетворения ограничений (Constraint Satisfaction Problem, CSP). В МТ он выбирает существующий тестовый пример (case_pre) из предоставленного пула (self.generated_cases) и генерирует последующие тестовые примеры с помощью CSP-решателя.

4.1 Использование методологии

Пример демонстрирует, как использовать фреймворк для тестирования веб-приложения. Предположим, мы хотим протестировать веб-приложение, которое позволяет пользователям добавлять закладки и смарт-теги в свои учетные записи. Мы можем определить набор параметров для наших тестовых случаев следующим образом:

```
params = OrderedDict(
    {
        "highlight": [0, 1],
        "status_bar": [0, 1],
        "bookmarks": [0, 1],
        "smart_tags": [0, 1],
    }
)
```

Ограничения определяются как функции, которые принимают тестовый пример на вход и выдают True или False, указывая, должны ли мы включить конкретный тестовый пример в наш домен. Метаморфные связи

определяются с использованием тестового случая в качестве входа и нового последующего действия в качестве выхода. В этом конкретном примере мы не определяем дополнительные MC для генерации конкретных тестовых случаев для простоты.

```
def constraint(highlight, status_bar, **kwargs):
    return highlight == status_bar


def mr(bookmarks, smart_tags, **kwargs):
    return OrderedDict({
        "bookmarks": smart_tags,
        "smart_tags": bookmarks,
        **kwargs
    })
```

После того как домен задан, мы можем создавать абстрактные тестовые примеры. В этом примере мы используем их в качестве входных данных для *Pytest* для создания параметризованных тестовых примеров.

```
@pytest.mark.parametrize("testcase", Comer(params, constraint, MR(mr)))
def test_simple(testcase: OrderedDict):
    assert testcase["highlight"] == testcase["status_bar"]
```

Глава 5

Выводы

В данной дипломе представлено комплексное исследование интеграции комбинаторного тестирования (КТ) и метаморфического тестирования (МТ) для автоматизированного тестирования безопасности веб-приложений. Предложенная реализация фреймворка COMER объединяет сильные стороны КТ и МТ для генерации тестовых примеров, которые могут эффективно выявлять уязвимости в сложном программном обеспечении.

Оценка фреймворка COMER показала его эффективность в генерации тестовых примеров, способных обнаружить уязвимости в веб-приложениях. Результаты показывают, что фреймворк COMER может генерировать тестовые примеры более эффективно, чем традиционные методы КТ, с более высокой частотой обнаружения и аналогичным временем генерации. Однако предварительные тесты текущей реализации показали, что время выполнения в некоторых случаях удваивается по сравнению с традиционным фреймворком CT из-за увеличения количества генерируемых тестовых примеров.

Исследование также продемонстрировало потенциал фреймворка COMER для стандартизации в виде фреймворка или набора инструментов

для автоматизированного тестирования безопасности. Способность фреймворка эффективно выявлять уязвимости и генерировать тестовые примеры делает его ценным инструментом для специалистов по кибербезопасности, стремящихся повысить уровень защиты своих приложений.

Однако исследование также выявило необходимость проведения дополнительных исследований в области систематической генерации метаморфических отношений (MR) для сложного программного обеспечения кибербезопасности. Ручное создание MRs для каждого конкретного тестового случая - трудоемкий и длительный процесс, требующий специальных знаний и опыта. Разработка методологий или инструментов, позволяющих автоматизировать процесс создания MR, значительно повысит эффективность фреймворка COMER.

Список использованной литературы

[1] *CVE statistics*, (accessed Mar. 18, 2024). url: https://www.cve.org/About/Metrics#PublishedCVERecords.

[2] D. E. Simos, J. Zivanovic и M. Leithner, «Automated Combinatorial Testing for Detecting SQL Vulnerabilities in Web Applications,» в *2019 IEEE/ACM 14th International Workshop on Automation of Software Test (AST)*, 2019, с. 55—61. DOI: 10.1109/AST.2019.00014.

[3] B. Garn, I. Kapsalis, D. E. Simos и S. Winkler, «On the Applicability of Combinatorial Testing to Web Application Security Testing: A Case Study,» в *Proceedings of the 2014 Workshop on Joining AcadeMiA and Industry Contributions to Test Automation and Model-Based Testing*, сер. JAMAICA 2014, San Jose, CA, USA: Association for Computing Machinery, 2014, с. 16—21, ISBN: 9781450329330. DOI: 10.1145/2631890.2631894. url: https://doi.org/10.1145/2631890.2631894.

[4] S. Segura, D. Towey, Z. Q. Zhou и T. Y. Chen, «Metamorphic Testing: Testing the Untestable,» *IEEE Software*, т. 37, № 3, с. 46—53, 2020. DOI: 10.1109/MS.2018.2875968.

[5] A. F. Donaldson, H. Evrard, A. Lascu и P. Thomson, «Automated Testing of Graphics Shader Compilers,» *Proc. ACM Program. Lang.*, т. 1, № OOPSLA, 2017. DOI: 10.1145/3133917. url: https://doi.org/10.1145/3133917.

[6] T. Y. Chen, P.-L. Poon и X. Xie, «METRIC: METamorphic Relation Identification based on the Category-choice framework,» *Journal of Systems and Software*, т. 116, с. 177—190, 2016, ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2015.07.037. url: https://www.sciencedirect.com/science/article/pii/S0164121215001624.

[7] S. F. Iakusheva и A. S. Khritankov, «A systematic review of methods for deriving metamorphic relations,» *Program Systems: Theory and Applications*, т. 15, с. 37—86, 2 2024. DOI: 10.25209/2079-3316-2024-15-2-37-86.

[8] X. Niu, Y. Sun, H. Wu и др., «Enhance Combinatorial Testing With Metamorphic Relations,» *IEEE Transactions on Software Engineering*, т. 48, с. 5007—5029, 2021. DOI: 10.1109/TSE.2021.3131548.