Научная статья
Original article
УДК 004.424

НЕЧЕТКОЕ ТЕСТИРОВАНИЕ ГЕНЕТИЧЕСКОГО АЛГОРИТМА

FUZZY TESTING OF THE GENETIC ALGORITHM

Куренная Валерия Олеговна, студент бакалавр, Донской государственный технический университет, г. Ростов-на-Дону (344003 Россия г. Ростов-на-Дону, Гагарина 1), kurinaaa@rambler.ru

Kurennaya Valeria Olegovna, bachelor student, Don State Technical University, Rostov-on-Don (344003 Russia, Rostov-on-Don, Gagarina 1), kurinaaa@rambler.ru

Аннотация. Браузеры стали основным интерфейсом практически для всего онлайн-контента на протяжении почти все пользователи. В результате они также стали чрезвычайно изощренными. Современный Браузер отображает контент, используя множество взаимосвязанных компонентов с интерпретаторы для растущего набора языков, таких как JavaScript, Flash, Java и XSLT. Неудивительно, что браузеры превратились в главные цели для злоумышленников, которые регулярно используют встроенные интерпретаторы для запуска сложных программ. Фаззи программы является один из способов тесирования гипертекстовые элементы на ошибку.

Annotation. Browsers have become the main interface for almost all online content for almost all users. As a result, they have also become extremely sophisticated. The Modern Browser renders content using many interrelated components with interpreters for a growing set of languages such as JavaScript,

Flash, Java, and XSLT. Not surprisingly, browsers have become prime targets for attackers who routinely use built-in interpreters to run complex programs. The fuzzy program is one way to test hypertext elements for error.

Ключевые слова: программирование, тестирование, нечеткая логика, ошибки.

Key words: programming, testing, fuzzy logic, errors.

Генетическое программирование — это вариант эволюционных алгоритмов, вдохновленный биологической эволюции и обеспечивает прозрачность в принятии решений. Он следует теории эволюции Дарвина и порождает новых особей в экосистеме путем рекомбинация текущих характеристик от особей с наивысшей приспособленностью.

Пригодность — это значение, вычисляемое целевой функцией, которая направляет эволюцию процесс. Генетическое программирование использует модульность и возможность повторного использования фрагментов решения в проблемном пространстве для улучшения приспособленности людей.

Было показано, что этот подход очень подходит для генерации фрагментов кода, но не использовался для фазз-тестирования в целом в качестве входных данных программы, обычно неструктурированы и сильно взаимозависимы. Однако наше ключевое понимание заключается в том, что, как описано выше, фаззинг интерпретатора является особым случаем. Использование кода как ввода, генетическое программирование кажется естественным.

Фази прогаммирование воспринимает контекстно-свободную грамматику языка как ввод для генерации тестов. Он использует грамматику для создания деревьев синтаксического анализа и извлекать фрагменты кода из заданного набора тестов.

В нечетком тестировании генетические алгоритмы могут использоваться для создания непрерывного набора тестовых случаев. Генерация тестовых случаев основана на структуре фаззинга и ответах,

полученных от целей фаззинга. Первый набор тестовых случаев создается с использованием генеративного или мутационного подхода, а последующие тестовые наборы создаются с использованием генетической прогрессии:

Каждый тестовый пример оценивается с использованием различных показателей, которые показывают, насколько успешно он вызвал ошибку.

Тестовые случаи с наименьшим баллом удаляются.

Небольшие изменения вносятся в каждый из оставшихся тестовых случаев.

Наконец, алгоритм объединяет тестовые наборы с высокой оценкой, чтобы найти тестовые наборы с более высокими оценками. Этот процесс также используется для дополнения отброшенных тестовых случаев и поиска более успешных кандидатов.

Лучшие практики Fuzz-тестирования

Вот несколько способов оценить эффективность решения для фаззинга.

Улучшить скорость тестирования

Критической метрикой для фаззинга теперь является количество тестовых случаев, которые вы можете запускать в секунду. Чем больше тестовых случаев вы можете запустить за определенный период времени, тем больше вероятность того, что вы обнаружите сбой или ошибку. Более быстрое фаззинг-тестирование также позволяет интегрировать фаззинг в автоматизированные процессы тестирования.

Есть много вещей, которые вы можете сделать, чтобы ускорить свои тестовые примеры, например сделать генеративные или мутационные процедуры более эффективными, распараллелить тестовые примеры, сократить время ожидания и запустить вашу программу в автономном режиме (без пользовательского интерфейса). Если вы сами проводите фазз-тестирование, вы также можете запустить его на более мощном оборудовании.

Сокращение тестовых случаев

Поскольку фаззинг изменяет входные данные случайным образом, тестовые случаи часто содержат вариации или мутации, которые на самом

деле не вызывают ошибки. Сокращение тестовых случаев означает сужение тестовых случаев до наименьшего набора изменений, которые могут вызвать ошибку или сбой.

Это сокращение можно выполнить вручную, но оно также может быть выполнено автоматически с помощью решения для фаззинга. Когда происходит сбой, фаззер может повторно запускать тестовый пример несколько раз. Каждый раз, когда запускается ошибка, она может постепенно уменьшать изменения во входных данных (по сравнению с базовыми допустимыми входными данными), пока не будет достигнуто минимальное изменение, необходимое для срабатывания ошибки. Это упрощает анализ и помогает точно понять, какая часть входных данных связана с ошибкой или сбоем.

Покрытие кода отслеживания

Покрытие кода — это мера того, какую часть программного кода выполнил фаззер. Идея состоит в том, что чем шире покрытие, тем тщательнее фаззер тестирует программы. Существует множество способов измерения покрытия кода, включая строки, блоки кода, ветви и пути кода.

Измерение покрытия кода для фаззинга может быть затруднено и может потребовать двоичного инструментария для отслеживания того, какой код выполняется во время каждого запроса фаззинга.

Покрытие кода не является идеальной мерой для фазз-тестирования, потому что большая часть кода приложения не приведет к ошибке, даже если она будет выполнена. Тем не менее, некоторая форма измерения покрытия кода может дать представление о том, что на самом деле запускает фаззер в вашей программе, и позволить точную настройку активности фаззера.

Литература

1. Ван Хайтао Исследование системы осведомленности об информационной безопасности на основе технологий больших данных и искусственного интеллекта, Технологии и приложения сетевой безопасности, 2018 (3): 60-63.

2. Чжэн Яньфан Исследования и практика применения искусственного интеллекта и технологий анализа в системе ситуационной осведомленности информационной безопасности Мир цифровых коммуникаций, 2018, № 160 (4): 229.
3. Чжэн Фан Информационная безопасность в эпоху искусственного интеллекта, Исследования в области информационной безопасности, 2017, 3 (11): 966-967.
4. Дэн Вэньбинь Проблемы и меры противодействия надзору за информационной безопасностью в эпоху искусственного интеллекта Информационная безопасность Китая, 2018 г., 106 (10): 106-108.

References

1. Wang Haitao Research on Information Security Situation Awareness System Based on Big Data and Artificial Intelligence Technology, Network Security Technology and Application, 2018 (3): 60-63.
2. Zheng Yanfang Research and Practice of Artificial Intelligence Application and Analysis Technology in Information Security Situational Awareness System Digital Communication World, 2018, No.160 (4):229.
3. Zheng Fang Information Security in the Age of Artificial Intelligence, Information Security Research, 2017, 3 (11): 966-967.
4. Deng Wenbing Challenges and Countermeasures for Information Security Supervision in the Age of Artificial Intelligence China Information Security, 2018, 106 (10): 106-108.

© Куренная В.О., 2022 Научно-образовательный журнал для студентов и преподавателей «StudNet» №6/2022.

Для цитирования: Куренная В.О. НЕЧЕТКОЕ ТЕСТИРОВАНИЕ ГЕНЕТИЧЕСКОГО АЛГОРИТМА// Научно-образовательный журнал для студентов и преподавателей «StudNet» №6/2022.