

УДК 004.056.5

Классификация средств фаззинга по методу генерации данных

Ажгирей Вадим Сергеевич¹

azhgireyvs@student.bmstu.ru

Самрега Александр Андреевич²

samregaaa@alfastrah.ru

Малахов Михаил Валерьевич¹

malahovmv@bmstu.ru

Глинская Елена Вячеславовна^{1(*)}

glinskaya@bmstu.ru

SPIN-код: 5430-3023

¹ МГТУ им. Н.Э. Баумана, Москва, Россия² АО «АльфаСтрахование», Москва, Россия

Аннотация. В работе рассмотрен фаззинг, являющийся одним из наиболее популярных методов выявления уязвимостей в программном обеспечении. Представлена классификация средств фаззинга в зависимости от метода генерации данных. Описаны ключевые особенности, а также достоинства и недостатки различных типов фаззеров. Сделан вывод о целесообразности использования фаззеров различных типов в зависимости от особенностей целевого программного обеспечения.

Ключевые слова: информационная безопасность, выявление уязвимостей, безопасность программного обеспечения, фаззинг, способы генерации входных данных

Введение. В настоящее время особую актуальность приобретает задача обеспечения информационной безопасности отечественных автоматизированных систем. Решение указанной задачи напрямую связано с устранением угроз, связанных с наличием уязвимостей в программном обеспечении указанных систем. Одним из наиболее распространенных методов выявления уязвимостей является фаззинг [1], однако эффективность применения этого метода напрямую зависит от типа выбранного фаззера и особенностей целевого программного обеспечения. В связи с этим, возникает потребность в классификации фаззеров и определения целесообразности их применения к целевому программному обеспечению.

Методы и материалы; результаты. Фаззинг — это метод автоматического тестирования программного обеспечения, заключающийся в направленном введении аномальных или специально сконструированных входных данных с целью вызова нештатных состояний тестируемой системы [2].

В зависимости от метода генерации данных принято разделять подход к фаззингу на несколько типов [3]:

1. *На основе грамматики.* Таким фаззерам для работы требуется грамматика — определенный набор правил, описывающих структуру ожидаемых входных данных. Как только фаззеру будут известны эти правила, он сможет генерировать новые комбинации на их основе. Данный подход позволяет контролируемо отклоняться от правил грамматики, генерируя частично или полностью некорректные данные, нарушая синтаксис или семантику, пред-

писанные грамматикой. Это обеспечивает высокую точность и реалистичность входных данных с низким уровнем случайности, однако требует ручного описания грамматики. Он используется при тестировании сложных форматов и протоколов, где крайне важна корректность структуры.

2. *На основе мутаций.* Фаззеры на основе мутаций последовательно случайно видоизменяют входные данные из предыдущих запусков. Для начала работы ему необходим начальный корпус — набор репрезентативных входных данных, таких как реальные пользовательские данные или существующие тесты. В процессе работы фаззер применяет к данным из корпуса различные мутационные стратегии: изменяет биты и байты, переставляет блоки данных, добавляет или удаляет фрагменты, комбинирует части различных последовательностей. Полученные данные подаются на вход программе для выявления аномалий в ее работе. Этот подход отличается простотой и универсальностью, но ограничен в генерации сложных структурированных данных. Он эффективен для тестирования программ с неструктурированными данными или при отсутствии заранее заданной грамматики.

3. *Построение грамматики по данным.* Данный подход представляет собой гибридную методику вышеописанных методов, объединяющую преимущества фаззинга на основе мутаций и грамматики. Вместо использования заранее заданной грамматики, система автоматически выводит правила и структуры входных данных из предоставленного корпуса. Ключевым преимуществом такого подхода является способность создавать сложные структурированные входные данные, которые сохраняют семантическую корректность, но при этом содержат контролируемые отклонения и аномалии, потенциально способные выявить сложные и нетривиальные уязвимости. Данный тип фаззеров способен автоматически выявлять структуру входных данных и формировать сложные тесты, но характеризуется высокой вычислительной сложностью и зависимостью от качества корпуса. Он применяется в случаях, когда нет готовой грамматики, но есть большой набор валидных примеров для ее автоматического построения.

4. *На основе покрытия кода.* Данные фаззеры устроены по принципу генетического алгоритма и стремятся максимизировать покрытие кода тестируемой программы. В отличие от фаззеров черного ящика (dumb-fuzzers), которые не используют обратную связь по покрытию от программы, эти (coverage-guided fuzzers) анализируют выполнение программы и используют полученную информацию для отбора наиболее перспективных входных данных для последующих мутаций. Такие фаззеры демонстрируют высокую эффективность и способны находить скрытые уязвимости, однако требуют значительных вычислительных ресурсов. Они используются для глубокой проверки программ с большим количеством логики и разветвлений.

Заключение. В работе рассмотрен метод фаззинга. Представлена классификация фаззеров по методу генерации данных и сделаны выводы о целесообразности применения фаззеров каждого типа в зависимости от особенностей целевого программного обеспечения, при этом показано, что в общем

случае наибольшая практическая эффективность достигается при выборе средств фаззинга с обратной связью по покрытию.

Список источников

- [1] Sutton M., Greene A., Amini P. Fuzzing: Brute Force Vulnerability Discovery. Boston, Addison-Wesley Professional, 2007, 513 p.
- [2] Кулямин В.В. Обзор методов динамического анализа программного обеспечения. *Труды ИСП РАН*, 2023, т. 35, № 4, с. 7–44. [https://doi.org/10.15514/ISPRAS-2023-35\(4\)-1](https://doi.org/10.15514/ISPRAS-2023-35(4)-1)
- [3] Станкевичус А.А., Храпунов П.А., Трищенко А.В. Современное тестирование программного обеспечения. Технология автоматизированного тестирования — фаззинг. *Молодежь в науке. 20-я науч.-техн. конф.: сб. тр.* Саров, РФЯЦ-ВНИИЭФ, 2023, с. 559–565. https://doi.org/10.53403/9785951505460_559