# Исследование и разработка методов динамического анализа встраиваемого программного обеспечения

***Ниськов Федор Владимирович***

*Студент*

*Факультет ВМК МГУ имени М. В. Ломоносова, Москва, Россия*

*E-mail:* fedor.niskov@ispras.ru

***Научный руководитель*** *— Аветисян Арутюн Ишханович, Курмангалеев Шамиль Фаимович*

В современном мире надёжность и безопасность программного обеспечения имеют очень важное значение, поэтому разработчики уделяют существенное внимание процессу тестирования и выявления ошибок. Сложность и размеры современного программного обеспечения требуют автоматизации: было разработано большое количество специальных инструментов, помогающих обнаруживать ошибки в программах.

Эффективным способом поиска ошибок является фаззинг — метод динамического анализа для расширения тестового покрытия и поиска входных данных, приводящих к аварийному завершению. Эффективность фаззинга можно повысить, используя динамическое символьное выполнение (DSE), с помощью которого можно получать новые входные данные, дающие новые пути, что приводит к расширению покрытия кода.

В то время как достаточно большой арсенал средств анализа кода доступен для ЭВМ общего назначения, список программных средств для тестирования программного обеспечения специализированных встраиваемых устройств гораздо меньше. Здесь возникают существенные трудности для алгоритмов фаззинга и символьного выполнения, такие как необходимость наличия устройства, его ограниченная производительность и трудности при обеспечении масштабируемого тестирования.

Возможным подходом при анализе встраиваемого ПО является частичная эмуляция — выполнение фрагмента кода, при котором происходит моделирование основных механизмов процессора и памяти, а обращения к программно-аппаратному окружению отсутствуют или используют упрощённую реализацию. Т.е. для анализа можно выполнять только фрагмент кода, выполняющий обработку входных данных. Нужно воссоздать начальное состояние в эмуляторе и запустить процесс выполнения, в ходе которого можно обнару-

живать аварийные завершения, а также проводить символьное выполнение. Это означает, что нужна комбинация технологий — отладочных механизмов устройства и ПО эмуляции: отладчик фиксирует устройство в начальном состоянии на точке останова и может сообщать необходимую информацию о состоянии (значения регистров и участков памяти) эмулятору в ходе выполнения.

Таким образом, для преодоления трудностей анализа встраиваемого ПО нужна эффективная комбинация существующих технологий. В данном направлении уже есть важная разработка — фреймворк Avatar [1]. Это известный фреймворк оркестрации, т.е. его задача — комбинация нескольких инструментов и обеспечение взаимодействия между ними. С помощью Avatar можно осуществить описанную выше схему: проводить символьное выполнение, получая нужную информацию через отладочный интерфейс устройства. Avatar поддерживает необходимые для этого инструменты: инструмент символьного выполнения Angr и отладчик GDB.

В ходе данной работы были выявлены недостатки фреймворка Avatar, для их устранения были выполнены следующие улучшения:

- поддержка архитектуры MIPS и порядка байт Big Endian;
- повышение производительности за счёт более эффективной организации запросов к отладчику;
- возможность выполнения при отсутствии знаний о назначении диапазонов адресов памяти.

Описанные улучшения были предложены в официальный репозиторий фреймворка Avatar и одобрены его авторами [2].

Исправленный фреймворк был протестирован на различных модельных примерах и на примере реального ПО — ОС OpenWrt для маршрутизаторов.

Также в ходе данной работы был разработан модуль DSE для фаззера Crusher, разрабатываемого в ИСП РАН, с целью повышения эффективности фаззинга ПО для встраиваемых устройств.

**Литература**

1. Muench M. et al. Avatar 2: A multi-target orchestration platform // Proc. Workshop Binary Anal. Res. (Colocated NDSS Symp.). — 2018. — Т. 18. — С. 1–11.
2. Патч для Avatar (Ниськов Ф. В.): `https://github.com/avatartwo/avatar2/pull/65`