Ускорение фаззинга JAVA-приложений на базе фреймворков ISPFuzzer и Kelinci

Гайфутдинова Фарида Хамитовна[1]
Мишечкин Максим Владимирович[2]

1: *Студент, факультет ВМК МГУ имени М. В. Ломоносова, Москва, Россия*
2: *Аспирант, ИСП РАН, Москва, Россия*
E-mail: farida.gaif@ispras.ru, mishmax@ispras.ru

Научный руководитель — *Курмангалеев Шамиль Фаимович*

Фаззинг — эффективная методика, широко используемая для поиска уязвимостей в программном обеспечении. Использование инструментов фаззинг-тестирования необходимо на всех этапах разработки, тестирования и эксплуатации программного обеспечения. Ошибки можно обнаружить как при наличии, так и при отсутствии исходного кода.

Описание метода. Одним из самых первых и известных инструментов для фаззинг-тестирования является фаззер AFL [4]. Однако он предназначен для программ, написанных на языках C/C++. Фреймворк Kelinci [1] в связке с AFL позволяет осуществлять фаззинг-тестирование приложений, разрабатываемых на языке JAVA. В данном подходе общение AFL и Kelinci происходит не напрямую, а через программу `interface`. Она отправляет Kelinci входные данные через TCP соединение для запуска на них тестируемого приложения. Полученные статус завершения программы и покрытие в виде битовой карты отправляются программе `interface` снова через TCP соединение. Там они записываются в разделяемую память, для последующего прочтения их фаззером AFl. Заметим, что пересылка данных через TCP соединение вызывает дополнительные временные расходы. В работе были предложены оптимизации данного метода фаззинг-тестирования JAVA-приложений, которые позволяют избавиться от использования TCP сервера, многократно ускорить работу фаззера и скорость инструментации бибилиотек.

Оптимизации. В исследовании представлена связка фаззера ISPFuzzer [3] и фреймворка Kelinci, общение которых происходит напрямую. ISPFuzzer отправляет в Kelinci запросы на запуск тестируемого приложения через неименованные каналы, Kelinci запускает его. Результат запуска, то есть код завершения программы, фреймворк отправляет обратно в ISPFuzzer снова через анонимный канал.

В данном методе битовая карта записывается сразу в разделяемую память во время запуска приложения. AFL считывает массив напрямую из этой памяти, никаких дополнительных пересылок не производится. При запуске инструмента AFL+Kelinci было замечено, что вложенные jar-файлы не инструментируются, а также скорость инструментации class-файлов очень низкая. Решение этой проблемы предложено в данной работе - произвести распаковку всех jar-файлов, в т.ч. и вложенных, а далее проводить инструментацию полученных директорий. Таким образом экспериментально было показано уменьшение времени инструментации библиотек и была добавлена поддержка инструментации вложенных jar-файлов.

	`commons-imaging-1.0`		`commons-math3-3.6.1`	
	AFL+Kelinci	ISPFuzzer+Kelinci	AFL+Kelinci	ISPFuzzer+Kelinci
total execs	$26.5K$	$48K$	$35K$	$15M$
exec speed (execs/sec)	7	13.5	9.9	4200
instrumentation time (sec)	58.5	1	363	1
total paths	18	250	28	37

Таблица 1: Результаты тестирования.

Результаты. Тестирование метода производилось с помощью библиотек `commons-imaging-1.0` и `commons-math3-3.6.1`, написанных на JAVA. Они содержать 420 и 1301 class-файл соответственно, которые предварительно были проинструментированы. В одно ядро в течении часа проводилось фаззинг-тестирование функций из указанных библиотек при помощи двух связок инструментов: AFL+Kelinci и ISPFuzzer+Kelinci. Замерялись скорость запусков программы в секунду, их общее число и время инструментации библиотеки. В таблице 1 представлены полученные результаты, усредненные за 5 запусков. Описанные выше оптимизации позволили в несколько раз увеличить скорость запусков тестируемого приложения, а также многократно сократили время инструментации библиотек.

Литература

1. AFL-based Fuzzing for Java with Kelinci: `https://github.com/isstac/kelinci/blob/master/docs/ccs17-kersten.pdf`
2. ИСП Crusher: комплекс динамического анализа программ: `https://www.ispras.ru/technologies/crusher/`
3. AFL technical details: `https://github.com/google/AFL/blob/master/docs/technical_details.txt`