Для определения влияния инерционности систем РН и РЭВ, времени, затрачиваемого на обнаружение сигнала, вероятности правильного априорного определения параметров формирующих функций необходимо более детальное исследование.

Библиографические ссылки

1. Орощук И. М. Основные направления применения имитационных помех в системах радиосвязи. Классификация способов ими [Электронный ресурс] // Центр информационной безопасности : портал. 2005. URL: http://www.bezpeka.com/ru/lib/spec/metr/art184.html.

2. Ширман Я. Д. Теоретические основы радиолокации. М. : Сов. радио, 1970. 561 с.

3. Воронов Д. Н. Критерии оценки имитостойкости командно-телеметрических радиолиний // Системи обробки інформації. 2007. № 4(62). С. 14–16.

References

1. Oroshchuk I. M. [Main pathways of simulated echoes application in communication systems. Classification of simulated echoes]. *Tsentr informatsionnoy bezopasnosti: portal.* 2005 (In Russ.) Available at: URL: http://www.bezpeka.com/ru/lib/spec/metr/art184.html.

2. Shirman Y. D. *Teoreticheskiye osnovy radiolokatsii* (Theoretical foundations of radiolocation). Moscow, Soviet radio, 1970, 561 p.

3. Voronov D. N. *Sistemi obrobki iniformatsii.* 2007, no. 4 (62), p. 14–16.

УДК 004.056

МЕТОДИКА ДИНАМИЧЕСКОГО АНАЛИЗА УЯЗВИМОСТЕЙ В БИНАРНОМ КОДЕ*

М. О. Шудрак, В. В. Золотарев, И. А. Лубкин

Сибирский государственный аэрокосмический университет имени академика М. Ф. Решетнева
Российская Федерация, 660014, Красноярск, просп. им. газ. «Красноярский рабочий», 31
E-mail: mxmssh@gmail.com

Описывается методика и ее реализация в виде программного продукта, осуществляющая обнаружение уязвимостей в бинарном коде с использованием сочетания технологий динамической бинарной инструментации, анализа покрытия кода и так называемого фаззинга – технологии генерации потенциально ошибочных данных и мониторинга результата. В результате было разработано программное средство для поиска уязвимостей в сетевых и файловых приложениях, позволяющая значительно оптимизировать процесс динамического тестирования.

Ключевые слова: динамический анализ, уязвимость, фаззинг, бинарная инструментация.

THE TECHNIQUE FOR BINARY EXECUTABLES VULNERABILITIES DETECTION

M. O. Shudrak,V. V. Zolotarev, I. A. Lubkin

Siberian State Aerospace University named after academician M. F. Reshetnev
31, Krasnoyarsky Rabochy Av., Krasnoyarsk, 660014, Russian Federation. E-mail: mxmssh@gmail.com

The article describes dynamic analysis techniques and cloud-based platform for software security and reliability testing. In the article the authors contribute to the dynamic execution analysis techniques. In the first part of the article the authors describe the technique of dynamic binary analysis which is referred to as fuzzing. The authors show basic architecture of the tool for security and reliability testing of application and vulnerabilities detection. Due to the use of the dynamic binary instrumentation this technique implementation is much faster than ones applied previously. The technique described in the article has been implemented as the software platform which includes web-interface, virtual machine dispatcher, dynamic instrumentation library, debugger, special dll, protocol description and fuzzing manager tool.

Keywords: dynamic analysis, vulnerability, fuzzing, binary instrumentation.

* Работа выполнена в рамках гранта по соглашению № 14.132.21.1365 от 22.10.2012 федеральной целевой программы «Научные и научно-педагогические кадры инновационной России» на 2009–2013 годы.

На сегодняшний день компании-разработчики ПО сталкиваются с серьёзнейшими рисками, связанными с безопасностью собственных продуктов. Эти риски возникают вследствие того, что процесс разработки невозможен без тех или иных ошибок и недочетов. Так, за предыдущий год эксперты обнаружили более 9 000 различных уязвимостей «нулевого дня», а это 26 уязвимостей в день! Для решения этой проблемы используется множество различных методик и подходов, которые могут эффективно применяться на том или ином этапе разработки и эксплуатации ПО [1]. В данной статье мы остановимся лишь на динамическом анализе, являющемся наиболее эффективным для анализа бинарного кода, так как метод взаимодействует с реально работающим приложением [2; 3]. Наиболее существенным недостатком динамического анализа на сегодняшний день является сложность и высокая ресурсозатратность для полноценного поиска уязвимостей. Мы попытаемся частично решить эти проблемы.

Методика гибридного анализа бинарного кода. Методика динамической бинарной инструментации предусматривает проведение тестирования приложения в процессе его исполнения. Как правило, уязвимость в ПО возникает в процессе обработки внешних или пользовательских данных. Для тестирования корректности обработки этих данных часто используется так называемая технология фаззинга, базовая схема которого приведена на рис. 1.

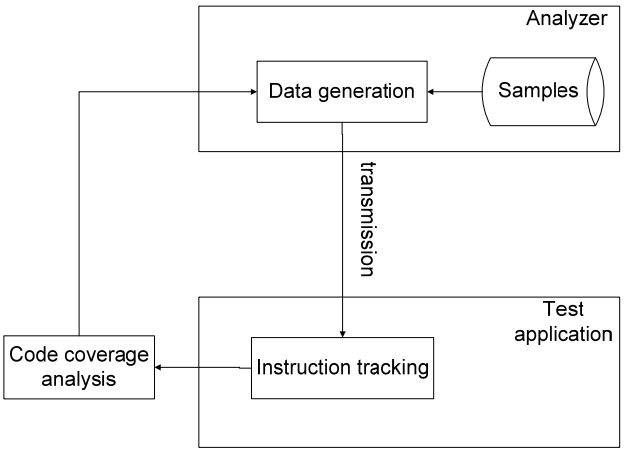

Рис. 1. Базовая схема системы

Опишем технологию динамического анализа и проведем сравнительный анализ полученного решения. Процесс динамического анализа ПО подразумевает анализ продукта в процессе его работы. Такой подход позволяет обнаружить ошибки и уязвимости, возникающие в процессе работы, которые непосредственно доступны для эксплуатации. Для этого система на основе предварительно описанного протокола автоматически генерирует тестовые данные и посылает их в выбранный интерфейс. Затем специальный модуль производит мониторинг исполняемых инструкций (строк кода) и ошибок в процессе выполнения программы. Однако необходимо отметить, что использование такой технологии значительно снижает производительность анализируемой программы. Для решения этой проблемы в рамках проекта было предложено использовать методику динамической бинарной инструментации (DBI) анализируемого модуля. Основная схема методики приведена на рис. 2.

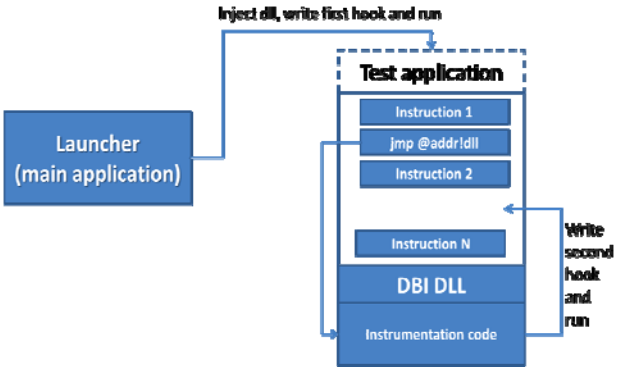

Рис. 2. Описание технологии DBI

Ее идея заключается в возможности перехвата и анализа графа потока управления программы с помощью вставки специальных инструкций в бинарный код анализируемого приложения. Эксперименты показали, что применение DBI позволяет анализировать программу на несколько порядков быстрее, чем при использовании классических методик автоматизированной отладки. Сравнение производительности выбранного решения приведено на рис. 3.

Для повышения гибкости описываемая платформа должна обладать возможностью описания протокола генерации тестовых кейсов с помощью специального языка. Для решения этой задачи был задействован хорошо зарекомендовавший себя ранее блоковый подход к описанию данных. Такой подход частично схож с описанием скриптов в классических языках программирования, однако в значительно урезанном виде. Тестовый пример и общее описание языковых конструкций приведено на рис. 4.

Помимо динамического анализа в процессе тестирования используются элементы методики статического анализа на этапе оценки степени опасности найденной ошибки. Для этого участок исходного или декомпилированного кода, на котором произошла ошибка, сравнивается с шаблонами уязвимого кода и в случае наличия совпадений производится классификация степени опасности данной уязвимости в соответствии с CVSS (Common Vulnerability Scoring System).

Отдельно необходимо остановиться на системе декомпиляции бинарного кода (декомпиляция – процесс восстановления относительно эквивалентного исходного кода из исполняемого модуля) в случае отсутствия исходного. Анализ бинарного кода является нетривиальной задачей в силу того, что в процессе компиляции исходного кода программы теряется важная с точки зрения анализа информация. Процесс анализа бинарного кода требует некоторого уровня его абстракции от машинного кода к человеку. Таким

образом, основная идея заключается в том, что декомпиляция бинарного кода не требует представления машинного кода на языке высокого уровня, достаточно восстановить алгоритм функционирования участка анализируемого кода (тем самым добиться соответствующего уровня абстракции). Для решения этой задачи необходимо рассматривать бинарные инструкции как совокупность элементарных операций над ресурсами, где в качестве ресурса могут выступать регистры, память или регистры флагов процессора.

Апробация технологии. Одним из самых важных требований, предъявляемых к системе, является возможность ее использования в различных операционных системах. Для этого было принято решение максимально (где это возможно) использовать кроссплатформенный язык программирования Python, а где невозможно – C/C++. Такой выбор был продиктован тем, что наряду со своими широкими возможностями, язык Python позволяет легко интегрироваться в проекты на C/C++. Для решения задачи отображения информации была реализована библиотека визуализации графа потока выполнения программы с возможностью фильтрации необходимой информации об уязвимых участках тестируемой программы. Пример детектирования уязвимости в тестовом FTP-сервере приведен на рис. 4.

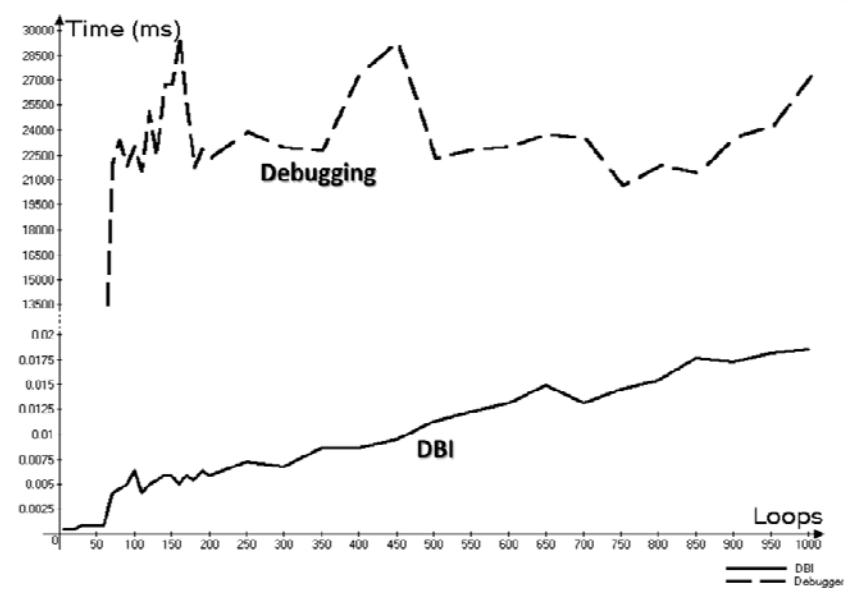

Рис. 3. Сравнение производительности решения

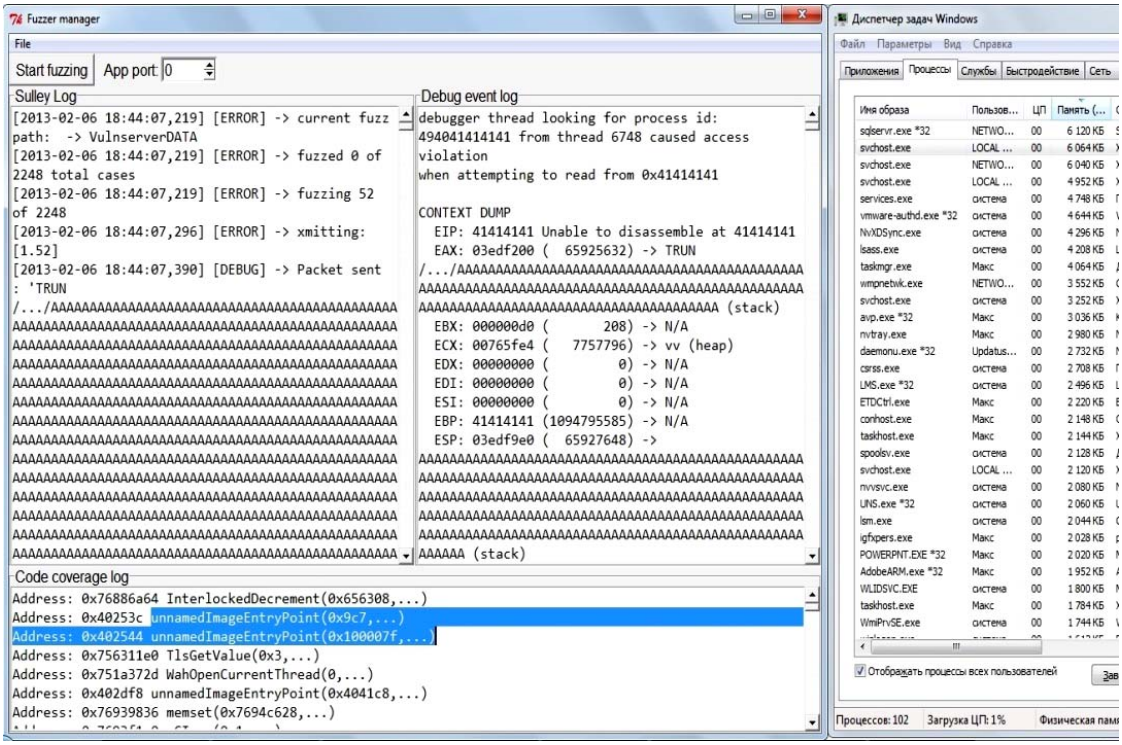

Рис. 4. Пример детектирования уязвимости

Ключевым этапом, отображенным на схеме, является генерация данных и анализ покрытия кода, который необходим для оценки эффективности проведенного тестирования. Не будем подробно останавливаться на каждом этапе работы системы, лишь отметим, что цель анализатора – сгенерировать данные, которые приведут к ошибке в тестируемом приложении. Для оценки эффективности такого тестирования необходимо оценить степень покрытия бинарного кода тестами. В ходе работы над программным средством были протестированы различные решения оценки покрытия кода: от отладки до динамической бинарной инструментации [4]. Последняя технология показала наиболее значительный прирост к скорости анализа (на 3–4 порядка в сравнении с отладкой). DBI-технология использует виртуальную машину уровня процесса ОС, в которую внедряется специальная dll, описывающая то, как необходимо проводить анализ тестируемой программы. Такая схема позволяет в значительной степени оптимизировать анализ, так как операционной системе нет необходимости в переключении контекста процессора между анализатором и тестируемой программой.

Разработанная система была реализована с использованием следующих фреймворков:

1) Sulley-фреймворк – для генерации тестовых данных [5].

2) Intel PIN – DBI фреймворк для виртуализации тестируемого приложения [6].

3) библиотека PyDBG – для отладки тестируемого приложения и перехвата исключений.

Таким образом, авторами была разработана методика и программное средство, обладающее возможностью тестирования различных приложений, обрабатывающих сетевой трафик и различные пользовательские файлы. На данный момент проект находится в стадии альфа-тестирования. Дальнейшая работа предполагает расширение числа доступных протоколов для тестирования, использование более эффективных методик анализа покрытия кода, а также применение интеллектуальных алгоритмов для генерации тестовых данных с учетом результатов покрытия кода тестами.

В заключение хотелось бы отметить, что поиск уязвимостей является на сегодняшний день очень важным и необходимым этапом жизненного цикла разработки и поддержки ПО.

References

1. Giuseppe Desoli, Nikolay Mateev, Evelyn Duesterwald, Paolo Faraboschi, and Joseph A. Fisher. Deli: A new run-time control point. *In Proceedings of the 35th Annual Symposium on Microarchitecture (MICRO35)*, p. 257–270, Istanbul, Turkey, November 2002.

2. Henry S., Kafura D. Software structure metrics based on information flow. *IEEE Transactions on Software Engineering,* vol. SE-7, Issue 5, Sept. 1981, p. 510–518.

3. Marco Cova, Viktoria Felmetsger, Greg Banks, Giovanni Vigna. Static Detection of Vulnerabilities in x86 Executables, *Computer Security Applications Conference, 2006. ACSAC '06. 22nd Annual*, vol. 2, no. 7, p. 269–278.

4. DBI (2007). Available at: http://uninformed.org/index.cgi?v=7&a=1&p=3 (accessed 14.10.2013).

5. Pedram Amini. Fuzzing Framework. *Black Hat USA*, vol. 14, Aug. 2007, p. 211–217.

6. PIN description. Available at: http://software.intel.com/en-us/articles/pin-a-dynamic-binary-instrumentation-tool (accessed 14.10.2013).

УДК 517.55

ОБ АНАЛИТИЧЕСКОМ ПРОДОЛЖЕНИИ КРАТНОГО СТЕПЕННОГО РЯДА С ПОМОЩЬЮ *m*-ОДНОРОДНЫХ ПОЛИНОМОВ МАТРИЧНЫМ МЕТОДОМ В ОБОБЩЕННУЮ ЗВЕЗДУ МИТТАГ-ЛЕФФЛЕРА

Е. И. Яковлев

Сибирский государственный аэрокосмический университет имени академика М. Ф. Решетнева
Российская Федерация, Красноярск, просп. им. газ. «Красноярский рабочий», 31
E-mail:yei@nm.ru

Рассмотрено аналитическое продолжение кратного степенного ряда в класс областей, обобщающих звездные. С помощью переразложения кратного степенного ряда по m-однородным полиномам строится продолжение этого ряда в $(m_1, \ldots, m_n)$*-круговые области, которые являются естественным обобщением круговых областей в* $\boldsymbol{C}^n$*. Опираясь на это разложение, данный кратный степенной ряд аналитически продолжается в максимальную m-звездную область, называемую m-звездой Миттаг-Леффлера функции f, определяемой этим рядом. Это аналитическое продолжение представляет собой суперпозицию m-однородных полиномов, по которым разлагается степенной ряд, с бесконечной треугольной матрицей, элементы которой не зависят от функции f. Приводится пример, когда m-звезда Миттаг-Леффлера отличается от обычной звезды Миттаг-Леффлера.*