даже при небольших изменениях объема исходных данных, поступающих на вход алгоритмов.

ЛИТЕРАТУРА

1. *Юдин Д. Б.* Математики измеряют сложность. М.: Книжный дом «Либроком», 2009. 192 с.
2. *Быкова В. В.* Математические методы анализа рекурсивных алгоритмов // Журнал СФУ. Математика и физика. 2008. № 1(3). С. 236–246.
3. *Василенко О. Н.* Теоретико-числовые алгоритмы в криптографии. М.: МЦНМО, 2006. 336 с.
4. *Быкова В. В.* Метод распознавания классов алгоритмов на основе асимптотики эластичности функции сложности // Журнал СФУ. Математика и физика. 2009. № 2(1). С. 48–61.

УДК 681.3

АНАЛИЗ РАБОТЫ ПРОГРАММЫ С РЕСУРСАМИ: ВЫЯВЛЕНИЕ НЕНАДЁЖНЫХ И НЕБЕЗОПАСНЫХ ОПЕРАЦИЙ[1]

В. В. Горелов

Современное программное обеспечение обладает большим числом разнообразных свойств. Их можно условно разделить на положительные — это те, которые позволяют решить задачу оптимальным методом, и отрицательные, которые мешают задуманному решению. Среди отрицательных свойств программы основными являются ненадёжность и небезопасность. Под надёжностью программы (или программно-аппаратного комплекса) подразумевается её способность, путём действия или бездействия, работать без причинения вреда вовне. Под безопасностью — устойчивость к внешнему воздействию, которое может нарушить работу программы. Данные понятия связаны друг с другом, но существуют разные методы для разработки надёжных и безопасных программ. Здесь предложен метод обнаружения (в процессе работы отлаживаемой программы) нежелательных с точки зрения надёжности и безопасности моментов её работы. Обнаружение подразумевает не только выявление факта нарушения правильной последовательности работы с ресурсами, но и предоставление разработчику детальной информации о конкретном месте ошибки в исходном коде программы. Кроме того, предоставляется информация о ходе её появления в случае не одномоментного (не в одном месте программы) её проистечения.

Уникальность предложенного метода в том, что число ресурсов, с которым он может работать, заранее не ограничено. Такой подход разработан потому, что для большого числа разных ресурсов был замечен общий класс однотипных ошибочных конструкций в программах. В частности, к этому классу отнесены следующие шаблонные ошибки: утечки ресурсов; использование ресурсов после их освобождения; повторные освобождения ресурсов; использование (неинициализированных) ресурсов без их предварительного захвата; использование ресурсов за их границами (относится к динамической памяти и адресному пространству); нарушение вызываемого механизма захвата и освобождения ресурсов (функция захвата возвращает идентификатор ранее захваченного и ещё не освобождённого ресурса) и другие.

Для решения задачи представления неограниченного количества ресурсов разработан язык описания ресурсов (для языка программирования Си). Для описания ресур-

[1]Работа выполнена в рамках реализации ФЦП «Научные и научно-педагогические кадры инновационной России» на 2009–2013 гг. (гос. контракт № П1010).

сов необходимо специальным образом составить файл-шаблон в формате XML. После этого подать его на вход программе-генератору инструментируемых библиотек. Эти библиотеки создаются в результате обхода дерева описания ресурсов с созданием кода, который будет встроен в анализируемую программу. Далее с помощью штатного компилятора происходит «вживление» инструментируемого кода. Затем следует запуск программы, в результате которой все функции, которые работают с описанными ресурсами, будут перехвачены, все входные и выходные параметры проанализированы. Результат работы протоколируется в файл или отсылается программе-анализатору. Данная программа на основании полученных протокольных сообщений генерирует отчёт для разработчика об обнаруженных дефектах в программе, что позволяет ему их устранить.

Метод анализа реализован для прикладных программ на языке Си под системы POSIX и Windows, однако при необходимости может быть расширен на Си++, а также адаптирован для ядер операционных систем.

УДК 004.94

О ПОДХОДЕ К ГЕНЕРАЦИИ ДАННЫХ ДЛЯ ТЕСТИРОВАНИЯ ПРИКЛАДНОГО И СИСТЕМНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ МЕТОДОМ ФАЗИНГА[1]

А. Н. Макаров

Для тестирования надежности системного и прикладного программного обеспечения (ПО) часто применяют метод, получивший название фазинг (fuzzing). Фазинг является одним из видов стрессового тестирования, потенциально позволяющий выявлять ошибки в процессе функционирования ПО [1]. Существует много различных инструментальных средств тестирования методом фазинга, являющихся как коммерческими продуктами, так и свободно распространяемыми. Интерес к тестированию методом фазинга обусловлен все возрастающими требованиями к надежности ПО.

Привлекательность метода заключается в простоте реализации, возможности проведения тестирования в автоматическом режиме, проведении тестирования без исходных кодов, а также в широкой применимости метода. Тестирование методом фазинга можно адаптировать ко многим видам ПО. Тестируемое ПО будем называть *целевым*, а входные данные, сгенерированные для целевого ПО, — *тестами*.

Тестирование посредством фазинга заключается в передаче некорректных данных целевому ПО. Основная задача в том, как описать модель данных и на ее основе создать некорректные данные, т. е. тесты. Различают два типовых подхода к генерации тестов: мутационный и генерационный фазинг. Первый подход предполагает внесение искажений в уже готовые корректные данные. Второй подход требует использования генератора данных заданного формата. Для мутационного фазинга требуется определения «места» внесения искажений, поскольку беспорядочное внесение искажений приводит к тому, что целевое ПО игнорирует тесты. Для применения генерационного фазинга требуется формальное описание структур данных, которое не всегда возможно выполнить. Ряд исследователей отдают предпочтение генерационному фазингу. В работе [2] авторы приводят результаты исследований, которые показывают преимущество генерационного фазинга перед мутационным. На примере формата файлов *png* показывается, что область покрытия тестируемого кода существенно больше при

[1]Работа выполнена при поддержке гранта МД № 2.2010.10.