УДК 004.415.532.3

***БЕДЕРДИНОВА Оксана Ивановна**, кандидат технических наук, доцент кафедры информатики института судостроения и морской арктической техники (Севмашвтуз) филиала САФУ имени М.В. Ломоносова в г. Северодвинске. Автор 51 научной публикации*

***ИВАНОВА Людмила Александровна**, программист ООО «ИТДефенсор» (г. Москва)*

СОВЕРШЕНСТВОВАНИЕ МЕТОДА ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ «БЕЛЫЙ ЯЩИК»

На основе анализа процесса тестирования программного обеспечения методом «Белый ящик» разработана функциональная модель по нотации IDEF0, включающая три стадии: предварительная подготовка к тестированию, тестирование программного обеспечения и формирование отчетной документации по полученным результатам. Проведены исследования и проанализированы результаты влияния специализированного программного обеспечения на результаты тестирования при динамическом способе проверки программного кода. Предложены направления совершенствования метода «Белый ящик» для статического тестирования на примере статического анализатора «Cppcheck», заключающиеся в создании расширений для обнаружения использования опасных функций и проверки возвращаемого значения функций; обнаружения использования небезопасных генераторов случайных чисел, неспособных противостоять криптографическим атакам, двойного освобождения одного и того же блока памяти и использования кода с проблемами переносимости, использования памяти после ее освобождения, приводящего к нестабильности программного обеспечения в процессе работы.

***Ключевые слова:** тестирование программного обеспечения, метод «Белый ящик», метод статического анализа, метод динамического анализа.*

Цель любого метода тестирования безопасности программного обеспечения состоит в обеспечении надежности системы в условиях вредоносных атак и программных дефектов и сбоев. На производстве уязвимость программного обеспечения может привести к финансовым и временным потерям из-за остановки или некорректной работы системы, поэтому заблаговременное ее выявление является актуальной задачей для всех предприятий.

При анализе типовой схемы атаки с использованием уязвимостей (*рис. 1*) видно, что несвоевременно обнаруженная уязвимость может привести к реализации угроз конфиденциальности, целостности и доступности информации, обрабатываемой на автоматизированных рабочих местах. Успешно реализованная угроза при широкой огласке инцидента может привести к потере доверия к компании и потере репутации.

Рис. 1. Схема реализации атаки: АРМ – автоматизированное рабочее место, ПО – программное обеспечение

В настоящее время существуют разнообразные методы выявления дефектов, ошибок и уязвимостей программного обеспечения (ПО). Они имеют определенные достоинства и недостатки, различные области применения, что влияет на эффективность и конечный результат верификации.

Метод тестирования «Белый ящик» является наиболее распространенным. Его использование приводит к понижению рисков успешных вредоносных атак, повышая общую безопасность системы. Целью нашей работы является анализ и совершенствование этого метода.

Для применения метода «Белый ящик» требуется наличие полной информации об исследуемом ПО (исходный код, информация о результатах проведенных тестирований и т. д.), что позволяет провести его полный анализ на предмет дефектов, ошибок и уязвимостей. Метод тестирования может быть реализован динамическим и статическим способами.

При статическом анализе исследуются исходные коды компонентов и документированные возможности ПО, а при динамическом осуществляется проверка поведения ПО в реальных условиях с применением специализированного программного обеспечения (отладчиков, профилировщиков).

Функциональная модель процесса тестирования методом «Белый ящик» в соответствии с нотацией IDEF0 представлена на *рис. 2–5*.

Входными данными для процесса тестирования являются: исходный код и бинарные файлы ПО; документированные возможности программного и технического обеспечений; документация по предыдущим тестированиям ПО.

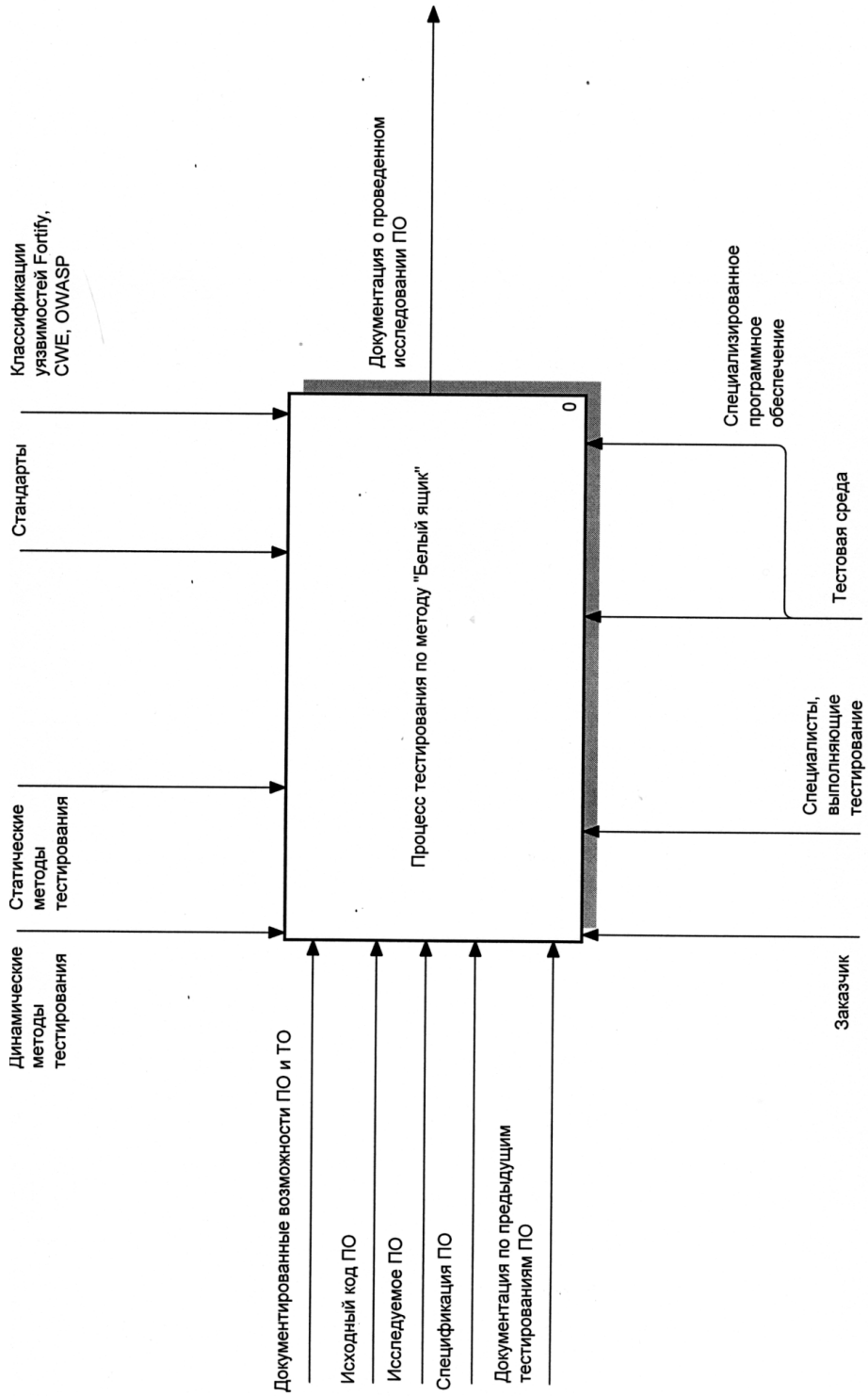

Рис. 2. Контекстная диаграмма процесса тестирования программного обеспечения методом «Белый ящик»

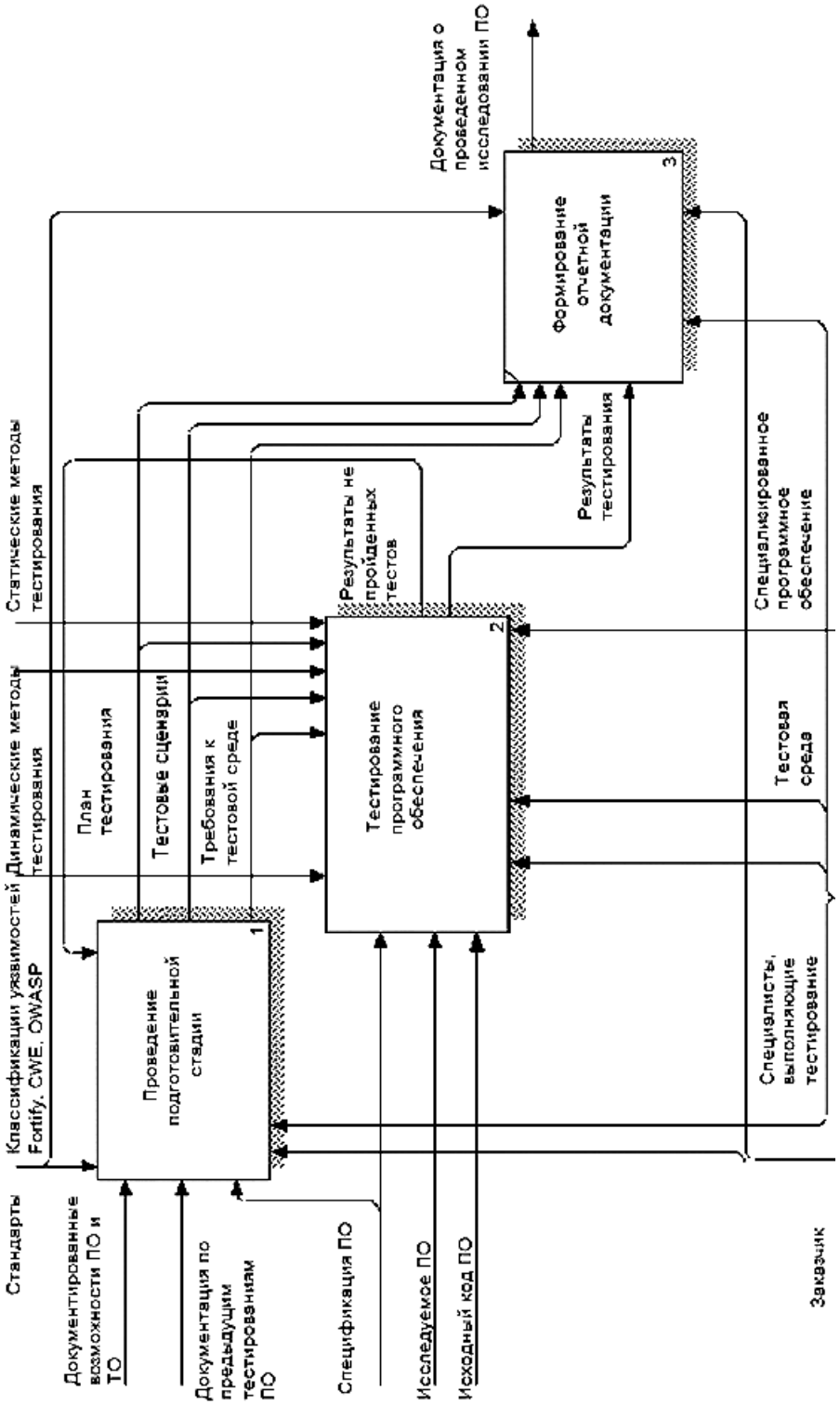

Рис. 3. Декомпозиционная диаграмма модели

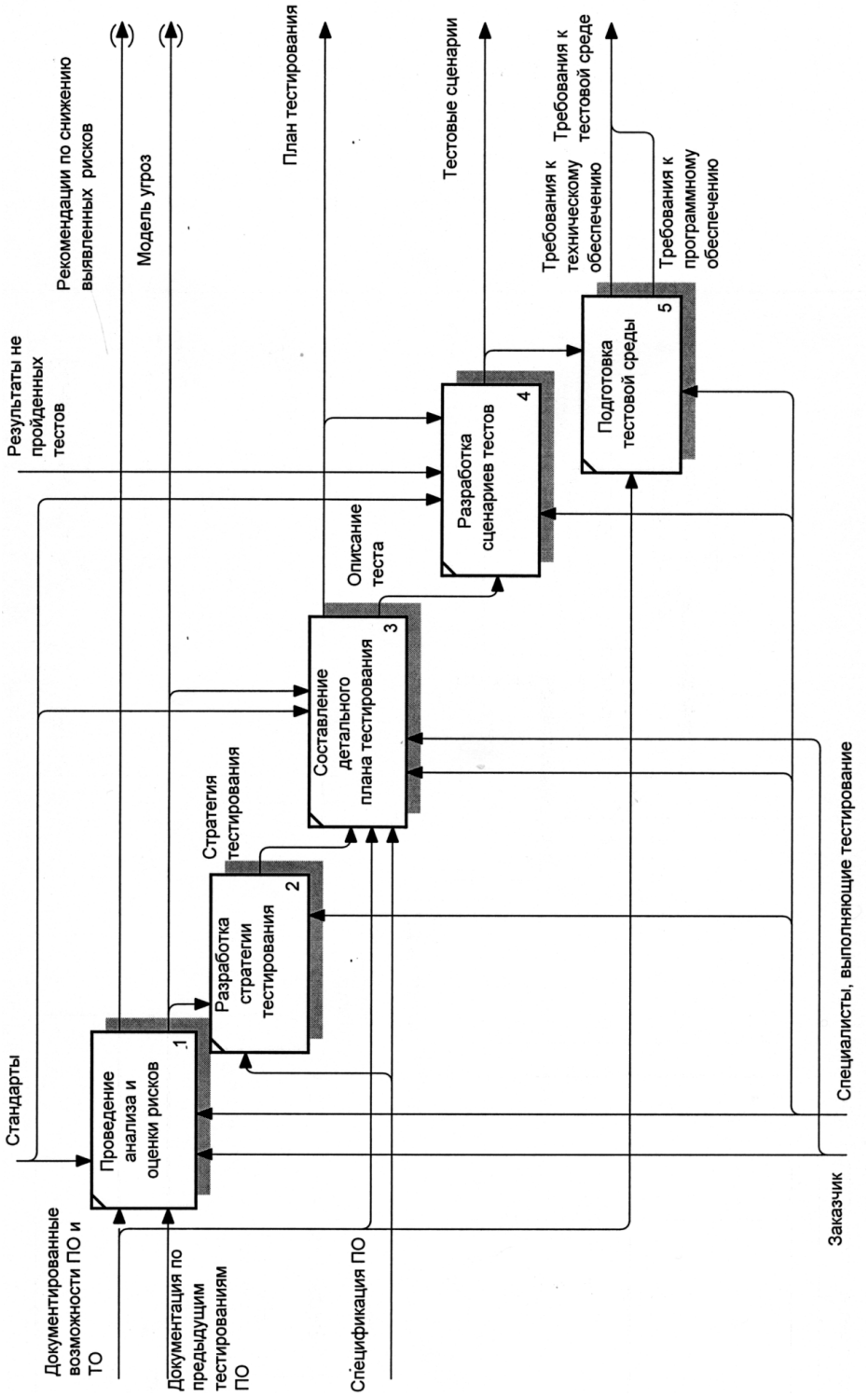

Рис. 4. Декомпозиционная диаграмма процесса «Проведение подготовительной стадии»

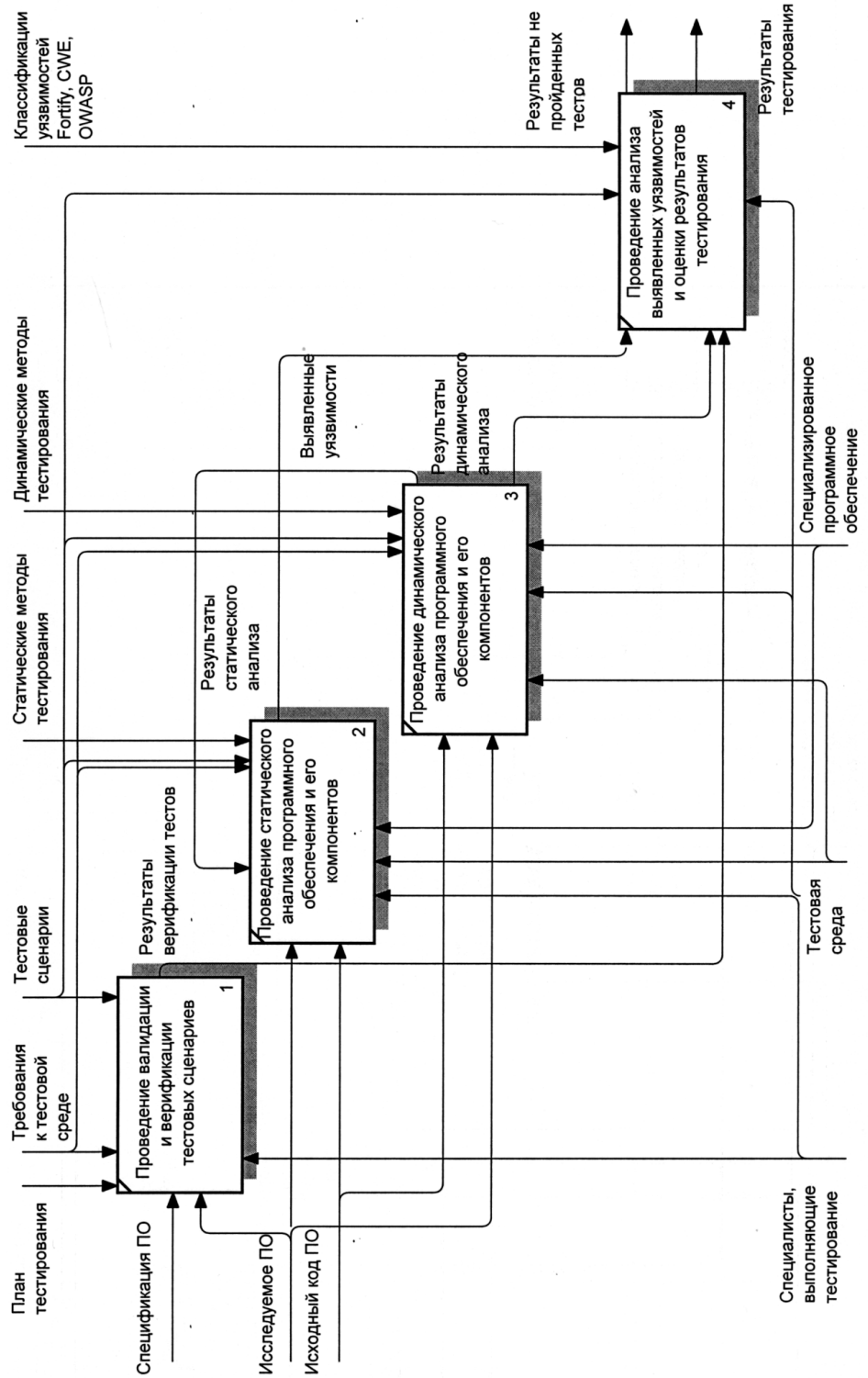

Рис. 5. Декомпозиционная диаграмма процесса «Тестирование программного обеспечения»

Документированные (декларированные) возможности программного и технического обеспечений включают сведения о структуре, взаимодействии компонентов и условиях эксплуатации программного обеспечения, о конфигурации, характеристиках и условиях эксплуатации технического обеспечения, а также спецификации и требования на все ПО и его компоненты.

Документация по предыдущим тестированиям содержит информацию, описывающую планы и результаты проведенных тестов, выявленные дефекты и рекомендации по устранению уязвимостей.

В результате проведения тестирования ПО формируется документация о результатах проведения исследования.

В качестве исполнителей выступают заказчик и специалисты, выполняющие тестирование. Механизмами исполнения являются тестовая среда и специализированное ПО, обеспечивающее проведение исследования.

Управляющими воздействиями являются стандарты, международные классификации уязвимостей OWASP, «CWE MITRE Corporation», CWE и «Fortify» и методы тестирования.

Классификация OWASP [1] используется для web-приложений. «CWE MITRE Corporation» [2], CWE [3] и «Fortify» [4] применяются для идентификации программных ошибок вне зависимости от сферы применения ПО. Например, по классификации «Fortify» программные ошибки делятся на семь основных групп: ошибки проверки входных данных и связанные с представлением данных; ошибки злоупотребления API (прикладным программным интерфейсом); ошибки в реализации защиты и программного кода; ошибки, обусловленные многопоточностью приложений и низким качеством кода; ошибки, связанные с недостаточной изоляцией программных компонентов друг от друга.

Метод тестирования программного кода «Белый ящик» включает следующие основные стадии и этапы.

1. Подготовительная стадия (планирования) состоит из этапов:

1.1. Проведение анализа и оценки рисков (составление модели угроз);

1.2. Разработка стратегии тестирования;

1.3. Разработка детального плана тестирования;

1.4. Разработка сценариев тестов и определения области покрытия каждого теста;

1.5. Подготовка тестовой среды.

2. Стадия проведения тестирования включает:

2.1. Проведение валидации и верификации тестовых сценариев;

2.2. Проведение статического анализа программного обеспечения и его компонентов;

2.3. Проведение динамического анализа программного обеспечения и его компонентов;

2.4. Проведение анализа выявленных ошибок, дефектов и уязвимостей программного обеспечения.

3. Стадия формирования отчетной документации представляет собой создание отчетной документации о результатах проведения исследований.

Этап анализа и оценки рисков включает анализ угроз и видов уязвимостей, присутствующих в каждом компоненте ПО, вероятности их реализации и оценку возможного ущерба предприятия при возникновении рисков (стоимость ошибки ПО), выработку рекомендаций для снижения выявленных рисков.

Этап разработки стратегии тестов производится на основе результатов анализа рисков с целью определения основных мероприятий и задач тестирования наиболее опасных фрагментов кода, т. е. области и методов тестирования, объема проверяемого кода, требований к архитектуре тестовой среды и квалификации специалистов, выполняющих исследования.

На этапе разработки детального плана тестирования определяются и согласовываются с заказчиком цели, график тестирования и степень покрытия тестами программы и ее компонентов, способы контроля и требования к отчетной документации.

На этапе создания тестов определяются сценарии тестов, описывающие критерии покрытия программного кода с учетом целей тестирования; начальные условия, входные данные, ожидаемые результаты и контрольные точки проведения тестов.

Этап подготовки тестовой среды включает настройку конфигураций двух составляющих – технического и программного обеспечений, удовлетворяющих требуемым условиям эксплуатации. При несоответствии среды тестирования реальным условиям эксплуатации программного и технического обеспечений невозможно корректно оценить риски и выявить максимально возможное число дефектов и уязвимостей ПО.

В качестве технического обеспечения может использоваться отдельная ПЭВМ или вычислительная сеть (ЛВС, муниципальная, глобальная), принтеры, сканеры и специализированное оборудование, с которым взаимодействует исследуемое ПО.

В состав программного обеспечения входят: специализированные программы для тестирования ПО (профилировщики, отладчики, анализаторы памяти, программное обеспечение для генерации ошибочных входных данных), пакеты прикладных программ, непосредственно взаимодействующие с исследуемым ПО в реальных условиях эксплуатации (операционная система, драйверы устройств и т. д.).

Стадия проведения тестирования ПО заключается в выявлении дефектов и ошибок в тестируемых компонентах ПО и подразделяется на этапы валидации и верификации, статического, динамического анализа и анализа результатов проведения тестовых сценариев.

Этап валидации и верификации тестовых сценариев включает инспекцию тестов на соответствие детальному плану и стандартам и анализ полученных результатов.

На этапе проведения статического анализа производится поиск ошибок без исполнения тестируемого ПО с помощью анализа исходного кода проверяемых компонентов с использованием специализированного программного обеспечения.

При динамическом анализе осуществляется проверка реального поведения исследуемого ПО в рамках определенных сценариев его работы с использованием специализированного программного обеспечения.

Этап анализа результатов тестирования включает диагностику и локализацию выявленных ошибок и дефектов в тестируемом ПО, анализ достигнутого тестового покрытия, принятие решения о создании дополнительных тестовых сценариев (в случае выявления не пройденных тестов) или о прекращении тестирования.

В качестве специализированного программного обеспечения, реализующего тестирование по методу «Белый ящик», применяются инструментальные средства для анализа исходного кода и поведения ПО (отладчики, анализаторы памяти, декомпиляторы или дизассемблеры), а также профилировщики (в основном для нахождения атак типа отказа в обслуживании).

Результатами тестирования безопасности ПО являются результаты ее проверки на соответствие требованиям и спецификациям системы, т. е. решение о том, является ли объект исследования качественным и надежным, а также устойчивым к атакам злоумышленников.

На стадии формирования отчетной документации составляется отчет о результатах тестирования ПО, который включает описание сценариев проведенных тестов, тестовые контрольные данные, описание архитектуры тестовой среды, степени покрытия программ сценариями тестов и статуса полученных ошибок и дефектов, информацию о документировании процесса тестирования, список задействованных в нем специалистов. Для каждой выявленной уязвимости указываются тестовый сценарий, условия ее воспроизведения и методы устранения.

В процессе тестирования программного обеспечения методом «Белый ящик» на результат тестирования оказывает влияние тестовая среда, что приводит к снижению результативности и эффективности исследования. Поэтому при настройке тестовой среды необходимо также учитывать, что специализированное программ-

ное обеспечение для тестирования не должно оказывать существенного влияния на работу исследуемого программного обеспечения.

Тестирование ПО проведено динамическим способом с профилировщиком и без него для определения степени влияния тестовой среды на результаты тестирования разных видов ПО.

Профилировщик предназначен для оценки производительности ПО, а также решения следующих задач: замеров производительности ПО; обнаружения утечек памяти, фрагментации динамической памяти, взаимоблокировок, неверного управления ресурсами и мест в ПО, потенциально содержащих уязвимости.

Для проведения динамического тестирования с целью определения влияния профилировщика и последующего анализа эффективности метода «Белый ящик» на разных видах приложений для различных архитектур тестовой среды созданы тесты на языке С в среде разработки «Microsoft Visual Studio 2008». Тестирование проведено для следующих видов прикладного ПО: с наличием функций работы с файлами, с принтером, со звуком, с графикой и сетевых функций передачи данных.

Для тестирования использовались два ПЭВМ, сервер и принтер. Показатели скорости выполнения функций определены для следующих архитектур тестовой среды:

– конфигурация ПЭВМ 1: операционная система – «Microsoft Windows 7 Домашняя расширенная» 64 бита, процессор – «Intel64 Family 6 Model 42 Stepping 7 GenuineIntel» ~ 792 МГц, ОЗУ – 8 Гб;

– конфигурация ПЭВМ 2: операционная система – «Microsoft Windows 7 Домашняя расширенная» 64 бита, процессор – «AMD64 Family 15 Model 104 Stepping 2 AuthenticAMD» ~1200 МГц, ОЗУ – 4 Гб;

– конфигурация тестового сервера: операционная система – «Microsoft Windows Server 2008 R2 Standard», процессор – «Intel64 Family 6 Model 45 Stepping 7 GenuineIntel» ~1180 Mhz, ОЗУ – 32 Гб, скорость передачи данных – 100 Мбит/с.

Для тестирования функций работы с принтером применялся принтер «Samsung SCX-3200». Объем обрабатываемых данных – 828 Кбайт с ftp сервера.

Для определения продолжительности выполнения функций передачи данных и воспроизведения звука использовались функции GetTickCount Windows API и PlaySound соответственно.

Исследования влияния профилировщика производились на примере «Intel VTune Amplifier XE 2013» для ПЭВМ 1 и «AMD CodeAnalyst 3.8» для ПЭВМ 2.

Определенные средние значения времени выполнения функций обработки информации приведены в *таблице*.

В результате анализа полученных результатов выявлено, что влияние профилировщика

ЗНАЧЕНИЯ ПРОДОЛЖИТЕЛЬНОСТИ ВЫПОЛНЕНИЯ ФУНКЦИЙ ОБРАБОТКИ ИНФОРМАЦИИ, мс

Функции	Для ПЭВМ 1		Для ПЭВМ 2	
	без профилировщика	с профилировщиком	без профилировщика	с профилировщиком
Передача данных по сети объемом 828 Кбайт	1206,2	1257,7	2378,7	2391,8
Воспроизведение звукового файла	951,6	971,2	1116,9	1120,0
Отображение одного окна в системе «Windows»	57,4	57,3	178,1	221,8
Печать 10 страниц данных	925,1	926,6	482,1	549,1
Запись данных в файл	3130,6	3993,0	251,8	273,5

на результаты функционирования тестируемого ПО зависит от его вида. Для ПО с функциями передачи данных, воспроизведения звука, печати данных, записи данных в файл время выполнения функций обработки информации с профилировщиком больше, чем без профилировщика на 4,3, 2,1, 0,2 и 27 % для конфигурации ПЭВМ 1 и на 0,5, 0,3, 13,8 и 8,6 % для конфигурации ПЭВМ 2 соответственно. При отображении графического окна разница составляет 0,2 и 24 %, причем скорость под профилировщиком выше.

На основании вышеизложенного можно сделать вывод, что использование профилировщика при проведении тестирования замедляет (ускоряет) выполнение функций обработки данных в диапазоне от 0,2 до 27 %.

В результате проведенных исследований доказано влияние профилировщика на тестируемое ПО и, как следствие, на результаты процесса тестирования. Это необходимо учитывать при замере производительности посредством профилировщика и отладчика для соответствующих типов ПО.

При статическом тестировании используются автоматические анализаторы и навигаторы программного кода. Данный вид специализированного программного обеспечения оказывает влияние на результаты тестирования за счет антропогенного фактора. Наибольшее влияние оказывают автоматические анализаторы исходного кода на известные уязвимости и дефекты, т. к. полученный результат напрямую зависит от базы известных ошибок. Характерной особенностью статических анализаторов является анализ исходного кода программ без их запуска, а также то, что анализируемый код должен быть написан только на определенном языке программирования.

Статический анализатор «Cppcheck» предназначен для анализа кода приложений, написанных на языках C и C++. Он выявляет ошибки переполнения буфера и целочисленного переполнения, ошибки использования функций и низкое качество кода, включающее ошибки утечки памяти, использование неинициализированных переменных и устаревших функций.

Таким образом, направлениями совершенствования специализированного программного обеспечения для статического тестирования (на примере статического анализатора «Cppcheck») являются создание расширений для следующих задач:

– обнаружение использования опасных функций и проверки возвращаемого значения функций;

– обнаружение использования небезопасных генераторов случайных чисел, неспособных противостоять криптографическим атакам;

– обнаружение двойного освобождения одного и того же блока памяти и использования кода с проблемами переносимости;

– обнаружение использования памяти после ее освобождения, приводящее к нестабильности ПО в процессе работы.

Для определенных задач были разработаны расширения функциональных возможностей анализатора «Cppcheck» на процедурном языке C. В результате проведения тестирования исходного кода первоначальной и расширенной версиями «Cppcheck» установлено, что усовершенствованная версия анализатора выявляет до 50 % ошибок кода больше.

Таким образом, повышение эффективности процесса тестирования методом «Белый ящик» возможно за счет минимизации влияния тестовой среды и антропогенного фактора путем совершенствования специализированных программных средств тестирования, т. е. создания расширений для автоматизации поиска сигнатур ошибок.

Список литературы

1. Category: OWASP Top 10 Project. URL: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project#tab=OWASP_Top_10_for_2010 (дата обращения: 19.05.2014).
2. CWE Version 2.3 / ed. S. M. Christey, J. E. Kenderdine and J. M. Mazella; project lead: Robert A. Martin. 2012. URL: https://cwe.mitre.org/data/published/cwe_v2.3.pdf (дата обращения: 19.05.2014).
3. 2011 CWE/SANS Top 25 Most Dangerous Software Errors. 2011. URL: http://cwe.mitre.org/top25/ (дата обращения: 19.05.2014).
4. Seven Pernicious Kingdoms: A Taxonomy of Software Security Errors / Fortify Software. URL: http://www.hpenterprisesecurity.com/vulncat/en/docs/Fortify_TaxonomyofSoftwareSecurityErrors.pdf (дата обращения: 19.05.2014).

References

1. *Category: OWASP Top 10 Project*. Available at: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project#tab=OWASP_Top_10_for_2010 (accessed 19 May 2014).
2. *CWE Version 2.3*. Ed. by Christey S.M., Kenderdine J.E., Mazella J.M. Project lead: Martin R.A. 2012. Available at: https://cwe.mitre.org/data/published/cwe_v2.3.pdf (accessed 19 May 2014).
3. *2011 CWE/SANS Top 25 Most Dangerous Software Errors*. 2011. Available at: http://cwe.mitre.org/top25/ (accessed 19 May 2014).
4. *Seven Pernicious Kingdoms: A Taxonomy of Software Security Errors*. Fortify Software. Available at: http://www.hpenterprisesecurity.com/vulncat/en/docs/Fortify_TaxonomyofSoftwareSecurityErrors.pdf (accessed 19 May 2014).

Bederdinova Oksana Ivanovna
Institute of Shipbuilding and Arctic Marine Engineering, Severodvinsk Branch of Northern (Arctic) Federal University named after M.V. Lomonosov (Arkhangelsk, Russia)

Ivanova Lyudmila Aleksandrovna
ITDefensor LLC (Moscow, Russia)

IMPROVEMENT OF WHITE-BOX SOFTWARE TESTING METHOD

Having analyzed the process of white-box software testing, we developed a functional model using IDEF0 notation, which includes three stages: preparation to testing, software testing, and making report documentation of the results. We studied and analyzed the effect of specialized software on the test results at software code validation. Further, we suggested ways of improving the white-box method for static testing exemplified by Cppcheck static code analysis tool. This can be achieved by creating extensions to detect hazardous functions and test the return value of functions; identify use of insecure random number generators unable to resist cryptographic attacks, double deallocation of the same memory block and use of code with transferability problems, as well as use of memory after its deallocation, which leads to software instability in operation.

Keywords: *software testing, white-box testing, static analysis, dynamic analysis.*

Контактная информация:
Бедердинова Оксана Ивановна
адрес: 164500, г. Северодвинск, ул. Капитана Воронина, д. 6;
e-mail: O.Bederdinova@narfu.ru

Иванова Людмила Александровна
адрес: 109316, Москва, Остаповский проезд, д. 5, оф. 114;
e-mail: ivanova_la2000@bk.ru

Рецензент – *Кремлева Л.В.*, доктор технических наук, профессор, заместитель директора по учебной работе института судостроения и морской арктической техники (Севмашвтуз) филиала САФУ имени М.В. Ломоносова в г. Северодвинске