УДК 004.4'2

ПРОВЕРКА КОДА НА УЯЗВИМОСТИ НА ВСЕХ СТАДИЯХ РАЗРАБОТКИ

Скрыпников А.В., Денисенко В.В., Высоцкая И.А., Савченко И.И., Евтеева К.С.

ФГБОУ ВО «Воронежский государственный университет инженерных технологий», Воронеж, e-mail: skrypnikovvsafe@mail.ru, v.denisenko1@yandex.ru, i.a.trishina@gmail.com, ilona.savchenko.2016@mail.ru, evteeva2020@inbox.ru

В работе рассматриваются и анализируются современные инструментальные средства проверки кода программ на неисправности и уязвимости. Необходимость такого исследования обусловлена тем, что использование любого программного обеспечения должно осуществляться под контролем службы информационной безопасности предприятия. Так, например, ошибки в программном коде могут привести к сбою отлаженных бизнес-процессов организации и привести к уничтожению или повреждению базы данных. Заказчикам программного продукта стоит производить тестирование программного обеспечения, которое приобретается и вводится в эксплуатацию, независимо от разработчиков. Создание нового программного обеспечения, гарантированно устойчивого к вредоносным воздействиям, сопряжено с существенными затратами как времени, так и материальных ресурсов. Всестороннее и глубокое рассмотрение методов и средств проверки кода программ на неисправности и уязвимости позволяет выбрать оптимальный подход к тестированию нового программного продукта. Для обеспечения корректного тестирования программного обеспечения сформулированы нормы инструментальных средств статистического анализа кода, приводится систематизация обнаруженных уязвимостей, даются рекомендации о содержании отчета после выполнения анализа кода. Инструментальные средства в статическом анализе кода, соответствующие всем нормам, точнее определяют неисправности кода и предоставляют возможности затрачивать меньше ресурсов на тестирование, разработку и устранение ошибок.

Ключевые слова: разработка программного обеспечения, уязвимость, тестирование, инструментальные средства статистического анализа кода, информационная безопасность

CHECKING CODE FOR VULNERABILITIES AT ALL STAGES OF DEVELOPMENT

Skrypnikov A.V., Denisenko V.V., Vysotskaya I.A., Savchenko I.I., Evteeva K.S.

Voronezh State University of Engineering Technologies, Voronezh, e-mail: skrypnikovvsafe@mail.ru, v.denisenko1@yandex.ru, i.a.trishina@gmail.com, ilona.savchenko.2016@mail.ru, evteeva2020@inbox.ru

The paper considers and analyzes modern tools for checking program code for faults and vulnerabilities. The need for such a study is due to the fact that the use of any software should be carried out under the control of the information security service of the enterprise. So, for example, errors in the program code can lead to the failure of the well-functioning business processes of the organization, and lead to the destruction or damage of the database. Customers of a software product should test software that is purchased and put into operation, regardless of the developers. Creation of new software that is guaranteed to be resistant to harmful influences is associated with significant expenditures of both time and material resources. A comprehensive and in-depth consideration of methods and tools for checking program code for faults and vulnerabilities allows you to choose the optimal approach to testing a new software product. To ensure correct software testing, the norms of tools for statistical analysis of the code are formulated, the systematization of the detected vulnerabilities is given, and recommendations are given on the content of the report after the analysis of the code. Code static analysis tools that comply with all standards more accurately identify code faults and provide opportunities to spend less resources on testing, development and troubleshooting.

Keywords: software development, vulnerability, testing, statistical code analysis tools, information security

В современном бизнесе элементы интернет-технологий и автоматизации являются одними из основных составляющих. Потребности в необходимом программном обеспечении сильно опережают возможности предоставления рынком всех необходимых для этого ресурсов. Очень часто при передаче готового программного продукта в использование заказчику проверка кода по стандартам информационной безопасности проходит некорректно, что может привести к весомым потерям.

В настоящее время существует ряд факторов, которые приводят к скачкообразному росту убытков, связанных с кибербезопасностью. По данным ВЭФ и Сбербанка, мировые убытки в области кибербезопасности в 2020 г. приближаются к отметке в 3 трлн долл. Официальный ежегодный отчет Cybersecurity Ventures о киберпреступности прогнозирует двукратный рост расходов в 2021 г., что кажется весьма реалистичным, если учесть последствия пандемии и связанных с ней изменений в области кибербезопасности, вызванных массовым переходом на удаленный режим работы.

Целью работы является анализ современных инструментальных средств проверки кода программ на неисправности и уязвимости. Сформированы нормы для инструментальных средств статистического анализа кода, приведена систематиче-

зация обнаруженных уязвимостей, даются рекомендации о содержании отчета после выполнения анализа кода.

Инструментальные средства статистической проверки кода

Новые структуры сделаны из большого количества компонентов, которые связаны сложными и непрозрачными связями. Такие связи тяжело удерживаются во внимании, потому что не всегда используются необходимые инструментальные средства (ИС), делающие визуализацию потоков информации, а также передачу управления с одного элемента на другой во время работы программы. Самые популярные уязвимости, которыми чаще всего пользуются злоумышленники, находятся в стыках различных элементов (рисунок).

Следовательно, полный анализ и тестирование кода по всем нормам информационной безопасности (ИБ) предоставляет возможность улучшить программный продукт [1]. Проверка кода по нормам безопасности является важнейшим этапом уже при проектировании. По исследованиям учёных Гарвардского университета, ошибки, выявляющиеся на этапах тестов, в среднем стоят 960 долл., а неисправности, найденные на этапе использования, стоят 7600 долл. И это не включая фактор того, какие проблемы может принести злоумышленник, используя уязвимости использования [2]. В связи с этим за последние годы спрос на разработку программного обеспечения значительно повысился, что стало причиной нехватки опытных разработчиков. Так, например, ошибки в программном коде могут привести к сбою отлаженных бизнес-процессов организации и привести к уничтожению или повреждению базы данных [3].

Заказчикам программного продукта стоит производить тестирование программного обеспечения, которое приобретается и вводится в эксплуатацию, наравне с разработчиками. Такой подход позволяет, узнав о неисправностях кода, применить настраиваемую защиту периметра, до момента корректировки программы.

IT-индустрия сейчас предоставляет огромное количество инструментальных средств, которые активно используются для разработки программных продуктов. Однако использование продуктов от разных производителей имеет недостатки: так, чтобы ввести в использование программный продукт, нужно предварительно настроить ПО, обучить персонал пользоваться им.

Чтобы контролировать качество разработки, создано много сценариев проведения тестов, отвечающих всем требованиям. Данные тесты используются в ПО автоматически или полуавтоматически. Это же касается проверки кода по нормам ИБ. На данный момент рынок имеет множество различных инструментальных средств (коммерческих и распространяемых) для проверки кода по нормам ИБ. Многие из этих средств применяются в разработке ПО, а также для выпущенного в работу программного обеспечения [4, 5].

Для инструментальных средств характерны следующие параметры:

а) ложные срабатывания – часть кода, которая выявлена как ошибочная, но такой не является. Ложные срабатывания бывают полезны тем, что анализ результатов проводится тщательно, однако возникают лишние затраты по времени;

б) пропущенные ошибки – неисправности в коде, которые не были обнаружены при анализе.

Схема процесса тестирования

Инструментальные средства статистической проверки кода, которые находятся в свободном распространении, представляют собой базу статистического анализа программы. Самая главная их ценность заключается в библиотеке правил, по которым определяются ошибки. Данные библиотеки разрабатываются большим количеством специалистов в сфере ИБ. Правила должны быть полноценными и точными, чтобы избегать ложных ошибок и прочих неполадок в анализе [6].

Также инструментальные средства проверки кода программы должны проводить анализ и внутренних частей кода, чтобы выявить все особенности разработки. Ошибки, которые не относятся к недокументированным возможностям (НДВ), считаются побочными дефектами в работе кода, предназначенного для функционирования структуры. Большую часть неисправностей, произведенных злоумышленниками, нельзя определить, проанализировав всего лишь текст программного продукта. Но при этом обнаружить НДВ можно, анализируя только исходный текст программного продукта, применяя специальную библиотеку примеров.

Для инструментальных средств статистического анализа кода, цель которых – определение неисправностей, можно сформулировать следующие нормы:

– использование эффективных технологий и способов для проведения глубокого анализа кода и определения всех неисправностей;

– обновление баз правил должно быть регулярным и гибким в настройках;

– постоянное предоставление информации по неисправностям и по их устранению;

– сравнение итогов анализа с повторным тестированием кода;

– поддерживание разнообразных языков программирования;

– наличие концепции контроля версий и наблюдения погрешностей;

– наличие функционала, гарантирующего взаимосвязь между нормами создателей и специалистов, отвечающих за защищенность;

– минимальное число ошибочных срабатываний;

– представление итогов работы в подходящем для восприятия образе;

– присутствие средств механического формирования сведений;

– возможность осуществлять исследование кодировки удалённо.

Инструментальные средства в статическом анализе кода, соответствующие всем нормам, точнее определяют неисправности кода и предоставляют возможности затрачивать меньше ресурсов на тестирование, разработку и устранение ошибок.

Интерфейс пользователя тоже является важной составляющей. Без качественного предоставления информации, визуализации и обобщения анализ будет терять эффективность.

Результаты исследования и их обсуждение

Компания HP предоставляет несколько различных программ для анализа ПО на соответствие нормам ИБ:

● HP Fortify Static Code Analyzer (SCA) – средство для проведения статистического анализа кода. Находит причину возникновения ошибки, обрабатывает результат и выдаёт способ по устранению ошибки в коде. Статистический анализ предоставлен для 21-го языка программирования.

● HP Интернет Inspect – механизм с целью динамического рассмотрения программного кода. Выполняет испытание дополнений, моделирующее настоящие атаки. Владеет механизмами управления ситуацией теста.

● HP Fortify SCA показывает высокое качество в сканировании. Этот вариант считается сбалансированным в отношении «время работы к количеству ложных срабатываний и пропущенных ошибок».

Также можно выделить HP WebInspect Real-Time, HP WebInspect Enterprise, HP Fortify Runtime Analyzer, HP Fortify SSC.

Решения, которые предлагает фирма IBM для проверки ПО на соответствующие требования, выглядят следующим образом:

● IBM AppScan Standard – механизм динамического рассмотрения программного кода («чёрный ящик»). Специализирован на розыске уязвимостей в функционирующем ПО, используется в прошедших стадиях исследования также уже после выпуска дополнения. Сравнительно прост в установке и настройке. Дает возможность отсылать информацию об обнаруженных уязвимостях в концепцию наблюдения погрешностей. В присутствии поддержки расширения JavaScript Security Analyzer способен осуществлять смешанное исследование JavaScript-программный код.

● IBM AppScan Source – механизм статического рассмотрения кодировки («белый ящик»). Он ориентирован на экспертов по информационной безопасности, требует определённой квалификации, создает объективную картину уязвимостей с привязкой к начальному коду. Гарантирует связь между работниками, отвечающими за информационную безопасность, и разработчиками.

Функционирует только лишь как приложение к IBM AppScan Enterprise и поддерживает 21 язык программирования.

● IBM AppScan Enterprise – интернет-механизм концентрированного динамического испытания программного кода и учёта уровня риска, которому он подвергается. Ориентирован в том числе на специалистов низкого уровня квалификации и дает возможность составлять отчеты по итогам распознавания. При присутствии AppScan Source способен осуществлять испытание согласно способу «прозрачного ящика» (Glass Box), сравнивая итоги динамического и статического рассмотрения программного кода.

Обнаруженные уязвимости могут быть систематизированы согласно последующим показателям:

– степень риска (большой, умеренный либо небольшой);

– вид уязвимости;

– документ, в каком месте незащищенность была найдена;

– интерфейс приложения (акцентируются опасный вызов API-функции, также переданные ей доводы);

– состояние (номера строчки и столбика в файле с начальным кодом, в каком месте находится опасный вызов);

– систематизация (незащищенность либо редкий случай).

Не все без исключения итоги поиска считаются уязвимостями. Например, AppScan дает последующий способ их систематизации:

1. Неисправность – место исходного кода, содержащее погрешности, позволяющие правонарушителю заставить приложение реализовывать внезапные действия, что приводит к неразрешенному доступу к сведениям, дефекту концепции и т.д.

2. Исключение вида I – опасное место исходного кода, скорее всего, представляющееся уязвимостью, при этом для отнесения его к классу с недостатком данных. К примеру, уровень подверженности нападению способен находиться в зависимости от характеристик применения динамических компонентов либо призыва библиотечных функций, о которых у AppScan мало данных.

3. Исключение вида II – опасное место исходного кода, для которого нереально оценить потенциал подверженности нападению.

С целью предоставления абсолютной защищенности дополнения необходимо осуществлять вспомогательные изучения для отнесения исключений или к уязвимостям, или к ошибочным срабатываниям, при этом обрабатывание исключений вида II требует больше усилий. В последующем с целью лаконичности станем именовать итоги распознавания AppScan «уязвимостями», осознавая под этим непосредственно уязвимости также исключения I и II видов.

Инструментальные средства позиционируются как способ контроля информативной защищенности, при этом возможность интеграции с иными концепциями управления проектом не предусматривается. Кроме этого, в настоящее время Application Inspector дает постоянный анализ только лишь с целью интерпретируемых стилей, но изготовитель свидетельствует о том, что помощь компилируемых стилей также станет присутствовать.

В настоящий период комплект используемых правил с целью поиска уязвимостей довольно небогат, но проводится постоянная деятельность по наращиванию основы правил, что дает возможность верить в успешность плана выхода данного продукта в фавориты рынка в ближайшей перспективе.

Еще в одном инструментальном средстве статического анализа кодировки Veracode статический анализ доступен только лишь как услуга. Код, который следует обследовать, необходимо загрузить в сервер Veracode, специалисты осуществят его распознавание и отправят отчет. Данный алгоритм работы не всегда является оптимальным и удобным, в особенности в банковской области, где к коду предъявляются высокие условия. Загрузка кодировки в посторонние серверы, тем более находящиеся не в Российской Федерации, очень ограничивает способности эксплуатации продукта.

На рынке услуг активно представлено инструментальное средство анализа программы Checkmarx. Результат работы программы демонстрирует качественные итоги распознавания, но отметим, что его основа уязвимостей менее абсолютная, нежели у соперников, в частности у HP Fortify. Сопоставление Checkmarx с HP Fortify SCA в пяти независимых проектах выявило, что при сравнимом времени работы средств число обнаруженных реальных уязвимостей у HP Fortify было больше. Соответственно, Checkmarx упускает уязвимости, которые распознаются автоматом.

Широко используется еще одно ИС – InfoWatch APPERCUT. APPERCUT осуществляет исследование по текстовому представлению программы. Однако все без исключения уязвимости, которые обнаруживает механизм APPERCUT при анализе проектов C/C++, можно выявить MVS (Microsoft Visual Studio) 2012 при включении функции абсолютной проверки.

Все уязвимости, найденные APPERCUT для Java-проектов, обнаруживает среда разработки Eclipse с плагином Find Bugs.

После выполнения правильного анализа кода, компании должен выдаваться отчёт, содержащий действительные неисправности, распределённые по уровням опасности. Для всех ошибок необходимо:

– дать полную информацию об опасности, которую может принести злоумышленник при использовании данной ошибки;

– описать пример использования данной ошибки злоумышленником, а также то, какой он должен обладать квалификацией для этого;

– предоставить информацию насчет решения данных проблем с учетом расходов.

После чего фирма будет знать все данные о коде и сможет принять решение по поводу его использования.

Заключение

Анализ современных инструментальных средств проверки кода программы показал, что даже при условии, когда инструментальные средства дают всю необходимую информацию о неисправностях, они всё равно должны быть проанализированы экспертами. Дополнительная проверка специалистом необходима для проведения качественного анализа кода. Иначе некоторые неисправности могут остаться незамеченными, что приведёт к нежелательным последствиям.

Несмотря на рассмотренные особенности и различия, при выборе инструментального средства статического анализа кода следует рассматривать все продукты, так как в каждой компании у специалистов по ИБ могут быть свои особенности в работе системы.

Использование программного обеспечения должно осуществляться под контролем службы информационной безопасности. Это необходимо, чтобы иметь управление над возможными рисками. Анализ кода программы должны выполнять ИБ-специалисты, потому что ошибки, которые не очень критичны для введения продукта в эксплуатацию, могут иметь большую значимость для информационной безопасности. Стоит всегда помнить о том, что всего лишь одна неисправность может привести к экономическим потерям, если она будет использована злоумышленниками. Благодаря качественному анализу кода, программное обеспечение будет максимально надёжным и работоспособным.

Список литературы

1. Буйневич М.В. Проблемные вопросы и тенденции обеспечения ИБ в сфере телекоммуникаций // Защита информации. Инсайд. 2017. № 1 (73). С. 49–55.

2. Skrypnikov A.V., Kozlov V.G., Denisenko V.V., Saranov I.A., Kuznetsova E.D., Savchenko I.I. Information security as the basis of digital economy. Advances in Economics, Business and Management Research. Proceedings of the Russian Conference on Digital Economy and Knowledge Management. 2020. P. 149–153.

3. Арапов Д.В., Скрыпников А.В., Денисенко В.В., Высоцкая И.А. Разработка и защита баз данных. Учебное пособие по дисциплине «СУБД Oracle». Воронеж: ВГУИТ, 2020. 100 с.

4. Израилов К.Е., Покусов В.В. Утилита для поиска уязвимостей в программном обеспечении телекоммуникационных устройств методом алгоритмизации машинного кода. Ч. 3. Модульно-алгоритмическая архитектура // Информационные технологии и телекоммуникации. 2016. Т. 4. № 4. С. 104–121.

5. Krasov A.V., Arshinov A.S., Ushakov I.A. Embedding the hidden information into Java byte code based on operands' interchanging. ARPN Journal of Engineering and Applied Sciences. 2018. Т. 13. № 8. P. 2746–2752.

6. Буйневич М.В., Израилов К.Е., Покусов В.В., Тайлаков В.А., Федулина И.Н. Интеллектуальный метод алгоритмизации машинного кода в интересах поиска в нем уязвимостей // Защита информации. Инсайд. 2020. № 5 (95). С. 57–63.