Интернет-журнал «Науковедение» ISSN 2223-5167 http://naukovedenie.ru/

Том 9, №2 (2017) http://naukovedenie.ru/vol9-2.php

URL статьи: http://naukovedenie.ru/PDF/63TVN217.pdf

Статья опубликована 26.04.2017

Ссылка для цитирования этой статьи:
Иванов К.И. Поиск уязвимостей в программном обеспечении, создаваемом для собственных нужд предприятия // Интернет-журнал «НАУКОВЕДЕНИЕ» Том 9, №2 (2017)
http://naukovedenie.ru/PDF/63TVN217.pdf (доступ свободный). Загл. с экрана. Яз. рус., англ.

УДК 62

Иванов Кирилл Игоревич
МУП «Йошкар-Олинская «ТЭЦ-1», Россия, Йошкар-Ола[1]
Инженер-программист
АНО ВО «Межрегиональный открытый социальный институт», Россия, Йошкар-Ола
Аспирант
E-mail: iv.kirill@bk.ru

Поиск уязвимостей в программном обеспечении, создаваемом для собственных нужд предприятия

Аннотация. В статье «Поиск уязвимостей в программном обеспечении, создаваемом для собственных нужд предприятия» автор рассматривает проблемы информационной безопасности, возникающие у предприятия при создании своего особого программного обеспечения. Автор сообщает, что созданный своими силами, оперативно дорабатываемый и узкоспециальный программный продукт, хотя и становится незаменимым инструментом в работе предприятия, все же может создавать новые уязвимости, которые не поддаются стандартным (антивирусы) методам контроля и устранения со стороны администраторов компьютерных сетей. В статье сообщены основные критические моменты оперативно создаваемых, для собственных нужд, приложениях. Далее, по ходу повествования, автор объясняет, что необходимо начинать заниматься безопасностью вновь создаваемого программного продукта еще на этапе его проектирования. Так же приводятся основные инструменты - коммерческие продукты применимые для анализа программного кода и поиска уязвимостей. Способов анализа и инструментов существует большое количество и большинство из них коммерческие, только комплексное их применение имеет максимальный эффект, поэтому автор советует применять экспертную оценку и нечеткую логику для сокращения расходов на этапе разработки и анализа программного обеспечения.

Ключевые слова: программное обеспечение; информационная безопасность; поиск уязвимостей; нечеткая логика; уязвимость в программном обеспечении; утечка данных; защита информации

Способы ведения деятельности каждого хозяйствующего субъекта уникальны, следовательно, «инструменты» ведения такой деятельности специфичны. У предприятия может возникнуть потребность в создании своего особого программного обеспечения. Созданный своими силами, оперативно дорабатываемый и узкоспециальный программный продукт становится незаменимым инструментом в работе предприятия. Но, там, где нестандартные

[1] 424039, Россия, Республика Марий Эл, г. Йошкар-Ола, ул. Лобачевского, д. 12

решения и новые возможности, там же и новые уязвимости неподдающиеся стандартам их контроля и устранения.

Следовательно, проблема поиска уязвимостей во вновь разрабатываемом специфичном программном обеспечении для собственных нужд хозяйственной деятельности предприятия носит крайне актуальный характер и требует анализа и решения. Более того, вопросы поиска уязвимостей в системе информационной безопасности предприятия всегда будут актуальны, в силу постоянного совершенствования методов хищения информации и вредоносного программного обеспечения. [2]

На практике, предприятию может потребоваться программное обеспечение осуществляющее получение информации из баз данных, их обработку заданным способом и запись обработанной, измененной, дополненной информации. Исходя из этого, высок риск возникновения следующих угроз информационной безопасности организации:

- несанкционированный доступ к внутренней сети предприятия через соединение, создаваемое приложением;
- несанкционированный доступ к таблицам баз данных предприятия;
- нанесение вреда, перегрузка серверов;
- прочие.

Данные угрозы могут быть обоснованы реализацией программного обеспечения [11], и могут представлять собой следующие проблемы:

- Ошибки или отсутствие проверки вводимых данных;
- Ошибки форматирующей строки;
- Неверная поддержка интерпретации метасимволов командной оболочки;
- SQL-инъекция;
- Инъекция кода;
- Подделка запросов в клиент-серверных приложениях;
- Прочие.

Очевидно, что неполный список вышеперечисленных проблем полностью дискредитирует даже высокозащищенную систему информационной безопасности предприятия. Все это еще раз подтверждает актуальность проблемы поиска и устранения уязвимостей во вновь создаваемом на предприятии узкоспециальном программном обеспечении. Следовательно, необходимо вывести комплекс мер направленных на устранение или нивелирование перечисленных ранее уязвимостей. Разделим эти мероприятия на две группы: мероприятия на этапе разработки кода приложения и мероприятия проверки готового приложения, поиска «слабых мест».

Во время разработки приложения, а именно его архитектуры и далее программного кода, возможно и необходимо закладывать особый и соответствующий уровню безопасности набор функций. Возможно, применение следующих мер:

- введение процедуры проверки перед запуском хэш-функции исполняемого файла приложения и его библиотек, с целью защиты от инъекции кода;
- защита от дизассемблирования;
- применение шифрованных протоколов обмена данным между клиентской и серверной частью программного продукта;
- при необходимости, разработка специфического протокола клиент-серверного обмена данными;

- создание процедур и функций, направленных на анализ вводимых пользователем данных;
- экранирование спецсимволов, служебных «слов», операндов применяемого языка запросов во вводимых пользователем данных;
- защита от переполнения буфера при работе с памятью, контроль утечек памяти, контроль выполнения потоков и уничтожения их по завершении операции;
- своевременное закрытие соединения с базами данных, контроль разрыва соединения.

Когда приложение готово, применены защитные функции, разработанный штатными программистами код возможно анализировать с применением готовых разработок или коммерческих программных продуктов. Так, О.Р. Маликов сообщает, что обнаружение уязвимостей в исходном коде программ обычно заключается в следующем:

- выявление переполнения буфера, с целью защиты от выполнения произвольного зловредного кода уязвимой программой, выполнившей переполнение памяти;
- выявление ошибок неконтролируемой форматной строки, с целью защиты от инъекции кода и SQL-инъекций [8].

Кроме того, в настоящее время разработано большое количество готовых инструментальных средств, предназначенных для автоматизации поиска уязвимостей программ. В них применяется как статический анализ, который производится над исходным кодом программы и реализуется без исполнения исследуемой программы, так и динамический анализ, который позволяет производить отладку программы в процессе её исполнения.

Для поиска ошибок переполнения буфера и ошибок форматных строк можно использовать следующие статические анализаторы: [5]

- ITS4 - инструмент, статически просматривающий исходный код для обнаружения потенциальных уязвимостей. Отмечает вызовы потенциально опасных функций, выполняет поверхностный семантический анализ, пытаясь оценить, насколько опасен такой код.
- Утилита RATS обрабатывает код, написанный на основных диалектах Си, скрипты на Perl, PHP и Python. Просматривает исходный текст, находя потенциально опасные обращения к функциям. Обеспечивает обоснованные выводы, опираясь на которые специалист сможет вручную выполнять проверку кода.
- PScan. Сканирует исходные тексты на Си в поисках потенциально некорректного использования функций, аналогичных printf, и выявляет уязвимые места в строках формата.
- Flawfinder. Статический сканер исходных текстов программ. Выполняет поиск функций, которые чаще всего используются некорректно, присваивает им коэффициенты риска (опираясь на такую информацию, как передаваемые параметры) и составляет список потенциально уязвимых мест, упорядочивая их по степени риска.

Все вышеперечисленные инструменты схожи и используют только лексический и синтаксический анализ. Поэтому результаты, выданные этими программами, могут состоять целиком из ложных сообщений. Поэтому, в дополнение к ним следует использовать и такие программные продукты, как:

- BOON. Инструмент, осуществляющий глубокий семантический анализ, автоматизирует процесс сканирования исходных текстов в поисках уязвимых мест, способных приводить к переполнению буфера.
- MOPS. Предназначен для динамической корректировки, обеспечивающей соответствие программы статической модели. MOPS использует модель аудита программного обеспечения, которая призвана помочь выяснить, соответствует ли программа набору правил, определенному для создания безопасных программ.
- Инструмент Viva64 помогает специалисту отслеживать в исходном коде программ потенциально опасные фрагменты, связанные с переходом от 32-битных систем к 64-битным. Анализатор помогает писать корректный и оптимизированный код для 64-битных систем.

Применение вышеперечисленных мер повышает стоимость программного продукта, увеличивает финансовые и временные затраты организации. Поэтому, при выборе инструментов и методов защиты и поиска уязвимостей, необходим поиск путей сокращения расходов с применением современных методов принятия решений. Одним из таких направлений является применение нечеткой логики при решении задач оценки уровня информационной безопасности создаваемого программного обеспечения. Это позволяет эффективно, быстро и удобно обрабатывать большие объемы данных полученных при применении экспертного метода оценки.

Так, применение нечетких множеств может использоваться для оценки общего уровня безопасности приложения, например, методом экспертной оценки, когда требуется на основе данного метода построить нечеткое множество, определяющее защищенность программного продукта на основе данных экспертов и ответов тестируемых пользователей, производящих оценку комплекса реализованных превентивных мер защиты.

Например, экспертам задают три вопроса, некие признаки выполнения мер безопасности, отвечать предлагается по некоторой шкале для каждого вопроса, эксперт может выбрать значение от K_{min} до K_{max}. Тогда K_x - выбранный экспертом ответ, а функция принадлежности $\mu_A(x)$ - оцениваемый уровень ситуации по заданному вопросу.

Так, разработчики оцениваемого программного продукта могут анкетироваться для определения степени незащищенности по определенным критериям, типовым уязвимостям.

Сводные данные опроса собираются в таблицу, с полями средний балл, максимум (K_{max}) и минимум (K_{min}) конкретному вопросу. А для оценки уровня разделим разницу минимального и среднего баллов на разницу между максимальным и минимальным баллами, т.е. функция принадлежности находится по формуле:

$$\mu_A(x) = (K_x - K_{min}) / (K_{max} - K_{min})$$

Оцениваемый уровень безопасности можно интерпретировать следующим образом: от 0 до 0,5 - плохой уровень защиты, больше или равно 0,5 до 1 - уровень защиты хороший. Далее уровень защищенности приложения будет представлен в виде носителя нечеткого множества **A(x)** и функции принадлежности $\boldsymbol{\mu_A(x)}$**.** Это позволит, используя преобразования Заде [6], получить общий уровень защищенности информационной систем предприятия.

Наличие такого инструмента для оценки защищенности вновь создаваемого программного обеспечения позволяет более гибко использовать существующие методики при оценке его защищенности, что повысит эффективность разработки программного обеспечения, т.к. больше внимания будет уделяться критическим уязвимостям, код будет дополняться только теми функциями проверки данных, которые актуальны и целесообразны требуемому уровню общей безопасности. Это, несомненно, сэкономит время разработчиков, а необходимость проводить проверки на наличие только критических уязвимостей позволить применять минимум дорогостоящих программных инструментов.

Таким образом, в статье были рассмотрены проблемы информационной безопасности, возникающие у предприятия при создании своего особого узкоспециального программного обеспечения. Из чего можно сделать вывод, что необходимо начинать заниматься безопасностью вновь создаваемого программного продукта еще на этапе его проектирования. Далее, после этапа проектирования вновь создаваемого программного продукта, уже на этапе разработки разработчикам следует применить анализ программного кода с целью исключения простых уязвимостей, связанных с ошибками и недочетами в коде приложения. Так же возможно применение коммерческих программных комплексов для статистического анализа кода и далее динамического анализа при исполнении приложения. Способов и инструментов анализа существует большое количество, только комплексное их применение возымеет максимальный эффект. Поэтому, следует применять экспертную оценку и нечеткую логику для распределения финансовых и трудовых ресурсов на отдельные, особо важные моменты уязвимости приложения. Это, несомненно, позволит уделять больше внимания более важным структурным элементам вновь создаваемого программного продукта и сократит расходы на этапе разработки и анализа программного обеспечения. Все это скажется на эффективности работы команды разработчиков программного обеспечения предприятия и внесет свою лепту в общую экономическую эффективность работы хозяйствующего субъекта.

ЛИТЕРАТУРА

1. Аветисян А.И., Белеванцев А.А., Чукляев И.И. Технологии статического и динамического анализа уязвимостей программного обеспечения // Вопросы кибербезопасности. 2014. №3 (4). С. 20-28.
2. Андреев Н.О. Современные проблемы безопасности корпоративных сетей // Прикладная информатика. 2008. №1. С. 25-31.
3. Бедердинова О.И., Иванова Л.А. Совершенствование метода тестирования программного обеспечения «Белый ящик» // Arctic Evironmental Research. 2014. №2. С. 113-123.
4. Буйневич М.В., Израилов К.Е., Щербаков О.В. Модель машинного кода, специализированная для поиска уязвимостей // Вестник ВИ ГПС МЧС России. 2014. №2 (11). С. 46-51.
5. Гайсарян С.С., Чернов А.В., Белеванцев А.А., Маликов О.Р., Мельник Д.М., Меньшикова А.В. О некоторых задачах анализа и трансформации программ // Труды ИСП РАН. 2004. С. 7-40.
6. Заде Л. Понятие лингвистической переменной и его применение к принятию приближенных решений. - М.: «Мир», 1976. С. 166.
7. Иванов К.И. Применение нечеткой логики для оценки уровня защищенности информационной системы // Труды ПГТУ. Серия: Социально-Экономическая. 2015. №3. С. 35-38.
8. Маликов О. Р. Автоматическое обнаружение уязвимостей в исходном коде программ // Известия ЮФУ. Технические науки. 2005. №4. С. 48-53.
9. Марков А.С., Матвеев В.А., Фадин А.А., Цирлов В.Л. Эвристический анализ безопасности программного кода // Вестник МГТУ им. Н.Э. Баумана. Серия «Приборостроение». 2016. №1 (106). С. 98-111.
10. Орлов А.И. Теория нечетких множеств - часть теории вероятностей // Научный журнал КубГАУ - Scientific Journal of KubSAU. 2013. №92. С. 51-60.
11. Тимофеев А.Э. Типовые уязвимости в программном обеспечении, предоставляющем защиту информации с помощью криптографических методов / А.Э. Тимофеев, И.В. Лунегов // Фундаментальные и прикладные проблемы механики, математики, информатики: сб. тр. науч.-практич. конф.-Пермь: ФГБОУ ВПО "Пермский государственный национальный исследовательский университет", 2015. - С. 335.

Ivanov Kirill Igorevich
Yoshkar-Olinskaya «TEC-1», Russia, Yoshkar-Ola
Interregional open social institute, Russia, Yoshkar-Ola
E-mail: iv.kirill@bk.ru

Searching for vulnerabilities in software created for the company's own needs

Abstract. In the article "Search for vulnerabilities in software created for the company's own needs", the author considers the problems of information security arising with the enterprise when creating its own special software. The author reports that the self-developed, quickly-developed and highly specialized software product, although it becomes an indispensable tool in the work of the enterprise, can still create new vulnerabilities that are not amenable to standard (antivirus) methods of control and elimination by computer network administrators. The article reports the main critical moments of applications that are created for their own needs. Further, in the course of the narrative, the author explains that it is necessary to begin to deal with the security of the newly created software product even at the stage of its design. Also, the main tools are provided - commercial products applicable for the analysis of program code and the search for vulnerabilities. There are a lot of methods of analysis and tools, and most of them are commercial ones, only their complex application has the maximum effect, therefore the author advises applying expert judgment and fuzzy logic to reduce costs during the development and analysis of software.

Keywords: software; information security; vulnerability search; fuzzy logic; vulnerability in software; data leakage; information security