МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

УСТОЙЧИВОСТЬ ВЕБ-ПРИЛОЖЕНИЙ ПО ОТНОШЕНИЮ К «SQL INJECTION»

АВТОРЕФЕРАТ МАГИСТЕРСКОЙ РАБОТЫ

студента 2 курса 273 группы
направления 01.04.02 — Прикладная математика и информатика
факультета КНиИТ
Маслова Михаила Андреевича

Научный руководитель
Зав.кафедрой, к. ф.-м. н. _____ И. А. Батраева

Заведующий кафедрой
к. ф.-м. н. _____ С. В. Миронов

Саратов 2019

ВВЕДЕНИЕ

Актуальность темы

В наши дни веб-приложения стали одной из составляющих жизни. Сотни тысяч и даже миллионы решений используются повсеместно. Однако, как разработчик, который заинтересован в стабильной работе своего приложения, так и клиент, зачастую оставляющий свои личные данные, документы и различную другую ценную и конфиденциальную информацию — обе стороны заинтересованы в безопасности и корректной работе веб-сервисов.

Безусловно, так как многие веб-приложения написаны по одинаковым шаблонам с использованием одинаковых языков программирования и одинаковых структур баз данных, хранящих информацию, существуют типичные уязвимости, используя которые злоумышленник может нарушить работу сервисов, внести изменения в какую-либо их часть и даже получить полный доступ, включая доступ к информации, внесенной туда пользователями.

Важным параметром любого веб-приложения является его устойчивость по отношению к различным уязвимостям.

Устойчивость – это критерий, который оценивает сложность реализации атак с использованием известных уязвимостей.

Особое внимание заслуживают такие уязвимости, как «SQL инъекции».

Актуальность данной темы состоит в том, что к этому типу атак уязвимы более 2/3 всех веб-приложений в мире.

Кроме того, большая часть уязвимых веб-приложений разрабатываются малыми командами разработчиков. Такие проекты не имеют большого бюджета, а также человеческого ресурса.

Существующие решения по обеспечению устойчивости веб-приложений по отношению к уязвимостям типа «SQL инъекции» имеют ряд недостатков:

1. Все актуальные и эффективные решения распространяются на коммерческой основе, что служит серьезной преградой для их использования в небольших проектах

2. Данные решения представляют из себя либо сервисы, являющиеся «прослойкой» между клиентом и сервером приложения, производящих фильтрацию данных, либо программное обеспечение, тестирующее веб-приложение «извне», пытаясь найти уязвимости в его коде. Все эти решения предназначены для больших приложений, и являются чрезмерно избыточными для небольшого проекта.

В небольшом проекте используется малое количество передаваемых между клиентом и сервером веб-приложения данных, поэтому разработчику наиболее удобно и комфортно вручную анализировать и изменять каждый участок кода, что является наиболее эффективным методом повышения устойчивости приложения по отношению к угрозам типа «SQL инъекции».

Цель магистерской работы – разработка программного обеспечения, способного анализировать код веб-приложения пользователя и предлагать советы по увеличению устойчивости конкретных участков кода по отношению к угрозам типа «SQL инъекции».

Поставленная цель определила **следующие задачи**:

1. Проанализировать существующие угрозы веб-приложений

2. Разработать алгоритм повышения устойчивости кода по отношению к данным уязвимостям

3. Рассмотреть различные методы защиты от угроз типа «SQL injection»

4. Разработать программное обеспечение, способное анализировать код вебприложения пользователя и предлагать советы по увеличению устойчивости конкретных участков кода по отношению к угрозам типа «SQL injection».

Методологические основы устойчивости веб-приложений по отношению к угрозам типа «SQL инъекции» представлены в работах H. Jagdish, S. Joel, A. Chris, S. Amichai, Д.А. Евтеева, Н.В. Скабцова и А.Н. Милосердцова.

Теоретическая и практическая значимость магистерской работы.

Теоретическая значимость магистерской работы заключается в обширном обзоре самой распространенной в мире уязвимости веб-приложений «SQL инъекции».

Практическая значимость магистерской работы заключается в реализации программного обеспечения, значительно упрощающего проблему увеличения безопасности отдельных участков кода по отношению к уязвимости вида «SQL инъекции».

Структура и объём работы. Магистерская работа состоит из введения, двух разделов, заключения, списка использованных источников и одного приложения. Общий объем работы – 82 страниц, из них 69 страниц – основное содержание, включая 55 рисунков и 0 таблиц, цифровой носитель в качестве приложения, список использованных источников информации – 21 наименование.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Первый раздел «Обзор уязвимостей веб-приложений» посвящен обзору современных наиболее популярных уязвимостей веб-приложений, а также анализу конкретных шагов по предотвращению атак данных типов.

Основные определения, понятия включают в себя такие термины, как:

SQL Инъекция — один из самых доступных способов взлома сервиса. Суть таких инъекций заключается во внедрении в данные (передаваемые через GET, POST запросы или значения Cookie) произвольного SQL кода;

API — описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой;

Хеширование — преобразование массива входных данных произвольной длины в (выходную) битовую строку установленной длины, выполняемое определённым алгоритмом;

Аутентификация — процедура проверки подлинности;

Хранимая процедура — объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере;

Placeholder — специальные текстовые поля, которые в процессе выполнения заменяются на соответствующие им динамически вычисляемые значения;

Веб-приложение — клиент-серверное приложение, в котором клиент взаимодействует с сервером при помощи браузера, а за сервер отвечает вебсервер;

Устойчивость веб-приложения — критерий, который оценивает сложность реализации атак с использованием известных уязвимостей.

Данный раздел посвящен обзору современных наиболее популярных уязвимостей веб-приложений. За основу был взят список самых критичных опасностей для веб-приложений в 2017 году, опубликованный открытым проектом обеспечения безопасности веб-приложений (OWASP).

Данный список включал в себя следующие уязвимости:

1. SQL Инъекция
2. Сломанная аутентификация и управление сеансом
3. Чувствительная к данным информация
4. XML Внешние объекты (XXE)
5. Нарушение контроля доступа
6. Неправильная конфигурация безопасности
7. Межсайтовый скриптинг (XSS)
8. Небезопасная десериализация
9. Использование компонентов с известными уязвимостями
10. Недостаточное журналирование и мониторинг

Все уязвимости были проанализированы на причину их возникновения, а также были разработаны конкретные шаги по предотвращению возможности реализации атак данного типа.

Отдельное внимание было уделено такой проблеме, как «SQL инъекции», чей механизм заключается в внедрении в данные, передаваемые от пользователя на сервер приложения, произвольного SQL-запроса.

Актуальность данной темы заключается в том, что уязвимостям вида «SQL инъекции» подвержено более 2/3 всех веб-приложений в мире.

Был составлен список наиболее популярных атак, использующих уязвимости типа SQL инъекции»:

1. Внедрение в строковые параметры
2. Использование оператора объединения
3. Экранирование хвоста запроса
4. Расщепление SQL-запроса

Также были разработаны методы по защите веб-приложения пользователя от них. Данные методы были поделены на две части:

- Стандартные методы защиты, позволяющие достаточно просто и быстро повысить безопасность веб-приложения от атак типа «SQL инъекции». К данным методам защиты относятся следующие манипуляции с кодом:

1. Создание менее привилегированного пользователя БД
2. Отключение сообщений об ошибках
3. Нестандартные имена таблиц и полей
4. Ограничение длины
5. Экранизация специальных символов
6. Использование хранимых процедур
7. Использование магических кавычек
8. Усечение входных параметров
9. Применение регулярных выражений

- Комбинированные методы защиты от SQL injection, являющиеся более сложными в реализации, но, при этом, более надежными.

К данным методам были отнесены использование «Placeholder»-ов и белого списка, а также таких технологий, как PDO и ORM.

Помимо обзора уязвимостей веб-приложений было также обращено внимание на Linux дистрибутив Kali Linux, используемый для нахождения различных уязвимостей в приложениях в автоматическом режиме. Данный дистрибутив предоставляет огромное количество инструментов, используемых практически всеми специалистами по информационной безопасности.

В рамках данного дистрибутива были рассмотрены инструменты, направленные на тестирование веб-приложений к уязвимостям типа SQLinjection, а именно:

- SQLMAP, являющийся одним из мощнейших инструментов, автоматизирующих процесс поиска уязвимых к SQL-инъекциям участков приложения с последующим извлечением данных и получением полного контроля над СУБД.
- jSQL Injection, использующийся на Java для автоматической инъекции в базы данных sql.

Таким образом, в рамках первого раздела были проанализированы современные наиболее популярные виды уязвимостей веб-приложений.

Кроме этого, отдельное внимание было уделено такой уязвимости, как «SQL инъекции», а также были разработаны конкретные методы по предотвращению возможности реализации атак данного типа. Также было также обращено внимание на Linux дистрибутив Kali Linux, используемый для нахождения различных уязвимостей в приложениях в автоматическом режиме.

Второй раздел «Разработка программного обеспечения для повышения устойчивости кода веб-приложений по отношению к угрозам типа SQL инъекции» посвящен реализации программного обеспечения, способного автоматически анализировать код веб-приложения пользователя и предлагать конкретные шаги по увеличению устойчивости кода по отношению к уязвимостям типа «SQL инъекции».

Данное программное обеспечение способно решать следующие поставленные задачи:

Поддержка загрузки файлов формата .txt, .aspx, .html и автоматический поиск участков кода, ответственных за прием данных со стороны пользователя.

Одними из наиболее популярных на сегодняшний день форматов файлов, используемых для взаимодействия пользователя с приложением, являются

HTML (язык гипертекстовой разметки, интерпретируя который браузер выводит на экран веб-страницу для пользователя) и aspx, представленный в рамках платформы разработки веб-приложений ASP.NET. Именно эти два формата файлов были выбраны в качестве основных для работы в приложении.

При указании пользователем желаемого файла вызывается метод, который производит процедуру построчного чтения файла. Получившийся массив строк передается блоку логики, который, в зависимости от типа файла, использует различные стратегии для поиска точек ввода данных.

Поиск участков кода, ответственных за прием данных со стороны пользователя, производится по ключевым словам, присущим соответствующему формату файла.

В файле формата .html поиск производится по следующим ключевым словам: «text», «input», «textarea», «password», «file» и «image».

В файле формата .aspx поиск производится по следующим ключевым словам: «TextBox» «TextMode» «type» «Label», «FileUpload». Кроме этого, так как в файле возможны html-вставки, также в список добавлены все ключевые слова, сигнализирующие о вводе данных, данного типа.

Выбор дополнительных полей для работы

Зачастую при разработке собственного веб-приложения разработчик использует нестандартные точки ввода данных, которое приложение обнаружить не может.

Нестандартные точки ввода данных – участки кода, от которых не ожидается процедуры ввода и передачи данных, но, по желанию разработчика, которые реализуют данный функционал.

У пользователя, путем взаимодействия с интерфейсом программы, присутствует возможность самостоятельно выбрать такие участки кода для их последующей обработки.

Автоматическая и ручная обработка участков кода с целью повышения устойчивости к атакам типа «SQL injection»

Обработка каждой найденной строки кода происходит в автоматическом и ручном режиме.

В автоматическом режиме пользователь, путем взаимодействия с интерфейсом приложения, отмечает ограничения на принимаемые данные и выбирает, какие дополнительные инструменты в данном месте он хотел бы использовать. При выборе параметров код автоматически изменяется.

В ручном режиме у пользователя есть возможность самостоятельно править строку, а также серверный код, который отвечает за ее обработку, в соответствии с рекомендациями приложения.

После завершения работы над выбранным участком кода происходит сохранение изменений путем нажатия на кнопку «Сохранить».

Система контроля версий

Так как пользователь постепенно работает с файлами, преобразуя нужные ему данные с целью повышения безопасности приложения, зачастую возникает потребность отмены каких-либо изменений ввиду различных факторов.

После завершения работы с файлом происходит сохранение получившегося результата. Приложение внесет изменения в выбранный файл, а также сохранит текущую версию файла в специальное хранилище, что реализует возможность продолжить работу с файлом в любой момент.

Для того, чтобы продолжить работу с уже сохраненной версией файла, необходимо перейти на вкладку «История», на которой предоставляется возможность выбора и загрузки для дальнейшей работы предыдущих модификаций файла.

Система контроля доступа к файлам и истории их изменения

Различные пользователи зачастую работают с файлами, будучи заинтересованными в конфиденциальности данных. Для того, чтобы воспользоваться данным инструментом, необходимо пройти процедуру аутентификации, либо, в случае если пользователь еще не зарегистрирован в системе, процедуру регистрации. После входа в систему, при сохранении новой версии файла, пользователю предоставляется возможность запретить доступ к данным другим, выбрав соответствующую настройку, после чего данная версия доступна для просмотра и модификации только его владельцу.

Удаленная работа с файлами

Так как функционал доступа к истории модификации файлов с возможностью продолжения работы с любой сохраненной версией, а также система контроля доступа к данным не могут быть раскрыты в полной мере в рамках локальной работы приложения (или даже в рамках единичной сессии),

возникла потребность в хранилище данных, доступ к которому был бы возможен из любой точки, имеющий доступ к интернету.

Была использована база данных, на основе которой была разработана структура хранения информации о пользователях, их файлах и истории их изменения. Были внедрены хранимые процедуры, реализующие доступ к данным. Вызов хранимых процедур, а также прием и передача данных реализованы с использованием технологии ADO.NET.

В рамках данного раздела были выполнены все поставленные задачи по разработке приложения, способного автоматически анализировать код веб-приложения пользователя и предлагать конкретные шаги по увеличению устойчивости кода по отношению к уязвимостям типа «SQL инъекции».

ЗАКЛЮЧЕНИЕ

В ходе данной выпускной квалификационной работы были выполнены поставленные задачи.

Были рассмотрены основные проблемы безопасности веб-приложений в настоящее время и способы их решения. За основу был взят список самых критичных опасностей для веб-приложений в 2017 году, опубликованный открытым проектом обеспечения безопасности веб-приложений (OWASP)

С целью увеличения устойчивости веб-приложений по отношению к уязвимостям типа «SQL injection» было спроектировано программное обеспечение, способное помочь пользователю в создании защищенного от SQLинъекций сервиса путем анализа кода веб-приложения пользователя и предложения советов по увеличению устойчивости конкретных участков кода по отношению к угрозам типа «SQL инъекции».

Приложение решает все поставленные задачи, предоставляя функционал для удобной и качественной разработки приложения, отвечающего современным стандартам безопасности, а именно:

— Поддерживает загрузку файлов формата .txt, .aspx, .html и автоматический поиск участков кода, ответственных за прием данных со стороны пользователя.

— Анализирует найденные участки кода и предлагает советы по увеличению их устойчивости по отношению к угрозам типа «SQL инъекции».

— Реализовывает автоматическую и ручную обработку найденных точек кода с целью повышения их устойчивости по отношению к атакам типа «SQL инъекции».

— Предоставляет возможность выбора пользователем дополнительных участков кода в обрабатываемом файле, не обозначенных автоматической системой поиска

— Предоставляет возможность удаления необязательных для обработки участков кода из списка

— Сохраняет правки в выбранный файл

— Поддерживает систему контроля версий, поддерживающая работу с историей изменения файлов

— Реализовывает систему контроля доступа к файлам и истории их изменения

— Предоставляет возможность удаленной работы с файлами

В завершении работы с приложением пользователь получает устойчивый по отношению к атакам типа «SQL инъекции» код веб-приложения.

Основные источники информации:

1. Jagdish, H. SQL Injection. Are Your Web Applications Vulnerable? /H. Jagdish. — London: SPILABS, 2014.
2. Joel, S. Hacking Exposed. Web Applications / S. Joel. — Москва: Вильямс, 2003.
3. Chris, A. Advanced SQL Injection In SQL Server Applications / A. Chris. — London: Next Generation Security Software Ltd, 2002.
4. Amichai, S. Blindfolded SQL Injection / S. Amichai. — New York: Imperva, 2013.
5. Д. А. Евтеев,. SQL Injection от А до Я / Д. А. Евтеев. — Сант-Петербург: Positive Technologies, 2013.
6. Н. В. Скабцов,. Аудит безопасности информационных систем / Н. В. Скабцов. — Сант-Петербург: Издательский дом «Питер», 2006.
7. А. Н. Милосердцов,. Kali Linux: тестирование на проникновение / А. Н. Милосердцов. — Москва: WebWare.biz, 2015.