Негосударственное образовательное учреждение
высшего образования
Московский технологический институт

Программа дополнительного профессионального образования
__Управление информацией и хранение данных на платформе EMC__
Наименование программы дополнительного профессионального образования в соответствии с договором

Допущена к защите
Проректор по ДПО
_____
« ____» _____ 2018 г.

ВЫПУСКНАЯ АТТЕСТАЦИОННАЯ РАБОТА

на тему:

«Анализ уязвимостей web-приложений и способы защиты с использованием Java-технологий на примере организации»

Слушатель

Научный руководитель Иванов Виталий Андреевич

Москва, 2018

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....3
1.ТЕОРЕТИЧЕСКИЕ ПОНЯТИЯ ЗАЩИТЫ WEB-ПРИЛОЖЕНИЙ ОТ УЯЗВИМОСТЕЙ.....6
1.1. Определение, виды и структура web-приложений.....6
1.2. Понятие и классификация уязвимостей web-приложений.....7
1.3. Методы защиты web-приложений.....13
2. ПРИМЕНЕНИЕ JAVA-ТЕХНОЛОГИИ ДЛЯ ЗАЩИТЫ WEB-ПРИЛОЖЕНИЙ ОТ УЯЗВИМОСТЕЙ НА ПРИМЕРЕ ОРГАНИЗАЦИИ.....23
2.1. Характеристика и направление деятельности организации. Описание аппаратного и программного обеспечения.....23
2.2. Анализ уязвимостей web-приложений на примере организации.....27
2.3.Мероприятия по обеспечению защиты web-приложений с использованием Java-технологий в организации.....34
ЗАКЛЮЧЕНИЕ.....53
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ.....55

ВВЕДЕНИЕ

В наше время, общество не может обойтись без разнообразных компьютерных систем и их программного обеспечения. Их можно встретить почти во всех сферах жизнедеятельности человека. Они делают жизнь человека гораздо комфортнее. Собственно, программное обеспечения и предоставляет человеку возможность решать огромный диапазон всевозможных задач, начиная с общения в социальных сетях в Глобальной сети, заканчивая проведением банковских операций.

Информационные технологии с каждым днём все больше входят в ответственные сферы жизнедеятельности человека. Люди совершают покупки в Интернет магазинах, свои деньги мы сохраняем в банках в электронном виде, все больше документов заполняется с компьютера, огромное количество личных данных сохраняется на разнообразных веб-ресурсах и в удалённых источниках. Чрезвычайно важным становится вопрос защиты такой информации от вредоносного влияния. Решение данной задачи считается совокупным походом и имеет довольно развитую научно-методическую основу.

В международной практике, вопрос обеспечения информационной безопасности рассматривают на разных уровнях и с разных сторон, начиная с аудита безопасности функционирующей компьютерной системы и заканчивая проверкой надёжности ПО на стадии планирования. По мнению российских и зарубежных учёных многофункциональные характеристики информационно- компьютерной системы, именно, определяются надёжностью и качеством ПО. Чем сложнее считается информационная система, тем более актуальным и критичным становится вопрос обеспечения надёжности безопасности информации. Вследствие этого обеспечение безопасности технологий по созданию ПО информационных систем становится важнейшей задачей, а обеспечение контроля на стадиях разработки ПО становится все более актуальным.

Одно из направлений по обеспечению контроля разработки заключается в анализе исходного кода ПО. В этом направлении выделяют 2 главных подхода к аудиту: динамическое тестирование и статический анализ. Динамические методы используются, когда имеется доступ исключительно к интерфейсу ПО, то есть, это тестирование способом чёрного ящика. Такой подход считается эффективным только при тестировании стабильной работы веб-приложения и корректности данных, но даёт низкий результат при нахождении ошибок, которые связаны с намеренно привнесёнными ошибками во входные данные.

Если к программе имеется абсолютный доступ, а вернее к ее исходному коду, то гораздо лучше использовать способы статического анализа. Эти способы применяют исходные тексты программы и разнообразную добавочную информацию, к примеру, связанную со сферой выполнения. Большим достоинством статического анализа кода считается отсутствие необходимости в прогоне программы при всевозможных условиях работы в стремлении достичь большей автоматизацией проверок. Достичь этого можно анализом конструктивных свойств исходного кода.

Следовательно, острой проблемой считается необходимость введения в разработку ПО процессов, предоставляющих возможность обеспечения надёжности и качества исходного кода в течении жизненного цикла разработки ПО.

Целью данной работы является анализ уязвимостей web-приложений и способы защиты с использованием Java-технологий на примере организации.

Для достижения заданной цели в работе необходимо выполнить ряд задач:

- разработать критерии для сравнения технологий поиска веб-уязвимостей, критерии сравнения обеспечения безопасности интегрированных средств тестирования;

- провести обзор и анализ современных технологий тестирования веб-уязвимостей;

- провести анализ современных веб-уязвимостей и рассмотреть варианты их реализации и способы защиты;

- провести сравнение интегрированных сред тестирования веб-уязвимостей;

- усовершенствовать интегрированное средство тестирования веб-уязвимостей путем разработки нового сценария тестирования.

При написании работы были применены такие методы теоретико-методологической базы, как изучение научной литературы по теме исследования, нормативно-правовой базы, аналитический и сравнительный методы.

Эмпирической базой исследования стало изучение основных принципов изучения методов изучения определения уязвимостей и угроз веб-приложения для защиты информации от несанкционированного доступа.

Основой работы послужили различные источники информации по защите веб-приложений. Это нормативные документы, периодические издания, электронные ресурсы, научная и учебная литература как отечественных, так и иностранных авторов.

1.ТЕОРЕТИЧЕСКИЕ ПОНЯТИЯ ЗАЩИТЫ WEB-ПРИЛОЖЕНИЙ ОТ УЯЗВИМОСТЕЙ

1.1. Определение, виды и структура web-приложений

Веб-приложение - комплекс программ, осуществляющих обработку информации в определенной области использования в сфере Веб.

Веб-приложение – это дополнительные программные средства, которые предназначаются для автоматизированного выполнения некоторых манипуляций на Web-серверах. Вместе с тем, они используют как пользовательский интерфейс Web-браузеры. Зачастую Web-приложения формируются в различных вариациях архитектуры клиент-сервер.

Веб-приложение – это приложение производящееся на сервере, доступ к которому реализуется посредством передачи-получения пакетов через протокол HTTP.

Совокупная архитектура Web-приложения – модель клиент-сервер, где элементы приложения делятся на пользовательские и серверные, объединяются посредством передачи -получения пакетов через протокол HTTP. Как клиент выступает Web-браузер, а как сервер – Web-сервер (рисунок 1.1).



Рисунок 1.1 - Совокупная архитектура Web-приложения

Цели Web-сервера – выполнить определенные функции по обработке информации и предоставить интерфейс (программный или пользовательский) через протокол HTTP для доступа к данным функциям. Серверный элемент Web-приложения принимает запрос от клиента, производит обработку информации, затем создает ответ и/или Web-страницу и пересылает пакет данных пользователю по сети через протокол HTTP.

Серверные элементы Web-приложения могут проявляться в качестве клиента прочих служб (сервисов), к примеру, информационной базы либо иного Web-приложения, которое расположено на другом сервере.

Одной из положительных сторон данного подхода считается то, что клиенты не зависимы от определенной пользовательской операционной системы, и Web-приложения, следовательно, являются межплатформенными сервисами.

Динамичные приложения содержат меняющееся значимо-информационное содержание (контент), а постоянное Web-приложение неменяющееся содержание.

Качество создаваемого приложения устанавливается соответствием требованиям, заложенным на этапе проектирования системы. Все условия к приложениям делятся на функционирующие и нефункционирующие. Функциональные условия определяют ту работу системы, которую авторы должны создать, для того, чтобы пользователи могли выполнять задачи в пределах своих бизнес-процессов. Нефункциональные условия к системе задают свойства надежности, скорости работы приложения, безопасности и масштабируемости.

1.2. Понятие и классификация уязвимостей web-приложений

Самым надежным на данный момент является рейтинг топ 10 уязвимостей от проекта OWASP.

OWASP является открытым проектом по обеспечению безопасности приложений. Опубликованный OWASP топ-10 уязвимостей считается перечнем самых критичных рисков безопасности приложений, отображающим текущие проблемы в сфере ИБ.

Одно из ключевых направлений жизнедеятельности A1QA состоит в тестировании безопасности программных продуктов. В данной компании работает центр компетенции по тестированию безопасности и лаборатория, проводящая тесты на проникновение и аудит безопасности ПО. Инженерами

компании учитывался обновлённый топ OWASP. В предоставленной работе буду рассмотрены изменения Топ-10 в сравнении с предшествующей версией. Рассмотрим, какие из уязвимостей были добавлены, какие исключены из перечня, о чём свидетельствуют такие изменения – рисунок 1.2.



Рисунок 1.2 – Рейтинг веб-уязвимостей топ 10 от проекта OWASP – сравнение

Безусловно, имеется значительно больше уязвимостей приложений, которые способны обеспечить злоумышленникам получение доступа к веб сайту и похищение ценной информации. Впрочем, проверка на уязвимости OWASP Топ-10 считается верным шагом по обеспечению защищённости и безопасности данных.

Рассмотрим подробнее данные уязвимости.

1. Инъекции. В качестве инъекций подразумевают уязвимости, возникающие вследствие передачи не проверенных данных, которые пользователь ввёл интерпретатору в целях выполнения. Следовательно, каждый из пользователей имеет возможность выполнения в интерпретаторе произвольного кода. Наиболее распространёнными типами инъекций считаются LDAP, XXE, OS и SQL.

Если 3 первые считаются мало известными, то SQL инъекции знакомы многим. С их помощью у злоумышленника появляется возможность доступа к БД; он может прочитать из БД секретные данные либо даже записать собственные значения. Возникают инъекции, когда данные, которые передаются интерпретатору не подвергаются проверке на наличие управляющих последовательностей, а также таких команд, как кавычки в SQL.

2. Вопросы аутентификации и проверки сессий. Большинство приложений запрашивает идентификацию пользователей, чтобы работать с ними. Зачастую функции проверки достоверности и управления сессиями реализовываются неправильно, что даёт злоумышленникам возможность доступа к учётным пользовательским записям в обход паролей. Также злоумышленники имеют возможность каким-либо способом осуществить перехват ключей либо токенов сеанса, выступающих как идентификаторы пользователей и применять их постоянно или временно.

3. XSS. Две рассмотренные выше уязвимости представляли гораздо больше опасностей для веб-сайта и его сервера. Уязвимость XSS считается не опасной для сервера, однако представляет опасность для пользователя. Работает она в пользовательском браузере, и поэтому создаёт условия только для кражи пользовательских данных. Cross-Site Scripting или XSS функционирует в JavaScript. Принцип такой же, как и в инъекциях. Злоумышленник осуществляет передачу специальной строке в каком-нибудь поле, строка содержит JS код, дальше браузер считает, что данный код направлен сайтом и реализовывает его, хотя код может быть любым. Чтобы

противодействовать подобным атакам требуется экранирование всех специальных символов посредством функции htmlspecialchars либо ее аналогов.

4. Вопросы контроля доступа. Зачастую в связи с недосмотром администраторов простым пользователям становятся доступными данные, которые обязаны быть закрыты. Данной проблеме могут подвергаться даже самые популярные движки. Примером могут послужить файлы в корне веб-сайта. К примеру, файл wp-config.php с паролями доступа к БД недоступен, так как имеет расширение php. Однако если его редактировать в Vim и сохранить неправильно, то формируется резервная копия с расширением .swp, которую можно уже без препятствий открыть в браузере.

В качестве проблемы контроля доступа также можно подразумевать ошибки в коде пользовательского приложения, которые могут открыть для не авторизованных пользователей доступ к засекреченным данным.

5. Неправильная конфигурация. Для безупречной безопасности приложение запрашивает безопасную конфигурацию не только спланированной и разработанной на уровне фреймворка и веб-приложения, а и правильно настроенных серверов. Настройки безопасности обязаны разрабатываться, осуществляться и стабильно поддерживаться. По умолчанию конфигурация большинства сервисов считается не безопасной. Помимо этого, ПО должно поддерживаться в актуальном состоянии.

6. Незащищенные конфиденциальные данные. Большинство веб-приложений, сайтов и API не обеспечивают защиту пользовательских конфиденциальных данных и осуществляют их передачу в открытом виде. Такими данными могут считаться не только ключи, токены и пароли, а и финансовая и медицинская информация. Применяя атаку «Человек посередине» у злоумышленников появляется возможность кражи или даже модифицирования важных данных. Конфиденциальные данные обязаны быть защищены, к примеру, посредством шифрования https либо при помощи других способов.

7. Недостаточная защита от атак. Большая часть API и приложений не располагают базовыми возможностями по выявлению, предотвращению и реакцией на автоматические и ручные атаки. Защитой от атак считается нечто большее, нежели базовая проверка соответствия пароля и логина. Также она должна содержать в себе выявление, документирование и даже блокировку действий по неверному входу и прочих несанкционированных действий. У разработчиков приложений также должна иметься возможность быстрого развёртывания исправлений, чтобы обеспечить защиту от новых атак.

8. Уязвимости CSRF. Атака Cross-Site Request Forgery или CSRF предоставляет злоумышленнику возможность вынудить браузер жертвы осуществить отправку в уязвимое приложение определённого HTTP запроса, включая файлы сеанса, куки и различную остальную, в автоматическом режиме включаемую информацию.

Таким способом, у злоумышленника появляется возможность генерации запросов из браузера пострадавшего, которые приложение принимает как правильные и отправленные самим пострадавшим. К примеру, пользователь просто открывает ссылку, а веб-сайт уже осуществляет отправку его друзьям сообщения, содержащего рекламное содержание без его ведома либо удаляет его аккаунт.

9. Применение элементов с уязвимостями. Такие элементы, как фреймворки, библиотеки и прочие программные модули функционируют с такими же полномочиями, как и приложение. Когда в одном из элементов имеется уязвимость, то вследствие атаки на неё у злоумышленника появляется возможность кражи важных данных или даже получения управления над сервером. API и приложения, применяющие элементы с уязвимостями могут ослабить защиту веб-приложений и сделать потенциальными различные атаки. Могут быть различные типы уязвимостей веб-сайтов, которые могут открывать доступ к всевозможным данным.

10. Незащищенные API. Большая часть современных приложений

зачастую содержат в себе веб-приложения клиентов, а также богатые API интерфейсы, которые доступны из мобильных приложений и через JavaScript в браузере. Они работают по протоколам GWT, RPC, REST/JSON, SOAP/XML и др. Такие API очень часто являются не защищёнными и также содержат большое количество ошибок, приводящих к уязвимостям.

По результатам анализа установлено, что 25% веб-приложений подвергаются 8-ми уязвимостям из перечня OWASP TOP-10. Кроме указанных выше, в число самых распространённых вошли:

- отсутствие функций контроля доступа (33%);
- небезопасная конфигурация (37%);
- недочёты системы аутентификации и сохранения сессий (41% веб-приложений).

Также исследователями обнаружено, что все из изученных веб-приложений имеют как минимум одну уязвимость (на одно веб-приложение приходится в среднем 45 проблем).

Помимо этого, специалистами проведён сравнительный анализ веб-приложений, которые разработаны по технологиям .NET и Java. Оказалось, что Java-приложения в основном могут подвергаться уязвимостям, которые позволят внедрение SQL-кода и CSRF-уязвимостям (соответственно 38% и 69% решений) и в меньшей степени проблемам, которые относятся к небезопасной конфигурации (14%) в сравнении с .NET-приложениями.

Соучредитель и технический директор Contrast Security Джефф Уильямс (Jeff Williams) констатирует, что результаты исследования пробуждают глубокое беспокойство. Абсолютно все выявленные уязвимости находятся в перечне OWASP более десяти лет, однако до сих пор продолжают оставаться серьёзной проблемой. Принимая во внимание возрастающие угрозы и уровень уязвимости, предприятиям необходимо осуществлять контроль безопасности своих приложений, заметил Уильямс.

В данной главе были рассмотрены виды уязвимостей веб-сайта, которые по версии ресурса OWASP, встречаются больше всего. Установлено,

что среди таких программных проблем, как CSRF, XSS или SQL инъекции, существуют и проблемы, затрагивающие настройку серверов, которые также часто приводят к проблемам. Зная, какие уязвимости сайтов существуют, можно любой ресурс сделать более безопасным.

1.3. Методы защиты web-приложений

В предоставленной работе будут рассматриваться некоторые вопросы, касающиеся анализа средства тестирования уязвимостей веб-приложений в качестве программных продуктов. Современные средства тестирования уязвимостей считаются многофункциональным и весьма сложным продуктом, поэтому их анализ и сравнение с похожими решениями имеет некоторые особенности.

В научной статье «Building a Test Suite for Web Application Scanners» раскрыты общие принципы анализа средств тестирования уязвимостей, на которых будем основываться дальше. Один из таких принципов состоит впроцедуре анализа в целях сравнения работы всевозможных средств тестирования уязвимостей веб-приложений. Немного видоизменённый вариант этой процедуры выглядит таким образом:

1. Выполнить подготовку необходимого контента анализа для функциональной проверки абсолютно всех технических требований, а также развернуть стенды для анализа.

2. Инициализировать анализ, получить все требуемые настройки для анализа.

3. Сконфигурировать анализируемое веб-приложение и избрать для него уровень защиты и тип уязвимости.

4. Запустить средство тестирования уязвимостей с подобранными настройками на анализируемом веб-приложении и пройти комплект функциональных анализов.

5. Осуществить подсчёт и классификацию найденных средством тестирования уязвимостей веб-объекты (векторы атаки, уязвимости, уникальные ссылки и пр.).

6. Шаги 2—5 повторить для каждого уровня защиты и типа уязвимостей.

Изменения после любой из итераций необходимо вносить в составленную таблицу итогов детектирования объектов. Таблица должна примерно иметь такой вид, как представлено на рисунке 1.3.

Тип уязвимости или функциональный модуль сканера	Уровень защиты	Кол-во обнаруженных объектов	False Positive	False Negative	Всего реальных объектов для поиска	Общее время сканирования
Модуль Crawler - для первоначального поиска объектов сканирования (ссылок)	0	100	0	50	150	500 с.
	1	90	0	60		600 с.
	2	80	0	70		700 с.
	...	...	...	...		...
Модуль поиска XSS-уязвимостей	0	2	0	2	4	15 с.
	1	1	1	3		20 с.
	2	0	2	4		30 с.
	...	...	...	...		...
Модуль поиска SQL-уязвимостей	0	1	2	1	2	60 с.
	1	0	3	2		120 с.
	2	0	4	2		240 с.
	...	...	...	...		...
Модуль определения установленных сетевых служб Application Fingerprint	0	3	0	0	3	50 с.
	1	1	2	2		60 с.
	2	0	2	3		70 с.
	...	...	...			...

Рисунок 1.3 – Пример таблицы итогов детектирования объектов

Разумеется, что отнюдь не все средства тестирования уязвимостей веб-приложений имеют одинаковый набор сканирующих модулей, впрочем, подобную таблицу можно все равно применять уменьшая рейтинг средства тестирования уязвимостей с случае отсутствия тех либо других модулей, той либо другой функциональности.

Подготовить для анализа приложение с заранее известным количеством уязвимостей конкретного типа невозможно. Вследствие этого,

составляя подобную таблицу мы неизбежно столкнемся с проблемами определения числа действительных объектов для поиска. Данная проблема может решаться таким образом.

Рассматривать в качестве одной уязвимости класс эквивалентных уязвимостей, которые могут быть выявлены в анализируемом приложении. К примеру, для SQL-инъекций классом эквивалентных уязвимостей могут считаться все уязвимости, которые найдены для одного и того же самого параметра GET-запроса к веб-приложению.

Осуществить для анализа разработку простейших приложений, реализующих или моделирующих некую уязвимость, однако используют различные фреймворки и разворачивая их на большом количестве вариантов ОС, всевозможных веб-серверах, с применением различных БД, с доступом по разнообразным сетевым протоколам, а также посредством различных цепочек прокси.

На стендах для анализа развернуть большое количество разнообразных CMS, уязвимых приложений (OWASP Site Generator, Gruyere, DVWA и др.) и сканировать их разными средствами тестирования уязвимостей. Общее количество уязвимостей, которые найдены всеми средствами тестирования уязвимостей, принять за эталон и применить в дальнейших анализах.

Конфигурировать приложение для анализа и осуществлять управление им, устанавливая необходимый уровень защиты, возможно, например, посредством инструмента OWASP Site Generator, конфигурация которого может храниться и редактироваться в простом XML-файле. К огорчению, сегодня этот инструмент уже считается устаревшим, поэтому для эмуляции современных уязвимостей рекомендовано создавать веб-приложения собственными силами.

Виды уязвимостей для осуществления в контенте анализа проверки средств тестирования уязвимостей перечислены в классификаторе WASC Threat Classification.

Ожидаемое количество запусков процедуры анализа для всех без исключения потенциальных комбинаций установленных веб-приложений будет чрезвычайно большим. Уменьшить это количество можно при помощи использования техники тестирования pairwise analysis.

По итогам сканирования приобретаем числовые векторы вида как представлено в таблице 1.1.

Потом следует ввести метрику качества сканирования, которая может использоваться, чтобы сравнить показатели и средства тестирования уязвимостей между собой.

В процессе анализа средств тестирования уязвимостей веб-приложений можно опираться ещё на одну статью, которая получила название «Analyzing the Accuracy and Time Costs of Web Application Security Scanners». Данная статья описывает проведение испытаний всевозможных средств тестирования уязвимостей (Acunetix, Hailstorm, Appscan, NTOSpider, WebInspect, Qualys, BurpSuitePro) и для каждого определённого инструмента было предложено провести 4 типа испытаний:

1. Провести анализ веб-приложения в режиме Point and Shoot (PaS) и установить количество обнаруженных и подтвержденных уязвимостей.

2. Провести повторный анализ после предварительного «обучения» и настройки средства тестирования уязвимостей на работу с данным типом веб-приложений, определить количество обнаруженных и подтверждённых уязвимостей в данном случае.

3. Дать оценку полноте и точности описания обнаруженных уязвимостей.

4. Дать оценку общему времени, которое специалисты потратили на подготовку и выполнение анализа, тестирование и обеспечение качества итогов анализа.

Режим PaS считается запуском сканирования с типовыми параметрами средства тестирования уязвимостей. Реализовывается согласно схеме: «постановка цели — сканирование — получение результата».

Обучение содержит в себя различные конфигурации настроек, связь с поставщиками средств тестирования уязвимостей, изменение скриптов и пр.

Чтобы определить количество времени, затраченное специалистами для получения качественного результата, в работе предлагается использовать простую формулу:

Общее время = Время обучения + #False Positive * 15 мин. + #False Negative * 15 мин.

При каждом испытании необходимо использовать процедуру анализа, которая была описана раньше.

Авторами ещё одной статьи «Top 10: The Web Application Vulnerability Scanners Benchmark» предложен единый поход к сравнению характеристик средств тестирования уязвимостей и набор этих характеристик с примерами. В данной работе предлагается задавать оценку возможностей средств тестирования уязвимостей веб-приложений в виде таблицы, используя такие критерии:

1. Сопоставление стоимости продукта относительно оценкам критериев.

2. Универсальность использования средств тестирования уязвимостей — это количество поддерживаемых протоколов и векторов доставки — способов доставки данных к серверу. Векторы доставки содержат в себе способы доставки данных к серверу, такие как HTTP-параметры строк запроса, параметры XML, JSON, HTTP-body, двоичные способы для таких определённых технологий, как Java-сериализованные объекты, AMF и WCF.

3. Число поддерживаемых векторов атаки: тип и число активных плагинов средства тестирования уязвимостей.

4. Точность выявления CSS.

5. Точность выявления SQL-инъекций (SQL Injection Detection Accuracy).

6. Точность обхода структуры веб-приложения и выявления локальных файлов.

7. Удаленное применение файлов, XSS, фишинг посредством RFI.

8. WIVET-сравнение: автоматизированный краулинг и получение входных векторов для атаки.

9. Адаптивность технологии: число дополнительных возможностей средства тестирования уязвимостей с целью преодолеть защитные барьеры.

10. Сопоставление особенностей аутентификации: тип и количество поддерживаемых способов аутентификации и авторизации.

11. Число дополнительных возможностей сканирования и интегрированных механизмов.

12. Общее представление о работе ключевой функции сканирования.

Отдельные из перечисленных выше пунктов входят в обобщённую таблицу детектирования объектов. Помимо этого, можно отметить, что для большей части критериев необходимы экспертные оценки, что затрудняет сравнение средств тестирования уязвимостей и автоматизированный анализ. По причине предлагаемые критерии оценок могут использоваться, например, как разделы для наиболее общего отчета по изучению характеристик средств тестирования уязвимостей, в котором имеются все результаты тестирования и сравнения определённого средства тестирования уязвимостей с конкурентами.

Основываясь на материалах, представленных выше, была разработана классификация типов анализов, которые можно применять в процедуре анализа.

Базовые функциональные (smoke) анализы должны осуществлять проверку работоспособности ключевых низкоуровневых узлов средства тестирования уязвимостей: работу подсистемы конфигурации, транспортной подсистемы, подсистемы логирования и др. В процессе сканирования простейших целей анализа сообщений об ошибках быть не должно: traceback и exceptions в логах, средство тестирования уязвимостей не должно зависать в процессе использования разных транспортов, прокси-серверов, редиректов и пр.

Функциональные (functional) анализы обязаны осуществлять проверку ключевых сценариев, чтобы проверить технические требования. Необходимо выполнять проверку работоспособности каждого из сканирующих модулей последовательно при разных настройках тестового окружения и модуля. Выполняются негативные и позитивные сценарии анализа, разнообразные стресс-анализы с применением больших массивов некорректных и корректных данных, которые оправляются от веб-приложения в ответ средству тестирования уязвимостей.

Анализы на сравнение (compare) функциональности, в процессе которых производится сопоставление качества и средней скорости нахождения объектов избранным модулем средства тестирования уязвимостей с похожими по функционалу модулями в продуктах конкурентов. Для каждого определённого сканирующего модуля обязаны предоставляться определения, что подразумевается в качестве объектов для него и качеством их нахождения.

Анализы на сопоставление показателей оценочных критериев (criteria), в процессе которых выполняется проверка того, что бы качество и скорость нахождения объектов каждым из сканирующих модулей в каждой из новых версий анализируемого средства тестирования уязвимостей не ухудшились в сравнении с предшествующей версией. Определения качества и скорости должны задаваться аналогично анализам на сопоставление функциональности: за исключением, что в этом типе анализов в роли конкурента для анализируемого средства тестирования уязвимостей выступает его предшествующая версия.

Применять описанную в предоставленной работе процедуру анализа можно для интегрированных сре6дств тестирования уязвимостей безопасности веб-приложений, а используя метрику качества сканирования, можно получить инструмент для качественного сопоставления средств тестирования уязвимостей через сравнение их метрик. При развитии данной

идеи можно применять нечёткие показатели, метрики и шкалы, чтобы упростить работы по сравнению средств тестирования уязвимостей.

Современный рынок сегодня предоставляет всевозможные решения в сфере статического анализа. Рассмотрим основные технологии тестирования веб-уязвимостей.

Referrer Policy Cont представляет собой технологию для сканирования слабых и уязвимых мест, которые созданы на Java. Технология используется для включения и выключения прокси процессов и автоматизированного сканера. Усовершенствованная версия технологии способствует нахождению и фиксированию XSS (сценариев перекрестного веб-сайта), SQL-инъекций и др. Технологию можно расширить мощным API на языке Javascript.

Fetch считается довольно простой в применении технологией нахождения и тестирования уязвимостей в приложениях. Ее можно использовать в качестве сканера или для прерывания прокси в целях ручного тестирования веб-страницы. Основные особенности: пауки Fuzzer, AJAX, REST базирование API и веб-поддержка сокета.

Wapiti представляет собой технологию, предоставляющую возможность сканирования веб-страниц в функционирующем приложении, выполнения «чёрного ящика» и введения нагрузки на веб-приложение для проверки уязвимости сценария. Wapiti находит уязвимости в процессах, таких как: раскрытие файла и его включение, обнаруживает слабые htaccess конфигурации и XSS сценарии пр. Располагает большой базой данных инъекций (XPath-инъекций и PHP/JSP/ASP SQL-инъекций).

Content Security Policy (CSP) представляет собой платформу аудита атак приложения, которая считается эффективной против большого количества уязвимостей. Данную технологию можно использовать для отправки запроса на HTTP и получения кластерных ответов HTTP. Если сайт считается защищённым, то с помощью технологии можно применять модули аутентификации для их сканирования.

SQLCo является технологией для выявления SQL-инъекции в БД веб-сайта, что может применяться на обширном диапазоне БД, поддерживает шесть видов способов инжекции SQL.

Secure contexts является технологией анализа безопасности сетевого трафика. Данный инструмент выполняет проверку приложения на неверные конфигурации и известные уязвимости TLS/SSL, осуществляет сканирование зашифрованных соединений и проверки SSL/TLS на уязвимость для атак в середине (MiTM).

Cross-Origin Resource Sharing является приложением, способным выявить «прорехи» веб-сайта применяя слабости веб-браузера. В противоположность другим технологиям безопасности Cross-Origin Resource Sharing может отслеживать атаки с клиентской стороны в пределах браузера. Эта технология разработана для функционирования с рядом веб-браузеров, чтобы контролировать изменения на веб-сайте.

Для проверки способности описанных выше технологий анализа исходного кода обнаруживать дефекты безопасности, требуется написать тест, содержащий в себе самые распространённые ошибки программистов. Для этой цели с официального веб-сайта Oracle был взят документ «Secure Coding Guidelines for Java SE», описывающий распространённые ошибки программирования, способные оказать влияние на безопасность программного продукта. Опираясь на этот документ, выполнено написание тестовых классов и проведение анализа (таблица 1.1).

В таблице представлен результат анализа. Из таблицы можно увидеть, что перечисленные выше технологии не смогли показать стопроцентный результат. По этим результатам технологию Cross-Origin Resource Sharing можно считать самой лучшей.

Таблица 1.1 - Итоги тестирования имеющихся технологий

Технология \ Критерий	SQL-инъекция	Незакрытый ресурс	Переполнение	Вызов нативного метода	Отсутствие обработки исключений	Результат
Referrer Policy Cont	1	1	0	0	1	3/5
Fetch	0	1	1	0	1	3/5
Wapiti	0	0	0	1	1	2/5
Content Security Policy (CSP)	0	1	1	0	0	2/5
SQLCo	1	0	0	1	0	2/5
Secure contexts	0	1	1	0	1	3/5
Cross-Origin Resource Sharing	0	1	1	1	1	4/5

Вывод заключается в том, что сегодня имеется большое количество средств и технологий, решающих общие вопросы кодирования. Практически не существует средств, которые непосредственно направлены на безопасности исходного кода, и, по всей вероятности, они не считаются открытыми и имеет большую стоимость, поэтому выполненное тестирование доказывает актуальность разработки.

2. ПРИМЕНЕНИЕ JAVA-ТЕХНОЛОГИИ ДЛЯ ЗАЩИТЫ WEB-ПРИЛОЖЕНИЙ ОТ УЯЗВИМОСТЕЙ НА ПРИМЕРЕ ОРГАНИЗАЦИИ

2.1. Характеристика и направление деятельности организации. Описание аппаратного и программного обеспечения

Деятельность компании ООО «Омега» – оказание сервисных услуг: транспортные, гостиничные, тех. обслуживания, общепита, в том числе облуживания дорожно-строительной техники.

Компания была создана в 90 –х годах прошлого века как союз консультантов по подбору персонала.

Отличительная особенность заключается в том, что наряду со стандартами, правилами, качеством работы у компании есть ценности. Ценности базируются на комплексе многосторонних отношений, в частности с Клиентами, Кандидатами, Конкурентами, где для нас важно и ценно следующее: честные и открытые взаимоотношения, взаимное доверие и уважение, качественная, эффективная работа, нацеленность на позитивные, конструктивные решения.

Организационная структура компании изображена на рисунке 2.1.

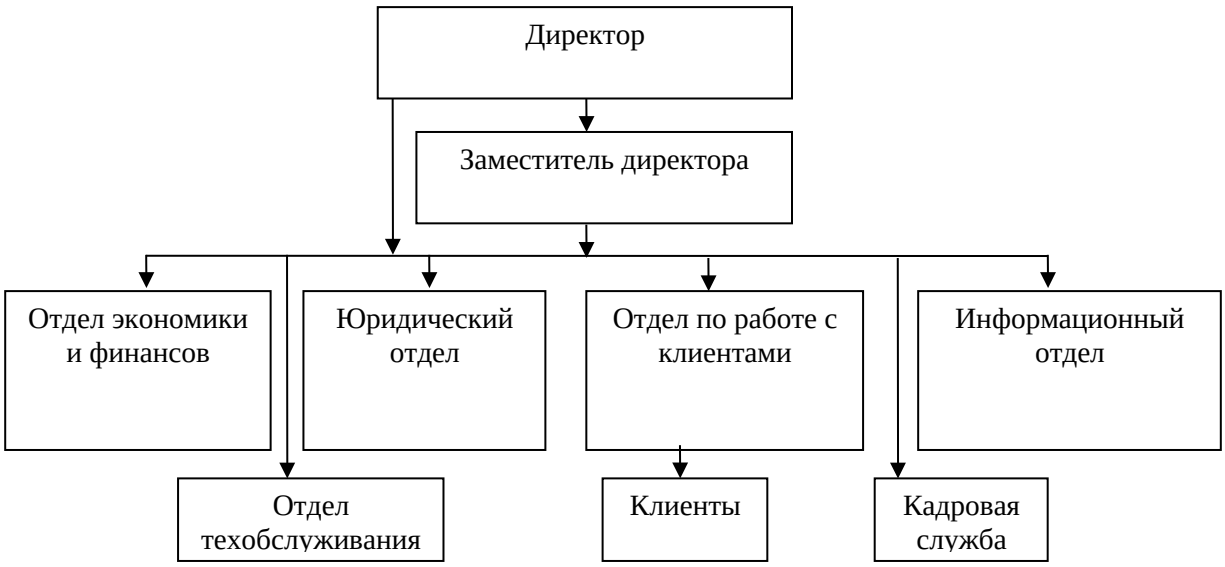

Рисунок 2.1 - Организационная структура предприятия

Для обеспечения качественного выполнения управленческих функций обязательным является умение эффективно принимать решения. Эффективность принятых решений часто зависит от таких факторов как своевременность, оперативность, достоверность, точность и полнота получаемой информации.

На рисунке 2.2 показана структурная схема технической архитектуры анализируемого предприятия [12].

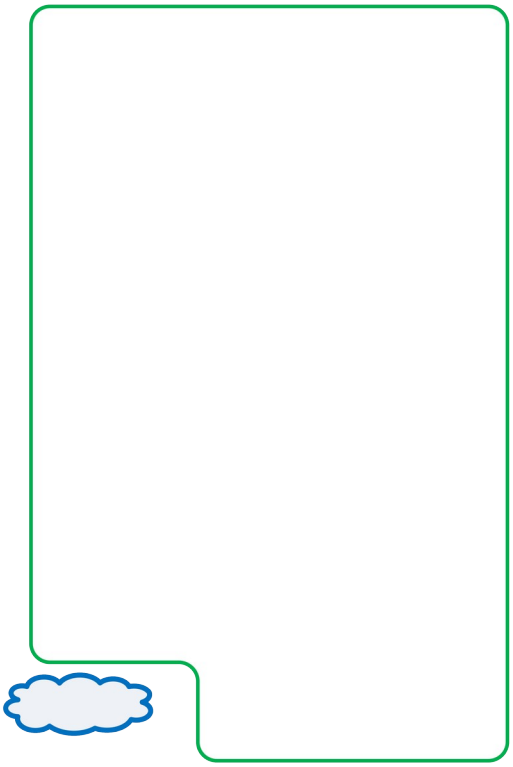

Рисунок 2.2 - Схема технической архитектуры предприятия

Как рабочие станции применяются модель OLDI Office 110 с такой конфигурацией:

o процессор Intel® Celeron 430 (Conroe-L, 3.60 ГГц, FSB800MHz, 1024Kb L2,Socket 775);

- жёсткий диск 500Gb (SATA III);
- оперативная память DDR III 2Гб;
- сетевая карта 2x10/100/1000 Мбит/с;
- видеокарта интегрирована в чипсет iG31;
- звуковая карта 8-канальная;
- размеры 520 мм x 210 мм x 340 мм.

Из всех серверов на предприятии применяются только сервер базы данных и файл–сервер. Как аппаратная основа серверов применяется Office Server TX 3000R15, имеющий такие технические характеристики:

- процессор Intel® Core™5 750 (Lynnfield, 3.32ГГц, 16Мб, LGA1156);
- чипсет Intel 3420;
- жёсткий диск 2000Gb (SATA III, 14400rpm, 64Mb);
- оперативная память DDR3 8192 Mb (pc-22120) 2666MHz;
- оптические накопители DVD±RW;
- видеокарта XGI® Z9s (64MB DDR2);
- сетевая карта 6x10/100/1000 Мбит/с;
- тип корпуса Full Tower.

Пользовательские сегменты и сервера посредством маршрутизатора 3Com13701-ME объединяются в сеть. Внутри сегментов пользователей как коммутаторы применяются 3COM Gigabit Switch 8 3CGSU08. Эти типы коммутаторов снабжены программными брандмауэрами. [53]

Телефонная сеть построена с применением АТС Panasonic KX-TEM824.

Как оконечные устройства применяются офисные телефоны Panasonic KX-T7710.

Как среда передачи данных применяется кабель витая пара 5 категории (100BASE-TX) со скоростью передачи до 100 мбит/сек.

Каждая из рабочих станций комплектуется МФУ Xerox Phaser 3100MFP/X.

На рисунке 2.3 показана наглядно программная архитектура ИС.

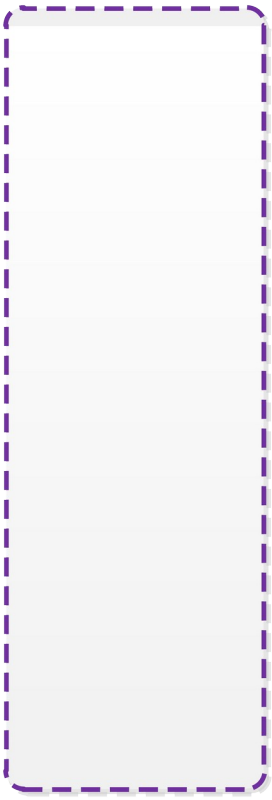

Рисунок 2.3 - Программная архитектура предприятия

Все ПК снабжены стандартным пакетом ПО:

- ОС Windows 7 Pro x64;
- Браузер Mozilla Firefox;
- Пакет MS Office 2007;
- Avira AntiVir Premium Security Suite;
- Adobe Acrobat 9.0 Professional.

Принимая во внимание специфику деятельности отделов, ПК предприятия обеспечиваются дополнительными программными обеспечениями (персональными пакетами ПО для отделов):

- Отдел IT: Radmin Viewer 3, Remote Administrator v2.2, Total Commander;
- Бухгалтерия: 1С Бухгалтерия: Версия 8.

2.2. Анализ уязвимостей web-приложений на примере организации

Проведем тестирование уязвимостей веб-приложений на предприятии.

Как тестовый стенд будет использоваться виртуальная машина Oracle VM VirtualBox, на которой установлена ОС Microsoft Window 7 x64.

Также будет необходимо веб-приложение, содержащее различные виды уязвимостей. Таким приложением будет Mutillidae. Выбор сделан по таким причинам:

1. Подробная документация;

2. Простота развёртывания;

3. Этот дистрибутив имеет наиболее свежую дату обновления, в противоположность остальным аналогичным продуктам.

Когда тестовый стенд подготовлен, можно перейти к процессу тестирования, который является следующей последовательностью действий:

1. Подготовка тестируемого приложения к сканированию;

2. Настройка средства безопасности;

3. Запуск процесса сканирования с избранными настройками;

4. Анализ итогов и внесение их в таблицу (таблицы 2.1-2.4).

После того как все серии тестирования для всех средств проведены, необходимо внести в сводную таблицу полученные результаты – таблица 2.5.

При выполнении тестов можно столкнуться с такими проблемами:

1. Framework могут отличаться своим функционалом и настройкой. Для отражения в тестировании специфичностей средств тестирования, необходимо провести несколько серий сканирования с разными конфигурациями настроек, чтобы получить более качественный результат (если такое возможно).

2. Framework могут специализироваться на конкретном виде уязвимости или могут обнаруживать обширный диапазон уязвимостей. Когда Framework специализируется на конкретном виде уязвимости, значит, он обязан определять их более качественно, в обратном случае следует акцентировать особенное внимание на ложно идентифицированной или не

идентифицированной уязвимости при помощи специализированного средства тестирования.

3. Виды уязвимостей. Поскольку имеется огромное количество видов уязвимостей, следует определить те типы, которые будут отражены в окончательном отчёте. В этом вопросе необходимо руководствоваться классификацией OWASP Top 10 2017. Из этого списка следует выбрать 5 видов уязвимостей.

4. Количество уязвимостей в веб-приложении. Заблаговременно узнать, какое число уязвимостей содержится в тестируемом веб-приложении невозможно, поэтому за общее количество примем сумму обнаруженных всеми Framework.

Metasploid Framework. Этот Framework является полностью автоматизированным инструментом с консольным интерфейсом и имеет небольшое количество настроек, при помощи команды skipmetas -h можно ознакомиться с полным их перечнем. Чтобы запустить процесс сканирования предлагается 3 базовых варианта:

1. skipmetas -W /dev/null -LV […other options…] – в этом режиме запуска средство осуществляет упорядоченный обход цели и в общем работает подобно другим средствам. Не рекомендован из-за недостаточного охвата ресурсов цели, однако сам процесс сканирования занимает гораздо меньше времени в сравнении с другими режимами;

2. skipmetas -W dictionary.wl -Y […other options…] – в этом режиме средство применяет фаззинг исключительно имён расширений или файлов. Такой режим считается предпочтительным, если имеется ограничение по времени и к тому необходимо получить желаемый итог;

3. skipmetas -W dictionary.wl […other options…] – в этом режиме средство выполняет перебор всех возможных пар расширение и имя. Такой режим гораздо медленнее предшествующих, однако при этом он осуществляет наиболее подробный анализ веб-приложения.

Прислушаемся к рекомендации и используем 3-ий вариант запуска,

добавив при этом необходимые опции: skipmetas -W dictionary.wl -o ~/report/ http://target/

-W – указываем путь к словарю, который будет использоваться;

-o – указываем директорию, в которой будет сохраняться отчет (таблица 2.1).

Таблица 2.1 - Итоги сканирования Metasploid Framework

Тип уязвимости	Найдено	Ложно найдено	Время
Инъекции	6	5	3ч18м
Вопросы аутентификации и проверки сессий	11	2	
XSS	1	1	
Вопросы контроля доступа	3	2	
Неправильная конфигурация	128	0	

Metasploid Framework хорошо решил поставленную задачу, невзирая на множество неверных срабатываний. Следует отметить, что, несмотря на то, что средство не располагает графическим интерфейсом, он крайне простой в настройке. Кроме результатов, предоставленных в таблице, Metasploid Framework выявил большой объем интересной информации о веб-приложение, выполнив ее анализ можно значительно повысить защищённость веб-приложения.

OneSQL. Основная цель этого средства заключается в автоматическом поиске и эксплуатация SQL уязвимостей. Это средство имеет огромное количество настроек, которые дают возможность оптимизации процесса обнаружения и эксплуатации уязвимостей. Чтобы запустить сканирование можно применить визард: onesql —wizard. Попытаемся полностью автоматизировать процесс обнаружения и максимизировать итог (таблица 2.2).

OneSQL представляет собой специализированный инструмент, предназначенный для обнаружения и эксплуатации SQL уязвимостей. OneSQL отлично справился с такой задачей, несмотря, что полная автоматизация не показала такого же результата. Следует заметить, что

длительность тестирования достаточно продолжительная, принимая во внимание то, что поиск осуществлялся исключительно одного вида уязвимостей. Настройка этого средства без досконального изучения его богатого перечня опций считается самой сложной из всех перечисленных в предоставленной работе Frameworkов.

Таблица 2.2 - Итоги сканирования OneSQL

Тип уязвимости	Найдено	Ложно найдено	Время
Инъекции	14	0	4ч27м
Вопросы аутентификации и проверки сессий	—	—	
XSS	—	—	
Вопросы контроля доступа	—	—	
Неправильная конфигурация	—	—	

BDD-Security. Framework располагает консольным и графическими интерфейсами. Чтобы запустить сканирование необходимо использовать визард, где будет предложение применить специально сконфигурированные и стандартные настройки.

Начнём настраивать средство тестирования:

1. Сначала будет предложено ввести адрес цели, или выбрать файл, содержащий структуру веб-сайта, следует выбрать первый вариант

2. Второй шаг состоит в выборе профиля сканирования; их довольно много, остановим выбор на профиле «Default», поскольку в нем содержатся тесты для обнаружения всех доступных видов уязвимостей и настройки сканирования, которые будут оставаться дефолтными.

3. На последующем шаге Framework стремится определить технологии, используемые целью, и отражает приобретённые значения, которые можно выбрать самостоятельно либо проставить значение «Unknown». В данном случае все без исключения значения были определены правильно, и они остаются без изменения

4. На четвёртом шаге предлагается сделать выбор способа аутентификации, поскольку она не понадобится, следует этот шаг

пропустить.

5. Последний шаг предлагает все настройки сохранить и после нажатия «Finish» начинается сканирование (таблица 2.3).

Таблица 2.3 - Итоги сканирования BDD-Security

Тип уязвимости	Найдено	Ложно найдено	Время
Инъекции	1	0	2ч 13м
Вопросы аутентификации и проверки сессий	31	0	
XSS	19	0	
Вопросы контроля доступа	4	3	
Неправильная конфигурация	231	0	

Минимум ошибочных срабатываний и простота в настройке — так можно охарактеризовать данный инструмент. Кроме результатов, внесённых в таблицу, BDD-Security собрал большой объем информации, которая относится к структуре приложения и пользовательских конфиденциальных данных. Применяя одну из небесполезных фич предоставляется возможность выделения предоставления детальной информации о уязвимости и способах её ликвидации, а также ссылок на ресурсы, которые содержат исчерпывающую информацию.

Web Application Attack and Audit Framework (w3af). Фреймворк с консольным и графическим интерфейсом, предоставляющий возможность обнаружения и эксплуатации уязвимостей в приложении. Располагая широким спектром плагинов, позволяет довольно тонко осуществить настройку процесса сканирования. К тому же w3af располагает готовыми шаблонами для сканирования, пользователь должен ввести лишь адрес цели.

Настраивая средство тестирования, будем опираться на шаблоны «full_audit_spider_man» и «full_audit», они отличаются тем, что первый шаблон как плагин краулер использует spider_man – локальный прокси, а второй web_spider – классический веб-паук. Для целей предоставленной работы плагины из группы «bruteforce» не понадобятся. Они включены по умолчанию в данных шаблонах, поэтому их следует отключить. Остаётся

выполнить настройку плагинов из группы «output». Выводится собранная информация по умолчанию исключительно в консоль, что в целях анализа итогов считается не очень удобным, поэтому следует включить плагин «html_file», позволяющий сохранять в HTML-файл всю полученную информацию.

Теперь следует ввести адрес цели и приступить к сканированию (таблица 2.4).

Таблица 2.4 - Итоги сканирования w3af

Тип уязвимости	Найдено	Ложно найдено	Время
Инъекции	4	2	2ч57м
Вопросы аутентификации и проверки сессий	3	1	
XSS	25	0	
Вопросы контроля доступа	4	3	
Неправильная конфигурация	17	0	

Данный инструмент не зря считается фреймворком, при соответствующих навыках настройки он может осуществлять сбор исчерпывающей информации о цели за оптимальное время. Однако и он имеет недостатки, при тестировании выявлена проблема со стабильностью работы средства, что доставило огорчение.

Таблица 2.5 - Итоговая таблица результатов

Средства	Инъекции		Вопросы аутентификации и проверки сессий		XSS		Вопросы контроля доступа		Неправильная конфигурация		Затраченное время
	Обнаружено	Ложно обнаружено	Обнаружено	Ложно обнаружено	Обнаружено	Ложно обнаружено	Обнаружено	Ложно обнаружено	Обнаружено	Ложно обнаружено	
Metasploid Framework	6	5	11	2	1	1	3	2	128	0	3ч 18м
OneSQL	14	0	—	—	—	—	—	—	—	—	4ч 27м
BDD-Security	1	0	31	0	19	0	4	3	231	0	2ч 13м
W3AF	4	2	3	1	25	0	4	3	17	0	2ч 57м

Всего	14	32	25	3	231	

Рассмотрим далее сравнение данных интегрированных средств тестирования по критериям, предложенным в первой главе работы – таблица 2.6.

Таблица 2.6 – Сравнение по критериям

№	Средство тестирования \ Критерий сравнения	Metasploid Framework	OneSQL	BDD-Security	W3AF
1.	Сопоставление стоимости продукта относительно оценкам критериев.	1	1	0	1
2.	Универсальность использования средств тестирования уязвимостей.	1	0	0	1
3.	Число поддерживаемых векторов атаки: тип и число активных плагинов средства тестирования уязвимостей.	1	0	0	1
4.	Точность выявления CSS.	0	1	0	1
5.	Точность выявления SQL-инъекций.	0	1	1	1
6.	Точность обхода структуры веб-приложения и выявления локальных файлов.	1	0	1	0
7.	Удаленное применение файлов, XSS, фишинг посредством RFI.	1	0	1	0
8.	WIVET-сравнение.	1	0	0	1
9.	Адаптивность технологии.	0	1	0	1
10.	Сопоставление особенностей аутентификации.	1	0	0	1
11.	Число дополнительных возможностей сканирования и интегрированных механизмов.	1	0	0	1
12.	Общее представление о работе ключевой функции сканирования.	0	1	0	1
Итого		8	5	3	10

Когда появляются задачи, касающиеся проведения аудита безопасности приложения, необходимо запастись временем и терпением. Из таблицы, которая представлена ниже можно увидеть время, занимающие процессом сканирования каждым инструментом. Время считается не весьма большим, причина состоит в следующем: первое, средства безопасности и тестовое приложение находились на одной и той же физической машине, второе, это время в себя не включает процесс анализа приобретённых результатов, в настоящих условиях время, затраченное на тестирование будет значительно

больше. Можно заметить, что итоги, предоставленные рассмотренными средствами, отличаются: какой-то из средств лучше справился с обнаружением той либо другой уязвимости, какой-то смог предоставить более доскональную информацию о веб-приложении в общем. Вследствие этого в процессе аудита безопасности не надо полагаться исключительно на один инструмент, необходимо применять комплекс всевозможных средств, включая и анализ веб-приложения в ручном режиме. Стоит также отметить, что развитие веб-технологий происходит крайне стремительно, поэтому интегрированные средства безопасности за их развитием не поспевают, вследствие этого перед тем, как проводить аудит необходимо досконально изучить технологии, применяемыми в тестируемом приложении для более точного подбора комплекса методик и инструментов.

По результатам анализа таблицы видно, что лучшими framework для тестирования web-уязвимостей являются более универсальные средства, такие как W3AF, но менее универсальные средства хоть и показали низкую общую оценку гораздо лучше тестируют определенные веб-уязвимости, поэтому, как говорилось ранее, для оптимального тестирования веб-приложений лучше всего использовать совмещение нескольких средств тестирования что доказывает актуальность работы в направлении расширения работу одного из вышеописанных framework.

2.3. Мероприятия по обеспечению защиты web-приложений с использованием Java-технологий в организации

Чтобы создать эффективную методику, предоставляющую возможность тестирования на проникновение веб-приложения по перечню угроз и уязвимостей, которые указаны в рейтинге рисков OWASP Top - 10 требуется:

- выработать требования к методике, которая разрабатывается для тестирования на проникновение веб-приложения на наличие каждого из десяти рисков из списка OWASP Top - 10;

- провести выбор необходимых этапов из имеющихся методик, которые позволяют проводить тестирование на проникновение веб-приложения;

- выработать методику по тестированию на проникновение веб-приложения с определением стадий тестирования и программных средств, позволяющих осуществлять проверку наличия каждого из десяти рисков перечня OWASP Top - 10.

Методика по тестированию на проникновение веб-приложений согласно рейтингу рисков и уязвимостей OWASP Top - 10 обязана соответствовать таким требованиям:

- методикой должна учитываться необходимость в начальном сборе информации об открытых портах на сервере, запущенных на сервере службах, версию и тип веб-сервера, тип операционной системы, тип межсетевого экрана;

- методикой должна учитываться необходимость в тестировании веб-приложения на наличие элементов, которые могут в себе содержать известные уязвимости;

- методикой должна учитываться необходимость в тестировании средств по проверке входных данных в веб-приложения на наличие уязвимостей, которые могут повлечь за собой вставку инструкций (вставку SQL, XML - инструкций, межсайтовое выполнение сценариев);

- методикой должна учитываться проверка средств аутентификации на возможность осуществления угроз некорректной аутентификации;

- методикой должна учитываться проверка веб-приложения на потенциальную утечку конфиденциальных данных;

- методикой должна предоставляться возможность проведения проверки корректности контроля доступа;

- методикой должна учитываться необходимость в проверке веб-приложения на наличие уязвимостей в элементах, которые отвечают за десериализацию объектов;

- методикой должна учитываться необходимость в проверке элементов приложения, которые отвечают за ведение журналов и мониторинг;

- необходимо после реализации необходимых этапов тестирования сформировать отчёт об итогах тестирования.

С учётом требований, которые методика обязана содержать и рекомендаций в имеющихся методиках, считается целесообразным разделить тестирование на проникновение на такие этапы:

- сбор информации о системе;

- тестирование элементов веб-приложения, в которых могут содержаться известные уязвимости;

- тестирование средств по проверке входных данных в веб-приложении;

- тестирование средств аутентификации;

- тестирование возможности утечки конфиденциальных данных;

- тестирование средств по контролю доступа

- тестирование веб-приложения на наличие элементов, которые отвечают за десериализацию объектов;

- тестирование средств ведения журналов событий и мониторинга;

- подготовка отчёта.

Рассмотрим реализацию для расширения функциональных тестов веб-приложений с использованием Java-технологий в организации.

Данный этап служит для определения информации о веб-приложении, запущенных службах, открытых портах, версии веб - сервера. Собранная информация свидетельствует об объёме информации, к которой может иметь доступ злоумышленник, который ее может использовать для поиска записей о имеющихся уязвимости в базах данных уязвимостей и реализовать атаку на веб-приложение.

Для реализации этапа по сбору информации был избран сканер Metasploit Framework, который входит в состав ОС Kali Linux. Зависимо от заданных опций программой формируются соответственные выходные

данные. Сканирование при помощи Metasploit Framework осуществляется после того, как заданы специальных команд, общий синтаксис которых можно представить таким образом: Metasploit Framework [<Тип сканирования>] [<Опции>] {<Веб - адрес цели сканирования> }

Metasploit Framework применяется для:

- определения типа операционной системы;
- определения типа межсетевого экрана;
- определения версии и типа веб-сервера;
- сканирования запущенных служб на сервере;
- проверки открытых портов на сервере.

На рисунке 2.4 показан пример работы Metasploit Framework.

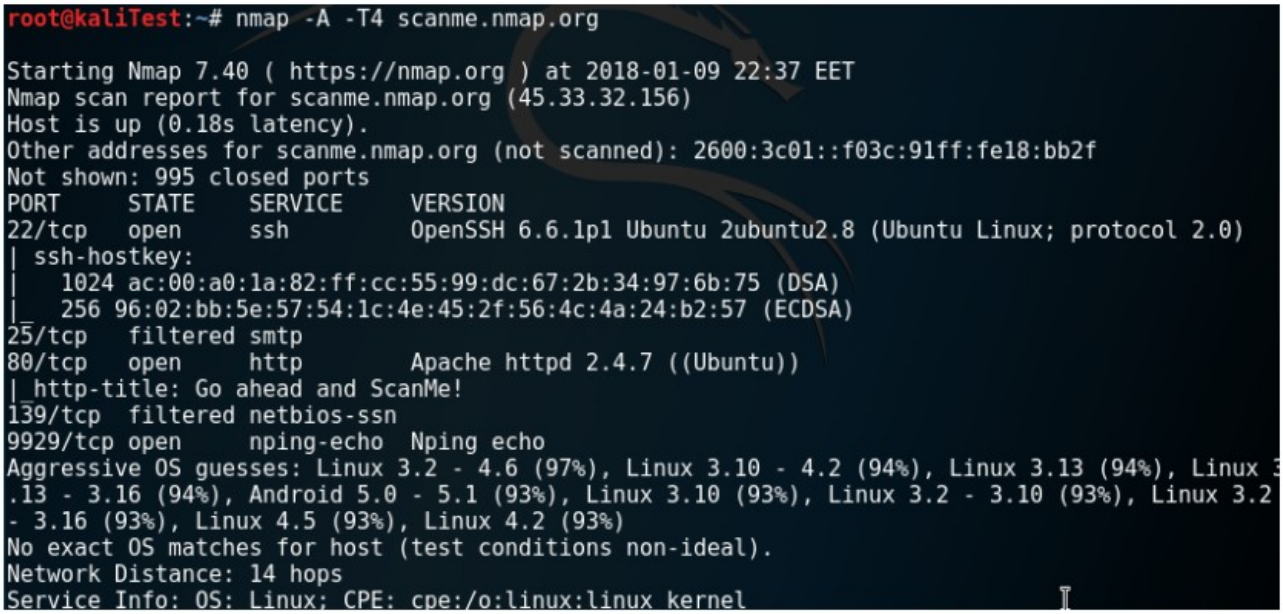

Рисунок 2.4 - Результат работы сканера Metasploit Framework

После того, как информация собрана, необходимо осуществить проверку наличия эксплойтов для операционных систем и веб - серверов в соответствии с версиями, для чего предлагается применять утилиту searchsploit, которая входит в состав ОС Kali Linux. Данная утилита применяет базу данных эксплойтов Exploit Database, которую можно использовать в целях проверки вероятности атаки посредством имеющихся уязвимостей.

Обнаруженные эксплойты реализовываются программой Metasploit Framework, которая входит в состав Kali Linux и предоставляет возможность:

- имитирования сетевых атак;

- выявления уязвимостей системы;

- проверки эффективности работы систем по обнаружению вторжений;

- разработки новых эксплойтов;

- создания отчётов по итогам работы.

Запуск Metasploit Framework выполняется командой msfconsole. Важнейшими командами Metasploit Framework считаются:

- use - выбор конкретного эксплойта для работы;

- back – команда, предназначенная, чтобы прекратить работу с избранным модулем;

- show - вывести перечень модулей конкретного типа;

- set - установить значение конкретного объекта;

- run - запуск вспомогательного модуля после того, как установлены требуемые опции;

- info - вывод информации о модуле;

- search - найти конкретный модуль;

- check - проверить возможность целевой системы к уязвимости;

- sessions - вывод списка доступных сессий.

Пример поиска эксплойтов, которые касаются веб - серверов Apache реализовывается при помощи команды: searchsploit apache. На рисунке 2.5 показаны итоги работы утилиты searchsploit.

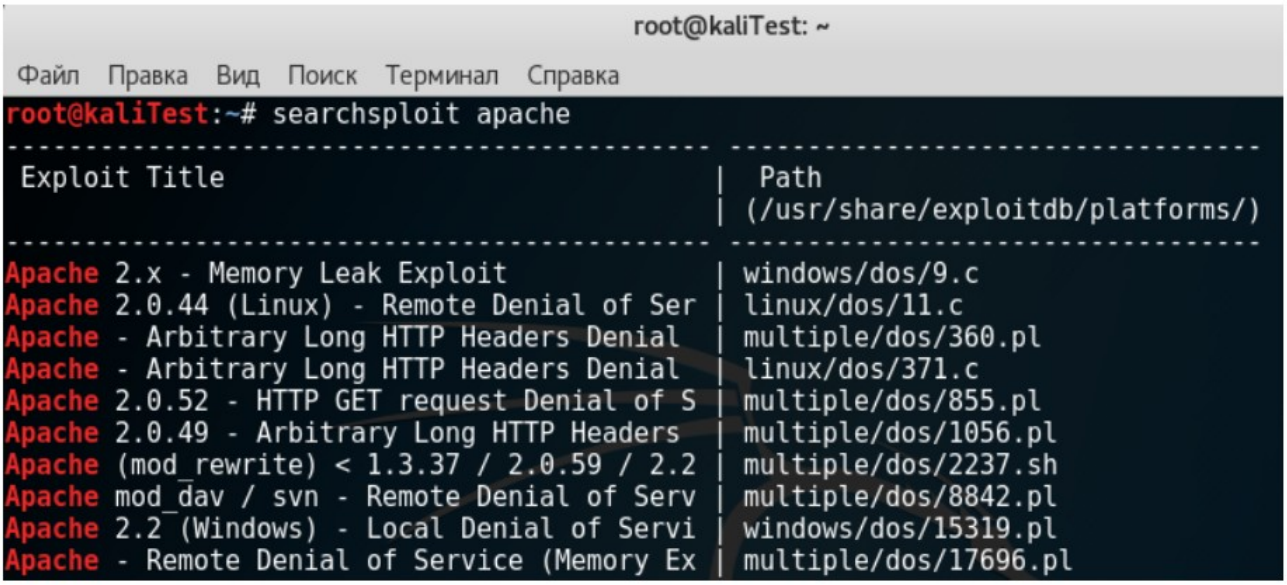

Рисунок 2.5 - Результат работы утилиты searchsploit

В качестве примера работы программы Metasploit Framework служит поиск и эксплуатация уязвимости при помощи эксплойта в системе по управлению контентом WordPress. Поиск следует начинать с запуска Metasploit Framework при помощи команды msfconsole, а поиск модулей, содержащих информацию об уязвимости WordPress, выполняют при помощи команды search wordpress. На рисунке 2.8 показан результат поиска модулей.



Рисунок 2.6 - Результат работы Metasploit Framework по поиску модулей, описывающих уязвимости WordPress

В соответствии с рисунком 2.6, для атаки на WordPress взят модуль auxiliary / scanner / http / wordpress_login_enum, отвечающий за взлом процесса аутентификации способом грубой силы.

Указанный модуль выбирается для работы при помощи команды:

use auxiliary/scanner/http/wordpress_login_enum.

Информацию о специфичностях применения данного модуля можно получить посредством команды:

info use auxiliary/scanner/http/wordpress_login_enum

Следующий этап состоит в указании адреса, на которой располагается веб – сайт, для чего применяется команда set RHOSTS <адреса_веб - сайта>. Помимо этого, требуется указать путь к файлу с паролями, перебор которых выполняется: set PASS_FILE /root710k-common-passwords.txt. Процесс перебора паролей по словарю при помощи Metasploit Framework представлен на рисунке 2.7.

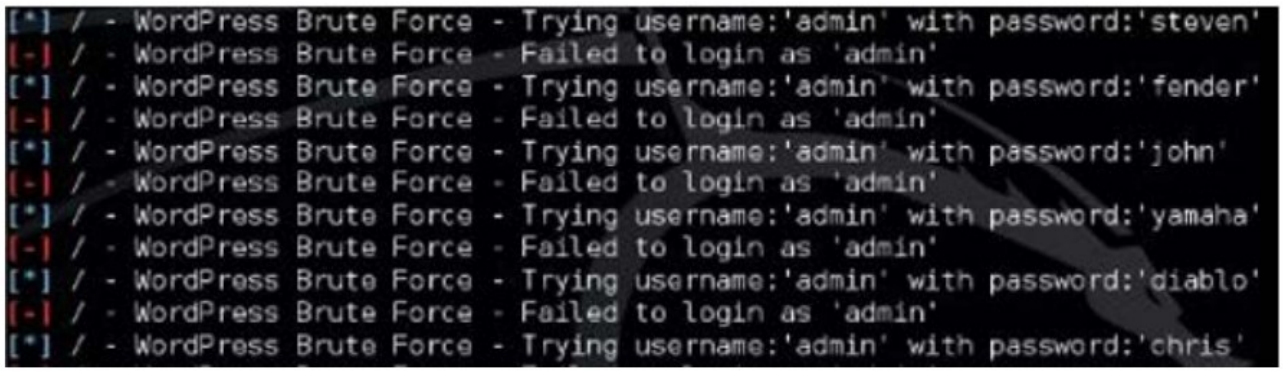

Рисунок 2.7 - Процесс перебора паролей по словарю при помощи Metasploit Framework

После выполнения перебора паролей из словаря в случае нахождения совпадения, будет отображаться информация о завершившейся успехом аутентификации, которая показана на рисунке 2.8:

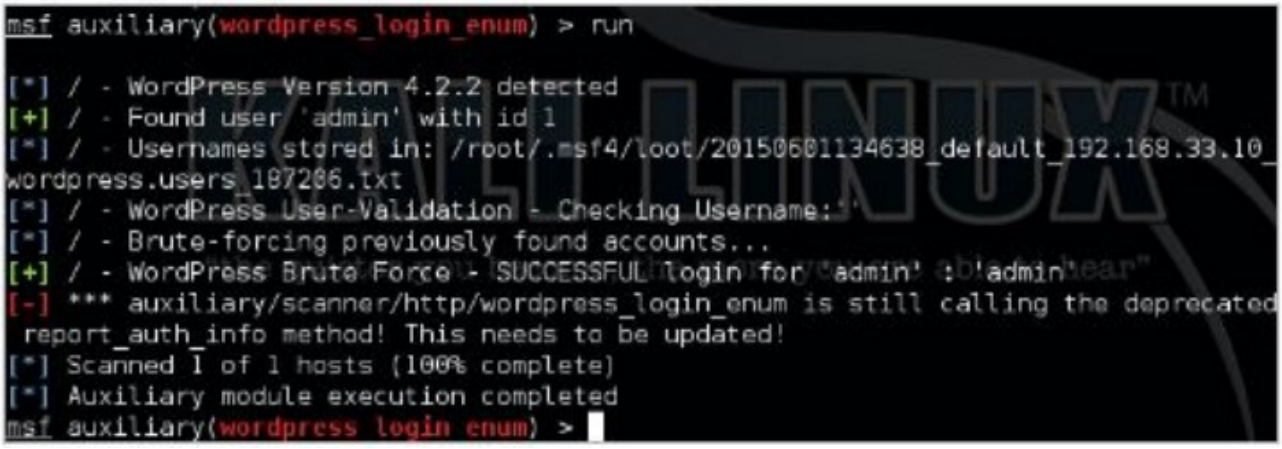

Рисунок 2.8 - Информация о найденном пароль способом перебора по словарю

Для тестирования средств по проверке входящих данных в веб-приложении необходимо выполнить такие действия:

- проверить параметры GET, POST в HTTP - запросах;

- проверить возможность переполнения буфера памяти, а также проверить препятствуют ли имеющиеся средства веб-приложения его переполнению;

- проверить, используется ли в веб-приложении единственный компонент управления проверкой ввода для любого из принятых типов данных.

- протестировать возможность регистрации ошибки проверки входных данных на стороне сервера

- протестировать возможность вставки SQL, XML - инструкций, межсайтовой реализации сценариев.

В первую очередь необходимо найти входные точки веб-приложения, которые могут считаться отправными точками для вероятных атак. В качестве точек входа данных могут служить части веб-приложения, выполняющие обработку HTTP - запросов (способы POST и GET), сокеты, порты. Необходимо также найти все параметры, которые передаются в HTTP - запросов, чтобы определить возможные места, где злоумышленник имеет возможность манипулировать данными. Чтобы провести тестирование на проникновение по этому этапу необходимо:

- выяснить, где применяются запросы GET и где применяются POST;

- найти все параметры, которые используются в запросе POST;

- найти все параметры, которые используются в запросе GET;

- найти все параметры строки запроса, многочисленные параметры могут находиться в одной строке и разделяться символами &, ~ ,: либо кодированием или каким-нибудь другим специальным символом.

В качестве примера GET и POST - запросов могут служить:

GET

https://x.x.x.x/shoppingApp/buyme.asp?CUSTOMERID=100

&ITEM=z101a&PRICE=62.50&IP=x.x.x.x

POST https://x.x.x.x/Avv/authenticate.asy?service=login

Анализ GET и POST - запросов может выполняться модулем Burp Proxy, который является компонентом программы Burp Suite. Burp Proxy представляет собой прокси-элемент, предоставляющий возможность перехвата и проверки трафика между браузером и веб-приложением. Пример работы Burp Proxy, который перехватывает GET – запросы показан на рисунке 2.9.

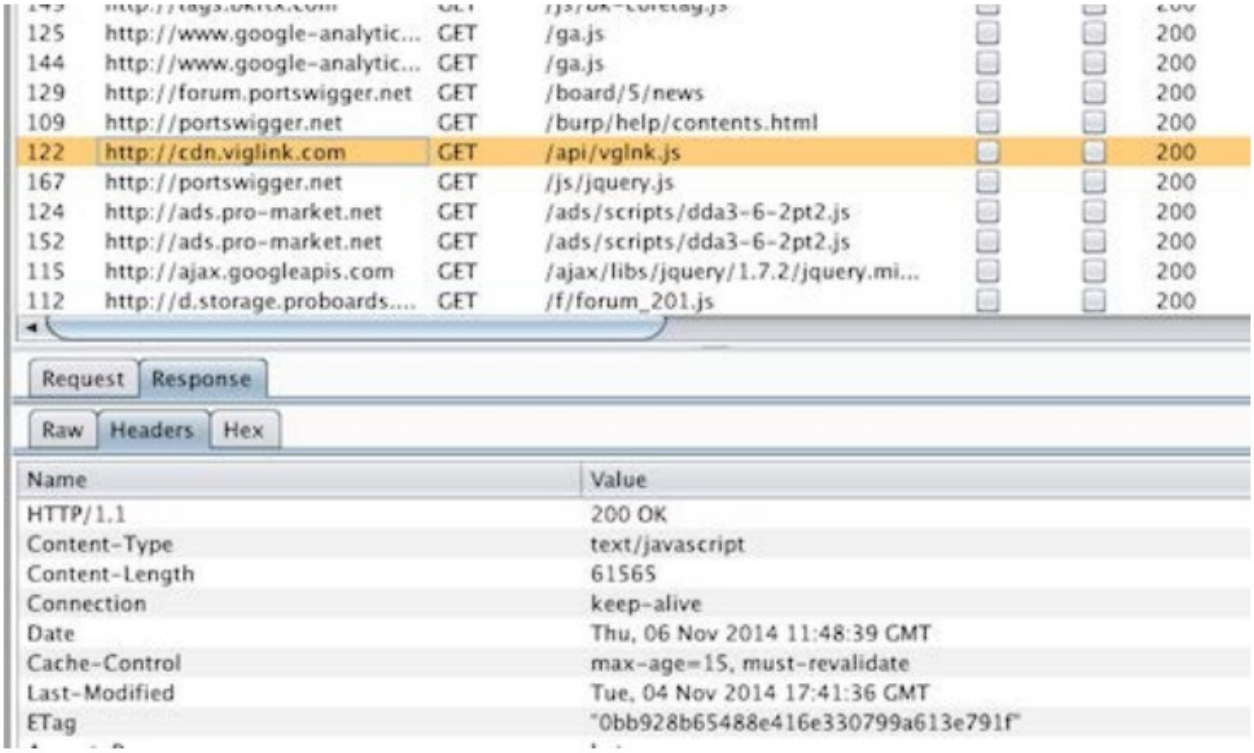

Рисунок 2.9 - Пример работы Burp Proxy по анализу HTTP - запросов

Чтобы провести проверку возможности вставки SQL - инструкций, необходимо сначала определить тип URL. Имеется множество разнообразных шаблонов URL, к примеру:

nurl:item_id=

nurl:readnews.php?id=

inurl:news.php?id=

inurl:shopping.php?id =

URL веб-приложения может быть следующего вида:

http://www.example.com/page.php?id =28

В конец URL необходимо добавить символ '', чтобы строка имела такой вид:

http://www.example.com/rubrika.php?id=28'

Если выполнить переход по ссылке, то может появиться окно с сообщением, что такой страницы не существует и с информацией об ошибке, которая имеет вид:

java.sql.SQLException: ORA-00933: SQL command not properly ended at oracle.jdbc.dbaaccess.DBError.throwSqlException(DBError.java:180) at oracle.jdbc.ttc7.TTIoer.processError(TTIoer.java:208)

Такие данные могут говорить о наличии уязвимостей в процессе обработки ввода информации пользователем. Последующие действия следует выполнять утилитой onesql. Onesql предоставляет такие возможности:

- поддержки БД: Microsoft Access, PostgreSQL, Oracle, MySQL, Firebird, SQLite, IBM DB2;

- поддержки 6-ти способов SQL - инструкций;

- поддержки подключения к БД;

- поддержки перечня пользователей, ролей, привилегий, хеш-паролей, БД, столбцов и таблиц.

Определяются существующие БД, которые применяются веб-приложением командой -u и --dbs:

onesql -u http://www.example.com/page.php?id=28--dbs .

На рисунке 2.10 представлен пример результата работы onesql.

Из рисунка 2.10 можно увидеть, что применяется СУБД MySQL 5, ОС веб - сервера FreeBSD, Apache 2.2.22, а также таблицы safecosmetics и information_schema. Последующие проверки могут состоять в проверке возможности доступа к записям в таблицах, а также в поиске эксплойтов для версий обнаруженного программного обеспечения. Остальные команды могут быть доступными по ссылке разработчика onesql. Поиск эксплойтов осуществляется посредством Metasploit Framework.

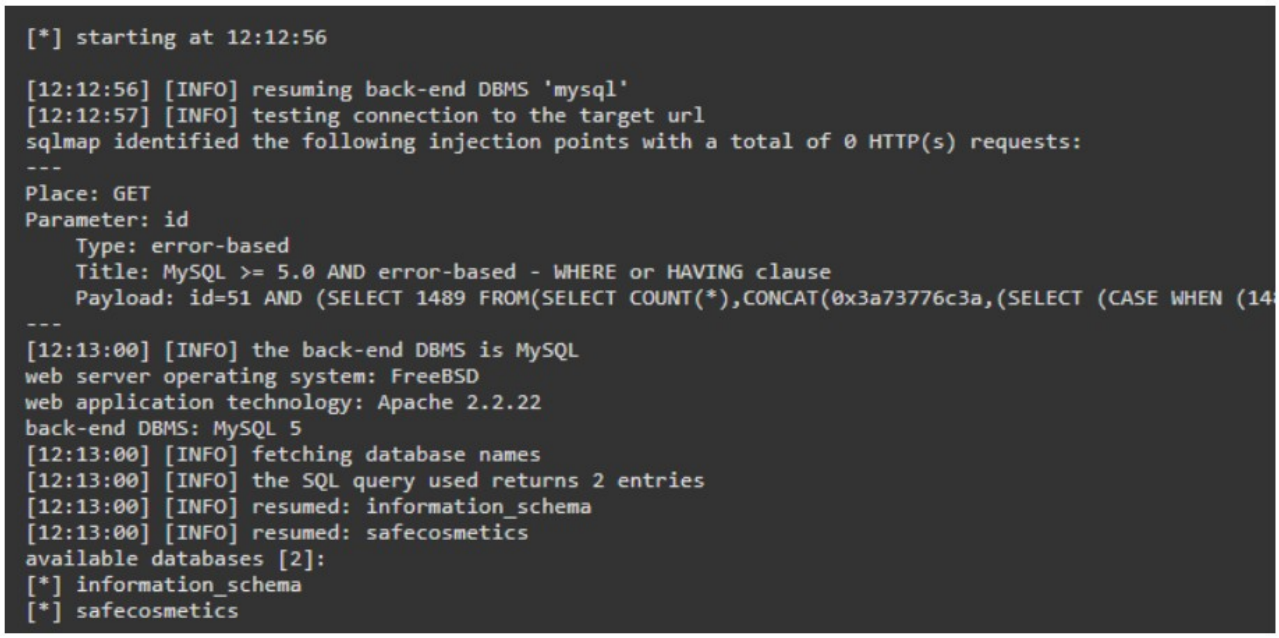

Рисунок 2.10 - Результата работы onesql

Проверка возможности межсайтовой реализации сценариев состоит в отправке в веб-приложение данных, которые должны или подвергаться проверке, или являться причиной нарушения логики работы веб-приложения. В качестве примера подобных данных могут послужить заполнение полей в форме регистрации, которую заполняет пользователь, а также их отправка на сервер. Данные для заполнения могут иметь такой вид: <script> alert (1) </ script>. Если имеются уязвимостей, которые приводят к межсайтовой реализации сценариев, то на стороне пользователя будет отображаться окно, показанное на рисунке 2.11.

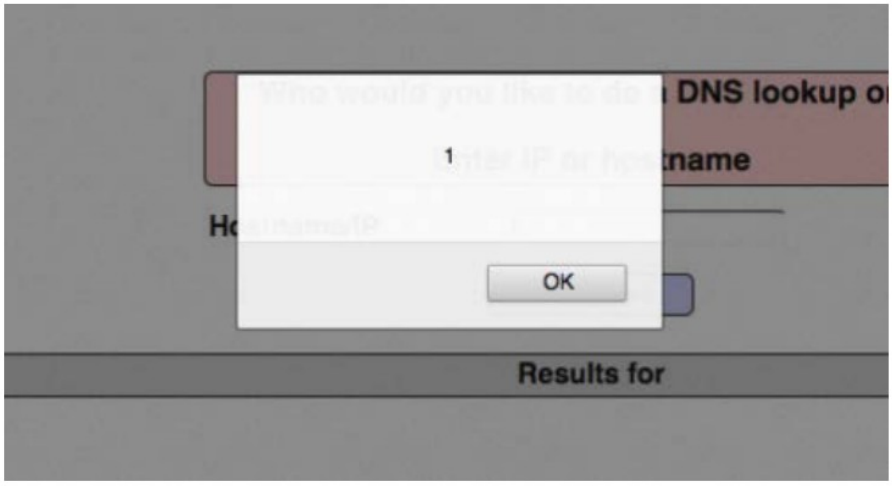

Рисунок 2.11 - Результат ответа от сервера при наличии уязвимостей межсайтовой реализации сценариев

Тестирование на возможность межсайтовой реализации сценариев осуществляется при помощи Burp Suite. Из рисунка 2.12 можно увидеть, что с веб-сайта разработчика Burp Suite, программа предоставляет возможность задачи необходимых параметров для запроса, а также отправки запроса на сервер.

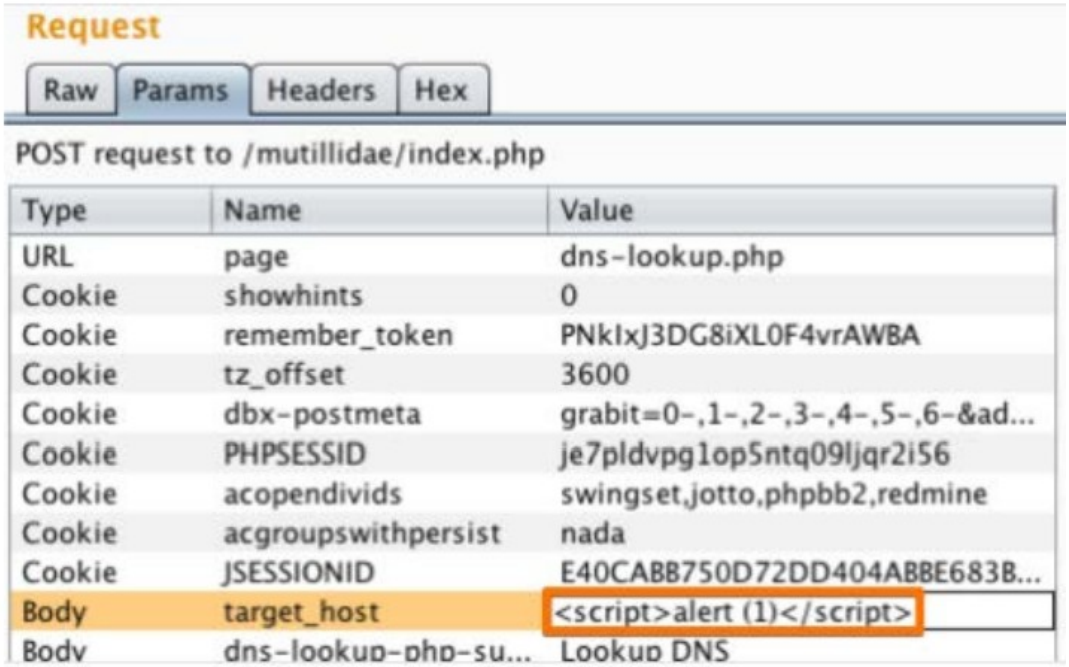

Рисунок 2.12 - Окно отображения параметров, передающихся в веб-приложение при помощи программы Burp Suite

При тестировании веб-приложения на этом этапе учитываются данные, которые были получены в процессе выполнения предшествующего этапа -

тестирования средств по проверке входных данных в веб-приложение. Для дополнительных проверок необходимо выполнить:

- проверку применения HTTPS - протокола в процессе работы с GET и POST – запросами, чтобы определить возможность перехвата данных;

- проверку наличия и функционирования механизмов защиты при входе от способов перебора паролей (нахождение порогового значения безуспешных попыток набора пароля);

- проверку механизмов по разблокировке акаунта администратора;

- определить период блокирования доступа по причине неверного ввода пароля;

- проверку того, необходима обязательная аутентификация перед доступом к абсолютно всем ресурсам и страницам, исключая те, для которых не требуется предыдущая аутентификация;

- проверить возможность замены параметров в запросах;

- проверить реализацию механизмов назначения идентификаторов сессии;

- проверить то, какие символы запрещаются и разрешаются для паролей (надо ли пользователю использовать прописные и строчные буквы, специальные символы и цифры);

- проверить то, как часто пользователь имеет возможность замены своего пароля;

- проверку того может ли пользователь использовать пароль повторно;

- проверить наличие в веб-приложении функции двухфакторной аутентификации, обеспечивающую защиту от распознавания пароля и логина;

- проверить то, что функция изменения пароля включает старый пароль, новый пароль, а также его подтверждение;

- проверить все ли попытки аутентификации система регистрирует.

Чтобы реализовать тестирование на проникновение на этом этапе необходимы такие средства: Burp Suite, Metasploit Framework.

В конфиденциальных данных могут содержаться данные об определённых программах, такие как реквизиты работников, финансовые записи, номера кредитных карточек, учётные данные, БД и записи в них. Значит, появляется необходимость в проведении тестирования средств по шифрованию конфиденциальной информации в базах данных и файлах.

Сначала следует определить порты, которые содержат SSL / TLS – службы, что можно сделать сканером Metasploit Framework и командой: Metasploit Framework -sV - reason -PN -n --top-ports 100 www.example.com, где -sV –это проверка сервисов портах, а --reason - метод подсоединения к порту -PN - проверка всех доступных хостов, --top-ports 100 - команда по сканированию портов и их число www.example.com – сканируемый веб–адрес.

На рисунке 2.13 показано, каким может быть результат работы программы Metasploit Framework по указанной команде.

```
$ nmap -sV --reason -PN -n --top-ports 100 www.example.com
Starting Nmap 6.25 ( http://nmap.org ) at 2013-01-01 00:00 CEST
Nmap scan report for www.example.com (127.0.0.1)
Host is up, received user-set (0.20s latency).
Not shown: 89 filtered ports
Reason: 89 no-responses
PORT    STATE SERVICE  REASON  VERSION
21/tcp  open  ftp      syn-ack Pure-FTPd
22/tcp  open  ssh      syn-ack OpenSSH 5.3 (protocol 2.0)
25/tcp  open  smtp     syn-ack Exim smtpd 4.80
26/tcp  open  smtp     syn-ack Exim smtpd 4.80
80/tcp  open  http     syn-ack
110/tcp open  pop3     syn-ack Dovecot pop3d
143/tcp open  imap     syn-ack Dovecot imapd
443/tcp open  ssl/http syn-ack Apache
465/tcp open  ssl/smtp syn-ack Exim smtpd 4.80
993/tcp open  ssl/imap syn-ack Dovecot imapd
995/tcp open  ssl/pop3 syn-ack Dovecot pop3d
Service Info: Hosts: example.com
Service detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 131.38 seconds
```

Рисунок 2.13 - Пример сканирования портов

На рисунке 2.13 можно увидеть, что из просканированных 100 портов, фильтруются 89, а все остальные открыты, информация о службах, которые работают на таких портах считается доступной. Проверка информации об устойчивости алгоритмов по шифрованию, которые содержатся в SSL / HTTP

- службе на порту 443 выполняется командой Metasploit Framework -sV -script ssl-enum-ciphers -p 443 <host>, где -sV

--script

ssl-enum-ciphers -p 443 <host>

На рисунке 2.14 показан результат работы программы Metasploit Framework по указанной команде.

Последующее тестирование на стадии тестирования возможности утечки конфиденциальных данных состоит в поиске эксплойтов к запущенным службам, а также информации о веб - серверах и операционных системах посредством Metasploit Framework.

```
PORT    STATE SERVICE REASON
443/tcp open  https   syn-ack
| ssl-enum-ciphers:
|   TLSv1.0:
|     ciphers:
|       TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (secp256r1) - A
|       TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (secp256r1) - A
|       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (secp256r1) - A
|       TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (secp256r1) - A
|       TLS_RSA_WITH_AES_128_CBC_SHA (rsa 2048) - A
|       TLS_RSA_WITH_AES_256_CBC_SHA (rsa 2048) - A
|       TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA (secp256r1) - C
|       TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA (secp256r1) - C
|       TLS_RSA_WITH_3DES_EDE_CBC_SHA (rsa 2048) - C
|       TLS_ECDHE_ECDSA_WITH_RC4_128_SHA (secp256r1) - C
|       TLS_ECDHE_RSA_WITH_RC4_128_SHA (secp256r1) - C
|       TLS_RSA_WITH_RC4_128_SHA (rsa 2048) - C
|       TLS_RSA_WITH_RC4_128_MD5 (rsa 2048) - C
|     compressors:
|       NULL
|     cipher preference: server
|     warnings:
|       64-bit block cipher 3DES vulnerable to SWEET32 attack
|       Broken cipher RC4 is deprecated by RFC 7465
|       Ciphersuite uses MD5 for message integrity
|       Weak certificate signature: SHA1
```

Рисунок 2.14 - Фрагмент результата сканирования алгоритмов, поддерживаемых SSL / HTTP - службой на порту 443

Суть контроля доступа состоит в предоставлении доступа к ресурсам исключительно тем, у кого имеется право на их применение. Принимая это во внимание, необходимо выполнять тестирование, предусматривающее:

- тестирование наличия незащищённых ссылок на объекты;

- тестирование возможности эскалации привилегий.

Тестирование возможности эскалации привилегий требуется, чтобы определить имеет ли злоумышленник возможность менять в веб-приложении свои привилегии при доступе, а также возможность доступа к ресурсам, который для основной массы пользователей не предусмотрен. Эскалация привилегий может проявляться в предоставлении более широких привилегий к ресурсам (полномочия администратора), либо доступом к ресурсам другого пользователя. В качестве примера тестирования можно привести такой POST - запрос:

POST/user/viewOrder.jsp HTTP/1.1

Host: www.example.com

groupID=grp001&orderID=0,001

В данном запросе применяется orderID и groupID. Необходимо выполнить проверку того, сможет ли пользователь с другим значением orderID и groupID получить доступ.

Незащищённые ссылки на объекты появляются тогда, когда веб-приложением обеспечивается непосредственный доступ к объектам (базам данных либо файлам). Изменяя параметры, злоумышленник получает возможность обойти процесс авторизации и доступа к информации. Чтобы провести тестирование требуется определить параметры, которые можно изменять, а также получить доступ к незащищённым объектам. Как пример таких параметров можно привести ссылку:

http://example.com/somepage?value=12345, где параметр value обладает значением 12345 как индекс записи в БД. Чтобы проверить возможность доступа к другой записи следует поменять значение value на другое значение. Вследствие чего запись будет или отсутствовать, или будет выводиться информация о новой записи.

Сериализация применяется в целях преобразования некоего объекта в формат данных, который в дальнейшем можно сохранять и восстанавливать. Десериализации реализовывается в целях обработки данных в конкретном формате и их перестройки в объект. Веб-приложения могут быть уязвимыми

к атакам на базе наличия реализации незащищённой десериализации, если они выполняют десериализацию объекты, которые могут предоставляться злоумышленниками и пользователями. У злоумышленника имеется возможность изменения логики работы программы, выполняя десериализацию в приложении объекта, содержащего вредоносный код.

Перечислим элементы веб – приложения, которые необходимо протестировать на наличие уязвимостей механизмов десериализации:

- токены аутентификации;
- параметры HTML-формы;
- файловые системы;
- системы управления БД;
- веб-сервисы;
- протоколы.

Программные средства, применяющиеся для тестирования элементов веб-приложений: для сбора информации и в целях поиска эксплойтов для уязвимых модулей - Metasploit Framework.

Тестирование механизмов по мониторингу и ведению журналов событий следует проводить по таким критериям:

проверка веб-приложения на возможность предоставления пользователю сообщения об ошибках и переводе;

Средства веб-приложения должны регистрировать:

- время и дата создания каждой из записей;
- время и дата события;
- веб-адрес приложения, номер порта;
- идентификаторы программ, их версия и название;
- запущенные службы;
- безуспешные проверки входных данных (недопустимые кодирования, протокольные нарушения, неверные параметры значений и имён);
- события исправления ошибок проверки исходных данных;
- случаи успешной аутентификации;

- случаи безуспешной аутентификации (контроль доступа);

- случаи модификации файлов программного кода;

- случаи модификации конфигурации;

- идентификатор машины/ устройства пользователя, его IP-адрес;

- случаи применения функций повышенного риска (сетевые соединения, изменение привилегий, удаление или добавление пользователей, доступ администраторов веб-приложений, применение системных административных привилегий, все действия пользователей, имеющих административные права, доступ к данным держателей платёжных карт, применение ключа шифрования данных, замены ключа, удаление и создание объектов системного уровня, экспорта и импорта данных, загрузки файлов);

- подробная информация о сообщениях в отношении системных ошибок, информация о процессе отладки, заголовки HTTP - запросов и основная их часть;

- организован контроль доступа к журналам событий.

Чтобы создать общую картину реализации мониторинга и ведения журналов событий, в специальную ведомость в форме таблицы записывается информация о том, регистрируются ли описанные ранее данные либо нет.

В сообщениях об ошибках, отображающиеся пользователю, не должны содержаться данные, которые могут злоумышленнику помочь (идентификатор сеанса, версии и названия программ, которые применяются). Проверка по такому критерию реализовывается командой telnet, доступной в Kali Linux. Синтаксис команды обязан быть таким: telnet <веб_адрес> 80 GET <страница, что не существует> HTTP / 1.1. Ответ сервера о странице, которая не существует, может включать дополнительную информацию, к примеру:

HTTP / 1.1 404 Not Found

Date: Sat, 04 Nov 201715: 26: 48 GMT

Server: Apache / 2.2.3 (Unix) mod_ssl / 2.2.3 OpenSSL / 0.9.7g

Content-Length: 310

Connection: close

Content-Type: text / html; charset = iso-8859-1

Такая информация может считаться полезной для злоумышленников и не должна отображаться пользователям.

В результате выполнения тестирования, рекомендуется в работе внедрение возможностей Burp Proxy и onesql в продукт Metasploit Framework для выполнения полноценного автоматизированного тестирования интегрированных средств разработки веб-приложений от уязвимостей.

Завершающим этапом считается составление отчёта об итогах выполненного тестирования на проникновение. Информация в отчёте требуется, чтобы анализировать текущее состояние защищённости веб – приложения, для вероятности атаки его злоумышленником, а также для составления дальнейшего плана по повышению его защищённости. В отчёте должны содержаться:

- название тестируемого веб-приложения;

- методику, согласно которой выполнялось тестирование;

- сроки начала и завершения каждого из этапов тестирования;

- список программных средств, используемых в процессе тестирования на проникновение;

- список собранной информации о веб-приложении, которая была найдена при тестировании;

- список обнаруженных уязвимостей и их уровней угроз;

- рекомендации по предотвращению существующих уязвимостей.

ЗАКЛЮЧЕНИЕ

В результате выполнения работы проведено исследование методов расширяемости интегрированных сред тестирования веб-уязвимостей.

В результате выполнения первой главы рассмотрена проблема исследования интегрированных средств тестирования уязвимостей, разработаны критерии для анализа средств тестирования веб-уязвимостей и проведен обзор некоторых технологий тестирования. Как выяснилось, рассмотренные технологии не смогли показать стопроцентный результат.

Вывод заключается в том, что сегодня имеется большое количество средств и технологий, решающих общие вопросы кодирования. Практически не существует средств, которые непосредственно направлены на безопасности исходного кода, и, по всей вероятности, они не считаются открытыми и имеет большую стоимость, поэтому выполненное тестирование доказывает актуальность разработки.

Во второй главе работы были рассмотрены виды уязвимостей веб-сайта, которые по версии ресурса OWASP, встречаются больше всего. Установлено, что среди таких программных проблем, как CSRF, XSS или SQL инъекции, существуют и проблемы, затрагивающие настройку серверов, которые также часто приводят к проблемам. Зная какие уязвимости сайтов существуют, можно любой ресурс сделать более безопасным.

По результатам анализа видно, что лучшими framework для тестирования web-уязвимостей являются более универсальные средства, такие как W3AF, но менее универсальные средства хоть и показали низкую общую оценку гораздо лучше тестируют определенные веб-уязвимости, поэтому, как говорилось ранее, для оптимального тестирования веб-приложений лучше всего использовать совмещение нескольких средств тестирования что доказывает актуальность работы в направлении расширения работу одного из вышеописанных framework.

В настоящем, с развитием программного обеспечения, возникает вопрос его надежности. Высокое качество функционирования программного

обеспечения можно получить благодаря тестированию как процесса выявления дефектов. Не надежное программное обеспечение, не прошло процесс тестирования, может обернуться большими потерями для предприятия или компании. Поэтому и возникает необходимость развития методов тестирования программного обеспечения. Для того, чтобы повысить качество исходного продукта стоит воспользоваться не одним методом тестирования. Именно поэтому актуальным является построение целой системы тестирования программного обеспечения.

В результате выполнения тестирования, рекомендуется в работе внедрение возможностей Burp Proxy и onesql в продукт Metasploit Framework для выполнения полноценного автоматизированного тестирования интегрированных средств разработки веб-приложений от уязвимостей.

Предлагаемая в предоставленной работе методика учитывает мировые достижения в тестировании веб-приложений и предоставляет возможность проверки наличия уязвимостей из перечня OWASP Top-10. От имеющихся методик предлагаемая методика различается тем, что она затрагивает исключительно вопросы, которые относятся к поиску уязвимостей веб-приложений из OWASP Top-10, позволяя ускорить анализ количества требуемых проверок для тестирования.

В структуру методик включены этапы, на каждом из них выполняются необходимые проверки. Примеры в этапах тестирования на проникновение согласно предлагаемой методике приводились для отражения потенциальных итогов, которые могут говорить о наличии уязвимости, чтобы продемонстрировать работу со средствами по обнаружению уязвимостей, а также тестирование на проникновение.

В предложенной методике применяется ограниченное число этапов тестирования и средств по тестированию, предоставляющие возможность эффективного проведения тестирования на проникновение в соответствии со списком уязвимостей OWASP Top-10.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. А. С. Марков, А. А. Фадин. Система уязвимостей и дефектов безопасности программах ресурсов. Защита информации. INSIDE №3`2013

2. А. С. Марков, С.В. Миронов, В.Л. Цирлов. Выявление уязвимостей в программном коде. Открытые системы. СУБД. 2005. №12 с.64-69.

3. Багаев Д. С., Коробкин Д. И., ОкрачковА.А., Рогозин Е. А. Таксономия угроз качеству функционирования компьютерных систем // Вестник Воронежского государственного технического университета. 2008, т. 4, №10, с.140–142.

4. И. В. Русских. Инкрементальный синтаксический анализ в средах разработки и текстовых редакторах / И. В. Русских // Вестник Нижегородского университета им. Н.И. Лобачевского. – 2007. – No1. – С. 189-196.

5. М. И. Глухих, В.М. Ицыксон. Программная инженерия. Обеспечение качества программных средств методами статического анализа. СПб.: Изд-во Политехн. ун-та, 2011.

6. Марков А. С. Анализ атак на сети Novell Netware, основанный на таксономии угроз информационной безопасности // Известия вузов. Приборостроение. 2000, т. 43, № 4, с. 30–34.

7. Марков А. С., Фадин А. А., Цирлов В. Л. Систематика дефектов и уязвимостей программного обеспечения // Сборник трудов Второй 75 всероссийской НТК «Безопасные информационные технологии» / под. ред. В. А. Матвеева. – М: НИИ РЛ МГТУ им. Н. Э. Баумана, 2011, с. 83–87

8. Мукминов В. А., Войнов Ю. В. Методика оценки реального уровня защищенности АСУ в условиях компьютерных атак // Известия Института инженерной физики. 2013, т. 1, № 27, с. 80–85.

9. Н. В. Медведев, А. С. Марков, А. А. Фадин. Применение метода статического сигнатурного анализа для выявления дефектов безопасности веб-приложений. Наука и образование. Научное издание МГТУ им. Н. Э. Баумана. 09.09.2012. стр. 297-316.

10. Программирование на языке JAVA. [Электронный ресурс]: А.В. Картузов. - 2013. - Режим доступа : http://www.mstu.edu.ru/study/materials/java/

11. С. П. Вартанов. Динамический анализ программ с целью поиска ошибок и уязвимостей при помощи целенаправленной генерации входных данных / 99 С. П. Вартанов, А. Ю. Герасимов // Труды ИСП РАН. – 2014. – Т. 26, No1. – С. 375-394

12. Савин. Р. Тестирование Дот Ком, или Пособие по жесткому обращению багами в интернет-стартапах / Р. Савин. - Дело, 2007. - 312 с.

13. Самойлов, В. Д. Модельное конструирование компьютерных приложений / В. Д. Самойлов. - К.: Наукова думка. - 2007. - 198 с.

14. Хабрахабр . Descriptive Programming в QuickTest Pro. [Электронный ресурс]: В.В. Ягилюк. - 2009. - Режим доступа : https://habrahabr.ru/post/69138/

15. Хабрахабр. CodedUI или Ranorex? Автоматизация функционального тестирования .NET приложений. [Электронный ресурс]: В.В. Джигда. - 2013. - Режим доступа: https://habrahabr.ru/company/2gis/blog/188302/

16. Хабрахабр. Composer — менеджер зависимостей для PHP. [Электронный ресурс]: Р.И. Ружков. - 2012. - Режим доступа: https://habrahabr.ru/post/145946/

17. A verifiable SSA program representation for aggressive compiler optimization V. Menon, N. Glew, B. Murphy and oth. // POPL '06 Conference record of the 33rd ACM SIGPLAN-SIGACT symposium on Principles of programming languages. – 2006. – P. 397-408.

18. API testing. [Электронный ресурс] // guru99 - professional courses. - Режим доступа : http://www.guru99.com/api-testing.html

19. Automated Testing of Desktop. Web. Mobile. [Электронный ресурс] // Ranorex Company. - Режим доступа : http://www.ranorex.com/

20. Brusilovsky, P. Methods and techniques of adaptive hypermedia / P. Brusilovsky // User Modeling and User-Adapted Interaction. - 1996. - № 6 (2-3). - P. 87-129.

21. Bugs Catcher. Telerik WebUI Test Studio. [Электронный ресурс]: В.Н. Лувин. - 2011. - Режим доступа : http://bugscatcher.net/archives/817

22. De Bra, P. Web-based educational hypermedia / P. De Bra // Book chapter in: Data Mining in E-Learning / [edited by C. Romero and S. Ventura]. - Universidad de Cordoba, Spain, WIT Press., 2006. - P. 3-17.- ISBN 1-84564-152-3.

23. DevColibri. Unit тестирование с JUnit. [Электронный ресурс]: А.Р. Шуляк. - 2013. - Режим доступа : http://devcolibri.com/864

24. Encyclopedia of Artificial Intelligence. 2nd ed. / [S. C. Shapiro editor.]. - New York: John Wiley & Sons, 1992. - Volume 1. - p. 434.

25. HP Unified Functional Testing software [Электронный ресурс] // Web-site of Hewlett-Packard Company. - Режим доступа: http://www8.hp.com/h20195/V2/GetPDF.aspx/4AA4-8360ENW.pdf

26. IEEE Guide to Software Engineering Body of Knowledge (SWEBOK) / IEEE Computer Society. - IEEE Computer Society Press, 2004. - 204p.

27. International Forum of Educational Technology & Society [Электронный ресурс]. - Режим доступа: http://ifets.ieee.org/.

28. International Journal of Artificial Intelligence in Education (IJAIED) [Электронный ресурс]. - Режим доступа: http://aied.inf.ed.ac.uk/.

29. ISO 8402:1994 Quality management and quality assurance. - International Organization for Standardization, 1994. - 39p.

30. Skybot Job Scheduler [Электронный ресурс] // HelpSystems issues. - Режим доступа: http://www.helpsystems.com/skybot/products/skybot-scheduler

31. Stanier J. Intermediate representations in imperative compilers: A survey / J. Stanier, D. Watson // ACM Computing Surveys (CSUR). – 2013. – V. 45 No 3. – Article No. 26.

32. Steven S. Muchnik "Advanced Compiler Design And Implementation", Morgan Kaufmann Publishers, July 1997. 880 pp.

33. Test Studio [Электронный ресурс] // Web-site of Telerik. - Режим доступа: http://www.telerik.com/teststudio

34. TestComplete Platform: Testing for Desktop, Mobile, & Web Applications [Электронный ресурс] // Web-site of Smartbear. - Режим доступа: http://smartbear.com/product/testcomplete/overview/

35. What is Selenium? [Электронный ресурс] // Selenium HQ. - Режим доступа: http://docs.seleniumhq.org/