МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ
имени Гумарбека Даукеева

Кафедра «Информационные системы и кибербезопасность»
Специальность: 7М06301 - «Системы информационной безопасности »

ДОПУЩЕН К ЗАЩИТЕ
Зав. кафедрой PhD,
Мукашева Асель Коптлеувна
(ученая степень, звание, ФИО)

_____
(подпись)
«_____»_____2021 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ
пояснительная записка

на тему: «Разработка и исследование методов и средств обнаружения уязвимостей Web- приложений»

Магистрант: Нурекен Е.Н. _____ группа МСИБн 19-1
(Ф.И.О.) (подпись)

Руководитель: к.т.н., проф. АУЭС _____ Тынымбаев С. Т.
(ученая степень, звание) (подпись)

Рецензент: _____ _____ Шаяхметова А.С.
(ученая степень, звание) (подпись)

Нормоконтроль: кт.н., проф. АУЭС _____ К. Кәден
(ученая степень, звание) (подпись)

Алматы 2021

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ
имени Гумарбека Даукеева

Институт Космической Инженерии и Телекоммуникаций

Специальность: 7М06301 - «Системы информационной безопасности »

Кафедра:«Информационные системы и кибербезопасность»

ЗАДАНИЕ
на выполнение магистерской диссертации

Магистранту Нурекен Елдосу Наурызбекулы
(фамилия, имя, отчество)

Тема диссертации: «Разработка и исследование методов и средств обнаружения уязвимостей Web- приложений»

Утверждена Ученым советом университета №133 от «03» октября 2019 года.

Цель исследования состоит в обеспечении конфиденциальности и безопасности информации путем разработки методологии поиска уязвимости и дальнейшей автоматизации данного процеса.

Перечень подлежащих разработке в магистерской диссертации вопросов или краткое содержание магистерской диссертации:

1. Анализ состояния вопроса и постановка задачи исследования
2. Моделирование методологии обнаружения веб-уязвимостей
3. Разработка сканера веб-уязвимостей

Перечень графического материала (с точным указанием обязательных чертежей)

Основная рекомендуемая литература

1 Elizabeth, F. Building a Test Suite for Web Application Scanners. Elizabeth F., Romain, G., Vadim, O., Paul, B.

2 Emre, E. Web Vulnerability Scanners: A Case Study. Emre, E., Angel, R. 2017.

Г Р А Ф И К
подготовки магистерской диссертации

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечание
1. Информационный обзор согласно темы	01.02.2019	
2. Основные направления развития и проблемы алгоритмов сканера веб уязвимостей	01.06.2019	
3. Анализ готовых решений на рынке сканеров веб-уязвимостей (исследовательская глава)	01.11.2019	
4. Моделирование методологии поиска уязвимостей	01.03.2020	
5. Разработка сканера веб-уязвимостей.	01.05.2020	

Дата выдачи задания 30 сентября 2019 г.

Заведующий кафедрой _____ (Мукашева А.К.)
(подпись) (Ф.И.О.)

Научный
руководитель диссертации _____ (Тынымбаев С. Т.)
(подпись) (Ф.И.О.)

Задание принял к исполнению
Магистрант _____ (Нурекен Е. Н.)
(подпись) (Ф.И.О.)

Аңдатпа

Магистрлік диссертация веб-қосымшалардың эволюциясы мен қауіпсіздігінің негізгі мәселелерін, веб-осалдықтарды анықтау әдістемесін модельдеуді, сондай-ақ веб-осалдықтардың сканерін әзірлеуді қарастырды. Қауіпсіздік-бұл веб-қосымшалардың маңызды бөлігі. Анықтама бойынша Веб-қосымшалар пайдаланушыларға орталық ресурсқа — веб-серверге және сол арқылы дерекқор серверлері сияқты басқаларға қол жеткізуге мүмкіндік береді. Тиісті қауіпсіздік шараларын түсіну және қолдану арқылы сіз өз ресурстарыңызды қорғайсыз, сонымен қатар пайдаланушылар сіздің қосымшаңызбен жұмыс істеуге ыңғайлы қауіпсіз ортаны қамтамасыз етесіз. Диссертация кіріспеден және үш тараудан тұрады, 15 атаудан тұратын пайдаланылған әдебиеттер тізімінен және 17 суреттен тұрады.

Аннотация

В магистерской диссертации рассмотрены основные проблемы эволюции и безопасности веб-приложений, моделирование методологии обнаружения веб-уязвимостей, а так же разработка сканера веб-уязвимостей. Безопасность — важная часть ваших веб-приложений. Веб-приложения по определению позволяют пользователям получать доступ к центральному ресурсу — веб-серверу — и через него — к другим, таким как серверы баз данных. Понимая и применяя надлежащие меры безопасности, вы охраняете свои собственные ресурсы, а также предоставляете безопасную среду, в которой ваши пользователи удобны в работе с вашим приложением. Диссертация состоит из введения и трех глав, содержит список использованной литературы из 15 наименований, включает 17 рисунков.

Abstract

In the master's thesis, the main problems of the evolution and security of web applications, modeling of the methodology for detecting web vulnerabilities, as well as the development of a web vulnerability scanner are considered. Security is an important part of your web applications. Web applications, by definition, allow users to access a central resource — a web server-and, through it, others, such as database servers. By understanding and applying appropriate security measures, you protect your own resources, and also provides a secure environment in which your users are comfortable with your application. The dissertation consists of an introduction and three chapters, contains a list of references from 15 titles, includes 17 figures.

Содержание

ВВЕДЕНИЕ

По многим оценкам алгоритмы распределенного хранения и обработки данных одно из самых перспективных направлений в области технологий (вместе с машинным обучением (ML), интернетом вещей (IoT) иискусственным интеллектом (AI)), которое может иметь влияние на нашу жизнь, сопоставимое с развитием Интернета и мобильных устройств в начале 2000-х годов. По оценкам Всемирного экономического форума, к концу 2027-го года уже 10% мирового ВВП будет храниться в блокчейне.

На данный момент размер рынка распределённого хранения и обработки данных составляет около 230 млрд. долларов, большую часть, которого занимают финансовые операции.

Снижение издержек, повышение уровня безопасности и более высокая прозрачность транзакций – три главные сильные стороны блокчейна. В связи с потребностью банков, бизнеса и общества в этих трех аспектах, любая теоретическая работа или разработка в этой области становится достаточно актуальной.

Так же актуальность обоснована низкой изученностью данного научного направления в Республике Казахстан. Открыты возможности для формирования центра компетенций в области децентрализованных технологий с целью дальнейшего их применения на корпоративном рынке и государственных структурах в нашей стране.

Новизна работы состоит в поиске новых сценариев использования технологии в других отраслях с целью совершенствования систем обмена информацией, обеспечивая следующие свойства:прозрачность, необратимость, анонимность/псевдонимность, децентрализованность.

Практическая значимость работы заключается в возможности использования результатов исследования для задач, связанных с адаптацией блокчейна к IoT.

В работе исследуетсятехнология блокчейна, анализируются проблемы на теоретического плана, возникающиев областях,в которых технология может быть применена. С использованием инструментов моделирования планируется исследование влияния интеграции блокчейна на характеристики IoT сети.

1 АНАЛИЗ СОСТОЯНИЯ ВОПРОСА И ПОСТАНОВКА ЗАДАЧИ ИССЛЕДОВАНИЯ

1.1 Эволюция веб приложения

В первые дни существования Интернета Всемирная паутина состояла только из веб-сайтов. По сути, это были информационные хранилища, содержащие статические документы. Веб-браузеры были изобретены как средство извлечения и отображения этих документов. Поток информации был односторонним, от сервера к браузеру. Большинство сайтов не проверяли подлинность пользователей, потому что в этом не было необходимости. С каждым пользователем обращались одинаково и им была представлена одинаковая информация. Любые угрозы безопасности, возникающие при размещении веб-сайта, были связаны в основном с уязвимостями в программном обеспечении веб-сервера (которых было много). Если злоумышленник скомпрометировал веб-сервер, он обычно не получал доступа к какой-либо конфиденциальной информации, поскольку информация, хранящаяся на сервере, уже была открыта для публичного просмотра. Скорее всего, злоумышленник обычно изменял файлы на сервере, чтобы испортить содержимое веб-сайта или использовать хранилище и пропускную способность сервера.

Сегодня Всемирная паутина почти неузнаваема по сравнению с ее более ранней формой. Большинство сайтов в Интернете на самом деле являются приложениями. Они очень функциональны и полагаются на двусторонний поток информации между сервером и браузером. Они поддерживают регистрацию и вход в систему, финансовые операции, поиск и создание контента пользователями. Контент, представленный пользователям, генерируется динамически на лету и часто адаптируется к каждому конкретному пользователю. Большая часть обрабатываемой информации является частной и очень конфиденциальной. Безопасность, поэтому это большая проблема. Никто не хочет использовать веб-приложение, если он считает, что его информация будет раскрыта неавторизованным лицам.

Развитие веб-приложения несет с собой новые и значительные угрозы безопасности. Каждое приложение отличается и может содержать уникальные уязвимости. Большинство приложений разрабатываются собственными силами-многие из них разработчиками, которые имеют лишь частичное представление о проблемах безопасности, которые могут возникнуть в создаваемом ими коде. Для обеспечения своей основной функциональности веб-приложениям обычно требуется подключение к внутренним компьютерным системам, содержащим высокочувствительные данные и способным выполнять мощные бизнес-функции. Пятнадцать лет назад, если хочешь чтобы осуществить перевод средств, вы посетили свой банк, и кассир выполнил перевод за вас; сегодня можно посетить

веб-приложение и выполнить перевод самостоятельно[1]. Злоумышленник, который компрометирует веб-приложение, может украсть личную информацию, осуществить финансовое мошенничество и выполнить вредоносные действия против других пользователей

Веб-приложения были созданы для выполнения практически всех полезных функций, которые вы могли бы реализовать в Интернете. Вот некоторые функции веб-приложений , которые стали заметными в последние годы:

Покупки (Amazon)

Социальные сети (Facebook)

Поиск в Интернете (Google)

Аукционы (eBay)

Приложения, доступ к которым осуществляется с помощью браузера компьютера, все чаще пересекаются с мобильными приложениями, доступ к которым осуществляется с помощью смартфона или планшета. Наиболее мобильные приложения используют либо браузер, либо настроенный клиент, который использует API на основе HTTP для связи с сервером. Функции и данные приложения обычно совместно используются между различными интерфейсами, которые приложение предоставляет различным пользовательским платформам. В дополнение к общедоступному Интернету, веб-приложения широко используются внутри организаций для поддержки ключевых бизнес-функций. Многие из них обеспечивают доступ к высокочувствительным данным и функциям:

1) HR-приложения, позволяющие пользователям получать доступ к информации о заработной плате, предоставлять и получать отзывы о результатах работы и управлять процедурами найма и дисциплинарными процедурами.
2) Административные интерфейсы для ключевых инфраструктур, таких как веб-и почтовые серверы, рабочие станции пользователей и администрирование виртуальных машин.
3) Программное обеспечение для совместной работы, используемое для обмена документами, управления потоком работ и проектами, а также отслеживания проблем. Эти типы функций часто связаны с критическими проблемами безопасности и управления, и организации часто полностью полагаются на элементы управления, встроенные в их веб-приложения.
4) Бизнес-приложения, такие как программное обеспечение для планирования корпоративных ресурсов (ERP) , доступ к которым ранее осуществлялся с помощью проприетарного клиентского приложения, теперь можно получить с помощью веб-браузера.

Во всех этих примерах то, что воспринимается как “внутренние” приложения, все чаще размещается снаружи, поскольку организации переходят к

внешним поставщикам услуг, чтобы сократить расходы. В этих так называемых облачных решениях критически важные для бизнеса функции и данные открыты для более широкого круга потенциальных злоумышленников, и организации все больше полагаются на целостность средств защиты, которые находятся вне их контроля[2].

Быстро приближается время, когда единственным клиентским программным обеспечением, которое понадобится большинству пользователей компьютеров, будет веб-браузер. Широкий спектр функций будет реализован с использованием общего набора протоколов и технологий, и при этом будет унаследован особый ряд общих уязвимостей безопасности.

1.2 Безопасность веб-приложений

Как и в случае с любым новым классом технологий, веб-приложения принесли с собой новый спектр уязвимостей в системе безопасности. Набор наиболее часто встречающихся дефектов несколько эволюционировал с течением времени. Были придуманы новые атаки , которые не учитывались при разработке существующих приложений. Некоторые проблемы стали менее распространенными по мере повышения осведомленности о них. Были разработаны новые технологии, которые открыли новые возможности для эксплуатации. Некоторые категории недостатков в значительной степени исчезли в результате изменений, внесенных в программное обеспечение веб-браузера.

Наиболее серьезными атаками на веб-приложения являются те, которые раскрывают конфиденциальные данные или получают неограниченный доступ к внутренним системам, на которых работает приложение. Громкие компромиссы такого рода по-прежнему происходят часто. Однако для многих организаций любая атака, приводящая к простою системы, является критическим событием. Атаки типа "отказ в обслуживании" на уровне приложений могут использоваться для достижения тех же результатов, что и традиционные атаки на исчерпание ресурсов против инфраструктуры. Однако они часто используются с более тонкими методы и цели. Они могут быть использованы для нарушения работы конкретного пользователя или сервиса, чтобы получить конкурентное преимущество перед коллегами в области финансовой торговли, игр, онлайн-торгов и бронирования билетов.

1.3 Механизм работы средств обнаружения веб-уязвимостей

Существует два основных механизма, при помощи которых сканер проверяет наличие уязвимости:

Сканирование- механизм пассивного анализа, с помощью которого сканер пытается определить наличие уязвимости без фактического подтверждения ее

наличия, т.е. по косвенным признакам. Этот метод является наиболее быстрым и простым для реализации. Этот процесс идентифицирует открытые порты, найденные на каждом сетевом устройстве, и собирает связанные с портами заголовки (banner), найденные при сканировании каждого порта. Каждый полученный заголовок сравнивается с таблицей правил определения сетевых устройств, операционных систем и потенциальных уязвимостей. На основе проведенного сравнения делается вывод о наличии или отсутствии уязвимости.
Зондирование - механизм активного анализа, который позволяет убедиться, присутствует или нет на анализируемом узле уязвимость. Зондирование выполняется путем имитации атаки, использующей проверяемую уязвимость. Этот метод более медленный, чем сканирование, но почти всегда гораздо более точный, чем он. Этот процесс использует информацию, полученную в процессе сканирования, для детального анализа каждого сетевого устройства. Этот процесс также использует известные методы реализации атак для того, чтобы полностью подтвердить предполагаемые уязвимости и обнаружить другие уязвимости, которые не могут быть обнаружены пассивными методами, например подверженность атакам типа "отказ в обслуживании" (DoS - denialofservice).

На практике указанные механизмы реализуются следующими методами:

1. Проверка заголовков (bannercheck) - указанный механизм представляет собой ряд проверок типа "сканирование" и позволяет делать вывод об уязвимости, опираясь на информацию в заголовке ответа на запрос сканера. Типичный пример такой проверки - анализ заголовков программы Sendmail или FTP-сервера, позволяющий узнать их версию и на основе этой информации сделать вывод о наличии в них уязвимости.

Это наиболее быстрый и простой для реализации метод проверки присутствия на сканируемом узле уязвимости. Однако эффективность проверок заголовков достаточно эфемерна. И вот почему. Во-первых, можно изменить текст заголовка, предусмотрительно удалив из него номер версии или иную информацию, на основании которой сканер строит свои заключения. Во-вторых, зачастую, версия, указываемая в заголовке ответа на запрос, не всегда говорит об уязвимости программного обеспечения. Особенно это касается программного обеспечения, распространяемого вместе с исходными текстами. Можно самостоятельно устранить уязвимость путем модификации исходного текста, при этом, забыв изменить номер версии в заголовке. В-третьих, устранение уязвимости в одной версии еще не означает, что в следующих версиях эта уязвимость отсутствует.Процесс, описанный выше, является первым и очень важным шагом при сканировании сети. Он не приводит к нарушению функционирования сервисов или узлов сети.

2. Активные зондирующие проверки (activeprobingcheck) - также относятся к механизму сканирования. Однако они основаны на сравнении "цифрового слепка" фрагмента программного обеспечения со слепком известной уязвимости.

Аналогичным образом поступают антивирусные системы, сравнивая фрагменты сканируемого программного обеспечения с сигнатурами вирусов, хранящимися в специализированной базе данных. Разновидностью этого метода являются проверки контрольных сумм или даты сканируемого программного обеспечения, которые реализуются в сканерах, работающих на уровне операционной системы.Специализированная база данных по сетевой безопасности содержит информацию об уязвимостях и способах их использования (атаках). Эти данные дополняются сведениями о мерах их устранения, позволяющих снизить риск безопасности в случае их обнаружения. Зачастую эта база данных используется и системой анализа защищенности и системой обнаружения атак. Этот метод также достаточно быстр, но реализуется труднее, чем "проверка заголовков".

3. Имитация атак (exploitcheck) - данные проверки относятся к механизму зондирования и основаны на эксплуатации различных дефектов в программном обеспечении.Некоторые уязвимости не обнаруживают себя, пока вы не "подтолкнете" их. Для этого против подозрительного сервиса или узла запускаются реальные атаки. Проверки заголовков осуществляют первичный осмотр сети, а метод "exploitcheck", отвергая информацию в заголовках, позволяет имитировать реальные атаки, тем самым с большей эффективностью (но меньшей скоростью) обнаруживая уязвимости на сканируемых узлах. Имитация атак является более надежным способом анализа защищенности, чем проверки заголовков, и обычно более надежны, чем активные зондирующие проверки.Однако существуют случаи, когда имитация атак не всегда может быть реализована. Такие случаи можно разделить на две категории: ситуации, в которых тест приводит к "отказу в обслуживании" анализируемого узла или сети, и ситуации, при которых уязвимость в принципе негодна для реализации атаки на сеть[3].

Многие проблемы защиты не могут быть выявлены без блокирования или нарушения функционирования сервиса или компьютера в процессе сканирования. В некоторых случаях нежелательно использовать имитацию атак (например, для анализа защищенности важных серверов), т.к. это может привести к большим затратам (материальным и временным) на восстановление работоспособности выведенных из строя элементов корпоративной сети. В этих случаях желательно применить другие проверки, например, активное зондирование или, в крайнем случае, проверки заголовков.

Однако, есть некоторые уязвимости (например, проверка подверженности атакам типа "PacketStorm"), которые просто не могут быть протестированы без возможного выведения из строя сервиса или компьютера. В этом случае разработчики поступают следующим образом, - по умолчанию такие проверки выключены и пользователь может сам включить их, если желает. При включении любой из проверок этой группы, системы выдают сообщение типа "WARNING:"

("Внимание: эти проверки могут вывести из строя или перезагрузить сканируемые узлы").

Практически любой сканер проводит анализ защищенности в несколько этапов:

1. Сбор информации о сети. На данном этапе идентифицируются все активные устройства в сети и определяются запущенные на них сервисы и демоны. В случае использования систем анализа защищенности на уровне операционной системы данный этап пропускается, поскольку на каждом анализируемом узле установлены соответствующие агенты системного сканера.
2. Обнаружение потенциальных уязвимостей. Сканер использует описанную выше базу данных для сравнения собранных данных с известными уязвимостями при помощи проверки заголовков или активных зондирующих проверок. В некоторых системах все уязвимости ранжируются по степени риска: высокая (High), средняя (Medium) и низкая (Low).
3. Подтверждение выбранных уязвимостей. Сканер использует специальные методы и моделирует (имитирует) определенные атаки для подтверждения факта наличия уязвимостей на выбранных узлах сети.
4. Генерация отчетов. На основе собранной информации система анализа защищенности создает отчеты, описывающие обнаруженные уязвимости. В некоторых системах отчеты создаются для различных категорий пользователей, начиная от администраторов сети и заканчивая руководством компании. Если первых в первую очередь интересуют технические детали, то для руководства компании необходимо представить красиво оформленные с применением графиков и диаграмм отчеты с минимумом подробностей. Немаловажным аспектом является наличие рекомендаций по устранению обнаруженных проблем, когда для каждой уязвимости выдаются пошаговые инструкции для устранения уязвимостей, специфичные для каждой операционной системы. Во многих случаях отчеты также содержат ссылки на FTP- или Web-сервера, содержащие patch и hotfix, устраняющие обнаруженные уязвимости.
5. Автоматическое устранение уязвимостей. Этот этап достаточно редко реализуется в сетевых сканерах, но широко применяется в системных сканерах. При этом данная возможность может реализовываться по-разному. Например, создается специальный сценарий (fixscript), который администратор может запустить для устранения уязвимости. Одновременно с созданием этого сценария, создается и второй сценарий, отменяющий произведенные изменения. Это необходимо в том случае, если после устранения проблемы, нормальное функционирование узла было нарушено. В других системах возможности "отката" не существует. У администратора,

осуществляющего поиск уязвимостей, есть несколько вариантов использования системы анализа защищенности – это различный набор существующих этапов.

Если сканер не находит уязвимостей на тестируемом узле, то это еще не значит, что их нет. Просто сканер не нашел их. И зависит это не только от самого сканера, но и от его окружения. Например, если Вы тестируете сервис Telnet или FTP на удаленной машине, и сканер сообщает Вам, что уязвимостей не обнаружено - это может значить не только, что уязвимостей нет, а еще и то, что на сканируемом компьютере установлен, например, TCP Wrapper. Да мало ли еще чего? Можно пытаться получить доступ к компьютеру через межсетевой экран или попытки доступа блокируются соответствующими фильтрами у провайдера и т.д. Для ОС Windows NT характерен другой случай. Сканер пытается дистанционно проанализировать системный реестр (registry). Однако в случае запрета на анализируемом узле удаленного доступа к реестру, сканер никаких уязвимостей не обнаружит. Существуют и более сложные случаи. И вообще различные реализации одного итого же сервиса по-разному реагируют на системы анализа защищенности. Очень часто на практике можно увидеть, что сканер показывает уязвимости, которых на анализируемом узле нет. Это относится к сетевым сканерам, которые проводят дистанционный анализ узлов сети. И удаленно определить, существует ли в действительности уязвимость или нет, практически невозможно. В этом случае можно порекомендовать использовать систему анализа защищенности на уровне операционной системы, агенты которой устанавливаются на каждый контролируемый узел и проводят все проверки локально.

Для решения этой проблемы некоторые компании-производители пошли по пути предоставления своим пользователям нескольких систем анализа защищенности, работающих на всех указанных выше уровнях, - сетевом, системном и уровне приложений. Совокупность этих систем позволяет с высокой степенью эффективности обнаружить практически все известные уязвимости. Например, компания InternetSecuritySystems предлагает семейство SAFEsuite, состоящее из четырех сканеров: InternetScanner, SystemScanner, SecurityManager и DatabaseScanner. В настоящий момент это единственная компания, которая предлагает системы анализа защищенности, функционирующие на всех трех уровнях информационной инфраструктуры. Другие компании предлагают или два (Axent) или, как правило, один (NetworkAssociates, NetSonar и др.) сканер.

Компания Cisco, предлагающая только систему анализа защищенности на уровне сети пошла другим путем для устранения проблемы ложного срабатывания. Она делит все уязвимости на два класса:

1. Потенциальные - вытекающие из проверок заголовков и т.н. активных "подталкиваний" (nudge) анализируемого сервиса или узла. Потенциальная

уязвимость возможно существует в системе, но активные зондирующие проверки не подтверждают этого.

2. Подтвержденные - выявленные и существующие на анализируемом хосте.

Проверки на потенциальную уязвимость проводятся через коллекцию заголовков и использование "несильных подталкиваний". "Подталкивание" используется для сервисов, не возвращающих заголовки, но реагирующих на простые команды, например, посылка команды HEAD для получения версии HTTP-сервера. Как только эта информация получена, система NetSonar использует специальный механизм (rulesengine), который реализует ряд правил, определяющих, существует ли потенциальная уязвимость.Таким образом, администратор знает, какие из обнаруженных уязвимостей действительно присутствуют в системе, а какие требуют подтверждения.

Однако в данном случае остаются уязвимости, с трудом обнаруживаемые или совсем не обнаруживаемые через сеть. Например, проверка "слабости" паролей, используемых пользователями и другими учетными записями. В случае использования сетевого сканера вам потребуется затратить очень много времени на удаленную проверку каждой учетной записи. В то же время, аналогичная проверка, осуществляемая на локальном узле, проводится на несколько порядков быстрее. Другим примером может служить проверка файловой системы сканируемого узла. Во многих случаях ее нельзя осуществить дистанционно.

Достоинства сканирования на уровне ОС кроются в прямом доступе к низкоуровневым возможностям ОС хоста, конкретным сервисам и деталям конфигурации. Тогда как сканер сетевого уровня имитирует ситуацию, которую мог бы иметь внешний злоумышленник, сканер системного уровня может рассматривать систему со стороны пользователя, уже имеющего доступ к анализируемой системе и имеющего в ней учетную запись. Это является наиболее важным отличием, поскольку сетевой сканер по определению не может предоставить эффективного анализа возможных рисков деятельности пользователя. Многие сканеры используют более чем один метод проверки одной и той же уязвимости или класса уязвимостей. Однако в случае большого числа проверок использование нескольких методов поиска одной уязвимости приносит свои проблемы. Связано это со скоростью проведения сканирования.В любом случае наиболее предпочтительным является проверка типа "имитация атак", которая обеспечивает наибольший процент точного обнаружения уязвимостей.Не все проверки, разработанные в лабораторных условиях, функционируют так, как должны. Даже, несмотря на то, что эти проверки тестируются, прежде чем будут внесены в окончательную версию сканера. На это могут влиять некоторые факторы:

1. Особенности конфигурации пользовательской системы.
2. Способ, которым был скомпилирован анализируемый демон или сервис.
3. Ошибки удаленной системы и т.д.

В таких случаях автоматическая проверка может пропустить уязвимость, которая легко обнаруживается вручную и которая может быть широко распространена во многих системах. Проверка заголовка в совокупности с активным зондированием в таком случае может помочь определить подозрительную ситуацию, сервис или узел. И хотя уязвимость не обнаружена, еще не значит, что ее не существует. Необходимо другими методами, в том числе и неавтоматизированными, исследовать каждый подозрительный случай.

Кроме того, если в созданном отчете не сказано о той или иной уязвимости, то иногда стоит обратиться к журналам регистрации (log) системы анализа защищенности. В некоторых случаях, когда сканер не может со 100%-ой уверенностью определить наличие уязвимости, он не записывает эту информацию в отчет, однако сохраняет ее в логах. Существуют различия и между тем, как влияет одна и та же проверка на различные версии сервисов в различных операционных системах.

2 МОДЕЛИРОВАНИЕ МЕТОДОЛОГИИ ОБНАРУЖЕНИЯ ВЕБ-УЯЗВИМОСТЕЙ

2.1 Отображение содержимого приложения

Первым этапом при поиске уязвимостей необходимо проводить разведку, для того чтобы собрать необходимое количество информации, нужно пройти следующие шаги (в соответствии с рисунком 2.1).

Рисунок 2.1 – Отображение содержимого приложения

2.1.1 Изучение видимого контента

Необходимо настроить свой браузер на использование вашего любимого встроенного прокси-сервера/инструмента spider. Как Burp, так и WebScarab можно использовать для пассивного спайдера сайта путем мониторинга и анализа веб-контента, обрабатываемого прокси-сервером.

Нужно настроить свой браузер на использование расширения, такого как IEWatch, для мониторинга и анализа содержимого HTTP и HTML, обрабатываемого браузером.

Необходимо просмотреть все приложение обычным способом, посещая каждую ссылку и URL, отправляя каждую форму и проходя через все многоступенчатые функции до завершения. Следует попробовать просматривать страницы с включенным и отключенным JavaScript, а также с включенными и отключенными файлами cookie. Многие приложения могут обрабатывать различные конфигурации браузера, и можно получить доступ к различным контентами путям кода в приложении.

Если приложение использует аутентификацию, и если есть возможность создать учетную запись для входа, нужно использовать ее для доступа к защищенным функциям.

Во время просмотра нужно следить за запросами и ответами, проходящими через перехватывающий прокси-сервер, чтобы получить представление о типах отправляемых данных и способах использования клиента для управления поведением серверного приложения.

Нужно просмотреть карту сайта, созданную пассивным спайдерингом, и определить любой контент или функциональность, которые не были просмотрены с помощью браузера. Из результатов spider нужно определить, где был обнаружен каждый элемент (например, в Burp Spider Нужно проверить ссылку из сведений). Необходимо получить доступ к каждому элементу с помощью браузера, чтобы spider проанализировал ответ с сервера для идентификации любого дополнительного контента. Этот шаг нужно повторять рекурсивно до тех пор, пока не будет идентифицировано дополнительное содержимое или функциональность.

После того как просмотр вручную и пассивный спайдеринг будет окончен, можно использовать своего спайдера для активного обхода приложения, используя набор обнаруженных URL-адресов в качестве исходных данных. Иногда это может привести к обнаружению дополнительного контента, который был пропущен при сборе данных вручную. Прежде чем выполнять автоматический обход, нужно определить все URL-адреса, которые опасны или могут нарушить сеанс приложения, а затем настроить spider, чтобы исключить их из своей области действия.

2.1.2 Рассмотрение общедоступных ресурсов

Нужно использовать поисковые системы Интернета и архивы (например, Wayback Machine) чтобы определить, какой контент они проиндексировали и сохранили для целевого приложения.

Следует использовать расширенные параметры поиска, чтобы повысить эффективность ваших исследований. Например, в Google можно использовать

site: для получения всего контента для необходимого целевого сайта и link: для получения других сайтов, которые ссылаются на него. Если производится поиск контента, которого больше нет в приложении, все равно можно просмотреть его из кэша поисковой системы. Этот старый контент может содержать ссылки на дополнительные ресурсы, которые еще небыли удалены.

Нужно выполнить поиск по любым именам и адресам электронной почты, которые были обнаружены в содержимом приложении, например по контактной информации. Для повышения эффективности следует включить элементы, не отображаемые на экране, такие как HTML-комментарии. В дополнение к поиску в Интернете следует выполнить поиск новостей и групп. Нужно искать любые технические подробности, размещенные на интернет-форумах, касающиеся целевого приложения и его инфраструктуры.

Следует просмотреть все опубликованные файлы WSDL, чтобы создать список имен функций и значений параметров, потенциально используемых приложением.

2.1.3 Обнаружение скрытого контента

Нужно подтвердить, как приложение обрабатывает запросы на несуществующие элементы. Сделать несколько запросов вручную для известных допустимых и недопустимых ресурсов и сравнить ответы сервера, чтобы установить простой способ определить, когда элемент не существует.

Получив список общих имен файлов и каталогов, а также общих расширений файлов, необходимо добавить в эти списки все элементы, фактически наблюдаемые в приложениях, а также элементы, выведенные из них. Также следует обратить внимание на именах функций и параметров, используемых разработчиками приложений. Например, если существуют страницы с именами addDocument.jsp и ViewDocument.jsp,могут также существовать страницы с именами EditDocument.jsp и RemoveDocument.jsp.

Следует просмотреть весь код на стороне клиента, чтобы определить любые подсказки о скрытом содержимом на стороне сервера, включая HTML-комментарии и отключенные элементы формы.

Используя методы автоматизации, нужно выполнять большое количество запросов на основе списков каталогов, имен файлов и расширений файлов. Нужно следить за ответами сервера на подтверждение того, какие элементы присутствуют и доступны.

Следует выполнять эти пункты по обнаружению содержимого рекурсивно, используя новое перечисленное содержимое и шаблоны в качестве основы для дальнейшего ориентированного на пользователя спайдеринга и дальнейшего автоматического обнаружения.

2.1.4 Обнаружение содержимого по умолчанию

Следует запустить Nikto против веб-сервера, чтобы обнаружить любой присутствующий по умолчанию или хорошо известный контент. Следует использовать возможности Nikto, чтобы максимизировать его эффективность. Например, можно использовать параметр –root, чтобы указать каталог для проверки содержимого по умолчанию, или -404, чтобы указать строку, идентифицирующую страницу "Пользовательский файл не найден".

Следует проверить все потенциально интересные результаты вручную, чтобы исключить любые ложные срабатывания в результатах.

Нужно запросить корневой каталог сервера, указав IP-адрес в заголовок хоста и необходимо определить, отвечает ли приложение каким-либо другим контентом. Если это так, Следует запустить сканирование Nikto по IP-адресу, а также по имени сервера.

Следует сделать запрос в корневой каталог сервера, указав диапазон заголовков агента пользователя.

2.1.5 Перечисление функций, заданных идентификатором

Необходимо определить любые экземпляры, в которых доступны определенные функции приложения, передав идентификатор функции в параметре запроса (например, //admin.jsp?action=editUser или /main.php?func=A21).

Следует применить методы обнаружения содержимого, используемые в шаге 1.3, к механизму, используемому для доступа к отдельным функциям. Например, если приложение использует параметр, содержащий имя функции, сначала Необходимо определить его поведение при указании недопустимой функции и Необходимо попытаться найти простой способ определить, когда была запрошена допустимая функция. Следует составить список общих имен функций или Нужно пройти по синтаксическому диапазону используемых идентификаторов. Автоматизируйте упражнение, чтобы перечислить допустимые функции как можно быстрее и проще.

Если это применимо, следует составить карту содержимого приложения на основе функциональных путей, а не URL-адресов, показывая все перечисленные функции, а также логические пути и зависимости между ними.

2.1.6 Проверка параметров отладки

Необходимо выбрать одну или несколько страниц приложения или функций, на которых могут быть реализованы скрытые параметры отладки (например, debug=true). Они, скорее всего, появятся в ключевых функциях, таких как вход в систему, поиск и загрузка или загрузка файлов.

Необходимо использовать списки общих имен параметров отладки (таких как debug, test,hide и source) и общих значений (таких как true, yes, on и 1).Повторите все их перестановки, отправляя каждую пару имя/значение каждой

целевой функции. Для запросов POST укажите параметр как в строке запроса URL, так и в теле запроса. Необходимо использовать методы, чтобы автоматизировать это упражнение. Например, вы можете использовать тип атаки кассетной бомбы в Burp Intruder для объединения всех перестановок двух списков полезной нагрузки.

Следует просмотреть ответы приложения на наличие любых аномалий, которые могут указывать на то, что добавленный параметр повлиял на обработку приложения.

2.2 Анализ веб-приложения

Анализ веб-приложения состоит из следующих важных шагов (в соответствии с рисунком 2.2):

Рисунок 2.2 - Анализ приложения

2.2.1 Определение функциональных возможностей

Необходимо определить основные функциональные возможности, для которых было создано приложение, и действия, для выполнения которых предназначена каждая функция при использовании по назначению.

Необходимо определить основные механизмы безопасности, используемые приложением, и то, как они работают. В частности, поймите ключевые механизмы, которые обрабатывают аутентификацию, управление сеансами и контроль доступа, а также функции, которые их поддерживают, такие как регистрация пользователей и восстановление учетных записей.

Необходимо определить все более периферийные функции и поведения, такие как использование перенаправлений, внешних ссылок, сообщений об ошибках, административных функций и функций ведения журнала.

Необходимо определить любую функциональность, которая отличается от стандартного внешнего вида графического интерфейса, именования параметров

или механизма навигации, используемого в других частях приложения, и нужно выделить ее для углубленного тестирования.

2.2.2 Определение точек ввода данных

Необходимо определить все различные точки входа, которые существуют для ввода пользовательских данных в обработку приложения, включая URL-адреса, параметры строки запроса, данные POST, файлы cookie и другие HTTP-заголовки, обрабатываемые приложением.

Изучите любые настраиваемые механизмы передачи данных или кодирования, используемые приложением, такие как нестандартный формат строки запроса. Необходимо определить инкапсулируют ли передаваемые данные имена и значения параметров или используются альтернативные средства представления.

Необходимо определить любые внеполосные каналы, по которым в обработку приложения вводятся контролируемые пользователем или другие сторонние данные.Примером может служить приложение веб-почты, которое обрабатывает и отображает сообщения, полученные через SMTP.

2.2.3 Необходимо определить используемые технологии

Необходимо определить каждую из различных технологий, используемых на стороне клиента, такие как формы, сценарии, файлы cookie, Java-апплеты, элементы управления ActiveX и флэш-объекты.

Насколько это возможно, необходимо определить, какие технологии используются на стороне сервера, включая языки сценариев, платформы приложений и взаимодействие с внутренними компонентами, такими как базы данных и системы электронной почты.

Нужно проверить заголовок HTTP-сервера, возвращаемый в ответах приложения, а также нужно проверить наличие любых других идентификаторов программного обеспечения, содержащихся в пользовательских HTTP-заголовках или комментарии к исходному коду HTML. Нужно обратить внимание, что в некоторых случаях разные области приложения обрабатываются разными внутренними компонентами, поэтому могут быть получены разные баннеры.

Следует запустить инструмент Httprint, чтобы снять отпечатки пальцев с веб-сервера.

Следует просмотреть результаты ваших упражнений по сопоставлению содержимого, чтобы определить любые интересные расширения файлов, каталоги или другие последовательности URL-адресов, которые могут дать представление о технологиях, используемых на сервере. Следует просмотреть имена любых токенов сеанса и других файлов выпущенных посредством cookie. Нужно воспользоваться Google для поиска технологий, связанных с этими элементами.

2.2.4 Нанесение на карту поверхность атаки

Необходимо попытаться определить вероятную внутреннюю структуру и функциональность серверного приложения, а также механизмы, которые оно использует за кулисами, чтобы обеспечить поведение, видимое с точки зрения клиента. Например, функция для получения заказов клиентов, скорее всего, будет взаимодействовать с базой данных.

Для каждого элемента функциональности необходимо определить типы распространенных уязвимостей, которые часто связаны с ним. Например, функции загрузки файлов могут быть уязвимы для обхода пути, сообщения между пользователями могут быть уязвимы для XSS, а функции связи с нами могут быть уязвимы для внедрения SMTP. Примеры уязвимостей, обычно связанных с конкретными функциями и технологиями.

Необходимо сформулировать план атаки, определив приоритеты наиболее интересных функций и наиболее серьезных потенциальных уязвимостей, связанных с ними. Необходимо использовать свой план, чтобы определить количество времени и усилий, которые вы уделяете каждой из оставшихся областей этой методологии.

2.3 Тестирование компонентов расширения браузера

Ниже представлены основные этапы тестирование компонентов расширения браузера (в соответствии с рисунком 2.3):

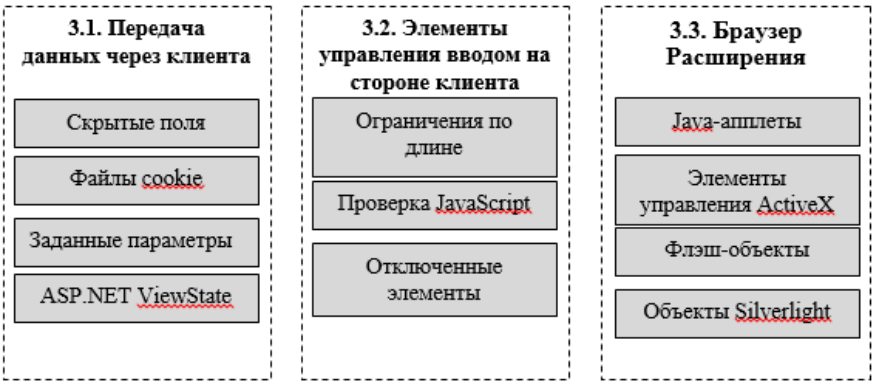

Рисунок 2.3 – Тестирование элементов управления на стороне клиента

2.3.1 Тестовая передача данных через клиента

Нужно найти все экземпляры приложения, в которых скрытые поля формы, файлы cookie и параметры URL, по-видимому, используются для передачи данных через клиента.

Необходимо попытаться определить цель, которую элемент играет влогике приложения, основываясь на контексте, в котором он появляется, а также на его имении значении.

Следует изменить значение элемента таким образом, чтобы оно соответствовало его роли в функциональности приложения. Необходимо определить, обрабатывает ли приложение произвольные значения, представленные в поле, и может ли этот факт быть использован для вмешательства в логику приложения или подрыва любых элементов управления безопасностью.

Если приложение передает непрозрачные данные через клиента, можно атаковать его различными способами. Если элемент запутан, можно расшифровать алгоритм запутывания и, следовательно, представить произвольные данные в непрозрачном элементе. Даже если он надежно зашифрован, можно воспроизвести элемент в других контекстах, чтобы вмешаться в логику приложения.

Если приложение использует ASP.NET ViewState, Нужно проверить, можно ли его подделать или содержит ли он какую-либо конфиденциальную информацию. Нужно обратить внимание, что состояние представления может использоваться по-разному на разных страницах приложения.

Необходимо использовать анализатор ViewState в Burp Suite, чтобы подтвердить, включена ли опция EnableViewStateMac, что означает, что содержимое ViewState не может быть изменено.

Следует просмотреть декодированное состояние представления, чтобы определить любые содержащиеся в нем конфиденциальные данные.

Следует изменить одно из значений декодированного параметра, повторно закодируйте и отправить состояние представления. Если приложение принимает измененное значение, вы должны рассматривать состояние представления как входной канал для ввода произвольных данных в обработку приложения. Следует выполнить такое же тестирование содержащихся в нем данных, как и для любых других параметров запроса.

2.3.2 Тестирование элементов управления вводом на стороне клиента с помощью данных пользователя

Необходимо определить любые случаи, когда элементы управления на стороне клиента, такие как ограничения длины и проверки JavaScript, используются для проверки пользовательских данных перед их отправкой на сервер. Эти элементы управления можно легко обойти, поскольку можно отправлять произвольные запросы на сервер. Например: <form action=”order.asp” onsubmit=”return Validate(this)”>

<inputmaxlength=”3” name=”quantity”>

Нужно проверить каждое затронутое поле ввода по очереди, отправив ввод, который обычно блокируется элементами управления на стороне клиента, чтобы проверить, реплицируются ли они на сервере.

Возможность обхода проверки на стороне клиента не обязательно представляет собой какую-либо уязвимость. Тем не менее, вы должны внимательно изучить, какая проверка выполняется. Следует убедиться, что приложение полагается на элементы управления на стороне клиента для защиты от искаженного ввода. Также подтвердите, существуют ли какие-либо условия эксплуатации, которые могут бытьвызваны таким вводом.

Следует просмотреть каждую HTML-форму, чтобы определить любые отключенные элементы, такие как серые кнопки отправки. Например:

<inputdisabled=”true” name=”product”>

Если будут обнаружены, следует отправить их на сервер вместе с другими параметрами формы. Необходимо посмотреть, влияет ли этот параметр на обработку сервера, которую можно использовать в атаке. Кроме того, необходимо использовать правило автоматического прокси-сервера для автоматического включения отключенных полей, таких как правила “модификации HTML” прокси-сервера Burp.

2.3.4 Тестирование компонентов расширения браузера

Понимание операции клиентского приложения

Следует настроить локальный прокси-сервер перехвата для рассматриваемой клиентской технологиии контролируйте весь трафик, проходящий между клиентом и сервером. Если данные сериализуются, необходимо использовать инструмент десериализации, такой как встроенная поддержка AMF Burpили пользовательский плагин Burp для Java.

Нужно пройти через функциональность, представленную в клиенте. Необходимо определить любые потенциально чувствительные или мощные функции, используя стандартные инструменты в прокси-сервере перехвата для воспроизведения ключевых запросов или изменения ответов сервера.

Необходимо определить все апплеты, используемые приложением. Следует искать любой из следующих типов файлов, запрашиваемых через ваш прокси-сервер перехвата:

.class, .jar : Java

.swf : Flash

.xap : Silverlight

Также следует искать теги апплетов в исходном коде HTML страниц приложений. Например:

<applet code=”input.class” id=”TheApplet” codebase=”/scripts/”></applet>

Следует просмотреть все вызовы методов апплета из вызывающего HTML и необходимо определить, передаются ли данные, возвращенные из апплета, на

сервер. Если эти данные непрозрачны (то есть запутаны или зашифрованы), чтобы изменить их, вам, вероятно, потребуется декомпилировать апплет, чтобы получить его исходный код.

- Загрузите байт-код апплета, введя URL-адрес в свой браузер, и сохраните файл локально. Имя файла байт-кода указывается в атрибуте кода тега апплета. Файл будет расположен в каталоге, указанном в атрибуте codebase, если он присутствует. В противном случае он будет находиться в том же каталоге, что и страница, на которой отображается тег апплета.

Необходимо использовать подходящий инструмент для декомпиляции байт-кода в исходный код. Например:

C:\>jad.exe input.class

Parsing input.class... Generating input.jad

Вот несколько подходящих инструментов для декомпиляции различных компонентов расширения браузера:

- Java — Jad
- Flash — SWFScan, Flasm/Flare
- Silverlight — .NET Reflector

Если апплет упакован в файл JAR, XAP или SWF, можно распаковать его с помощью стандартного средства чтения архивов, такого как WinRAR или WinZip.

Следует просмотреть соответствующий исходный код (начиная с реализации метода, возвращающего непрозрачные данные), чтобы понять, какая обработкавыполняется.

Необходимо определить, содержит ли апплет какие-либо общедоступные методы, которые можно использовать для выполнения соответствующей об фускации при произвольном вводе.

Если это не так, следует изменить источник апплета, чтобы нейтрализовать любую выполняемую им проверку. Затем можно перекомпилировать исходный код в исходный формат файла с помощью инструментов компиляции, предоставленных поставщиком.

Для больших клиентских приложений часто бывает непомерно сложно декомпилировать все приложение, модифицировать его и переупаковывать, не сталкиваясь с многочисленными ошибками. Для этих приложений, как правило, быстрее подключить отладчик среды выполнения к процессу. JavaSnoop делает этоочень хорошо для Java. Silverlight Spy-это свободно доступный инструмент, который позволяет отслеживать выполнение клиентов Silverlight.

Нужно найти ключевые функции и значения, используемые приложением для управления бизнес логикой, связанной с безопасностью, и поместите брэйкпойнты при вызове целевой функции. Следует изменить аргументы или возвращаемое значение по мере необходимости, чтобы повлиять на обход безопасности.

2.4 Тестирование механизма аутентификации

Основные вопросы тестирования механизма аутентификации представлены ниже (в соответствии с рисунком 2.4):

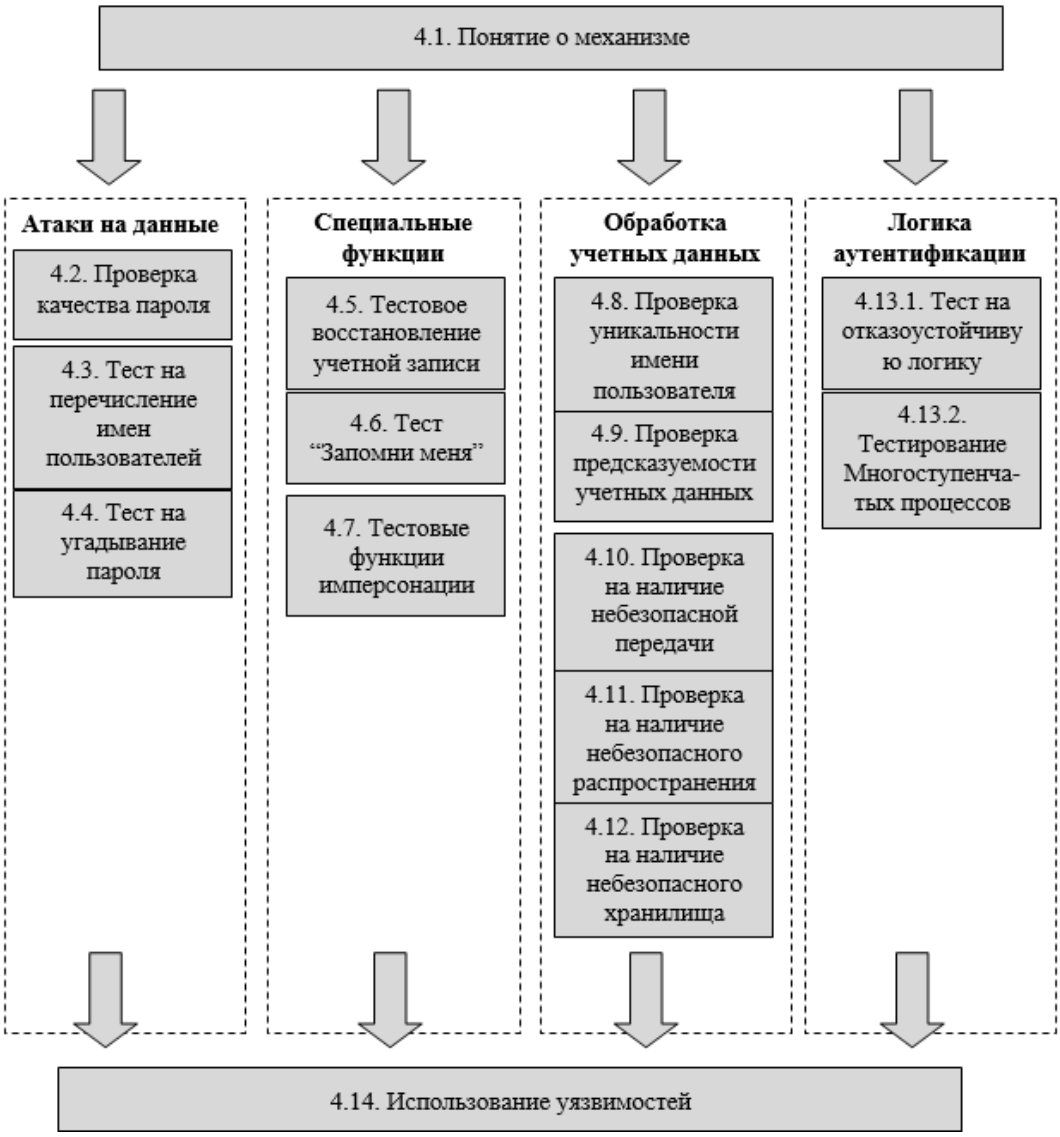

Рисунок 2.4 – Тестирование механизма аутентификации

2.4.1 Понятие о механизме

Следует установить используемые технологии аутентификации (например, формы, сертификаты или мультифактор).

Нужно найти все функции, связанные с проверкой подлинности (включая вход в систему, регистрацию, восстановление учетной записи и т. д.).

Если приложение не реализует механизм автоматической саморегистрации, необходимо определить, существуют ли какие-либо другие способы получения нескольких учетных записей пользователей.

2.4.2 Проверка качества пароля

Следует просмотреть приложение для любого описания правил минимального качества, применяемых к паролям пользователей.

Попытка установить различные виды слабых паролей, используя любые функции регистрации или смены пароля, чтобы установить правила, фактически применяемые. Нужно использовать короткие пароли, только буквенные символы, только символы в одном регистре, словарные слова и текущее имя пользователя.

Проверка на неполную проверку учетных данных. Следует установить надежный и сложный пароль (например, 12 символов со смешанными буквами, цифрами, и типографские знаки). Следует попробовать войти в систему, используя различные варианты этого пароля, удалив последний символ, изменив регистр символов и удалив любые специальные символы. Если какая-либо из этих попыток входа в систему окажется успешной, продолжайте систематически экспериментировать, чтобы определить, какая проверка на самом деле выполняется.

Установив минимальные правила качества пароля и степень проверки пароля, необходимо определить диапазон значений, которые необходимо будет использовать для атаки с использованием пароля, чтобы иметь хорошую вероятность на успех. Необходимо попытаться найти любые встроенные учетные записи, которые, возможно, не попадали под стандартные требования к сложности паролей.

2.4.3 Тест на перечисление имен пользователей

Необходимо определить каждое местоположение в рамках различных функций аутентификации, в которых передается имя пользователя, в том числе с помощью поле ввода на экране, поля скрытой формы или файла cookie. Общие местоположения включают основной вход в систему, регистрацию, смену пароля, выход из системы и восстановление учетной записи.

Для каждого местоположения следует отправить два запроса, содержащих действительное и недействительное имя пользователя. Следует просмотреть каждую деталь ответов сервера на каждую пару запросов, включая код состояния HTTP, любые перенаправления, информацию, отображаемую на экране, любые различия, скрытые в источнике HTML-страницы, и время, необходимое серверу для ответа. Нужно обратить внимание, что некоторые различия могут быть незначительными (например, одно и то же сообщение об ошибке может содержать незначительные типографские различия). Можно использовать функцию истории вашего прокси-сервера перехвата для просмотра всего трафика на сервер и с

сервера. WebScarabимеет функцию сравнения двух ответов, чтобы быстро выделить любые различия между ними.

Если будет замечено какие-либо различия между ответами, в которых указано действительное инедействительное имя пользователя, повторите тест с другой парой значения и уверенность в том, что существует систематическая разница, которая может послужить основой для автоматического перечисления имен пользователей.

Нужно проверить наличие любых других источников утечки информации в приложении, которые могут позволить вам составить список допустимых имен пользователей. Примерами могут служить функции ведения журнала, фактические списки зарегистрированных пользователей и прямое упоминание имен или адресов электронной почты в комментариях к исходному коду.

Нужно найти любую вспомогательную аутентификацию, которая принимает имя пользователя, и необходимо определить, можно ли ее использовать для перечисления имени пользователя. Следует обратить особое внимание на страницу регистрации, которая позволяет указать имя пользователя.

2.4.4 Проверка на устойчивость к угадыванию пароля

Необходимо определить каждое местоположение в приложении, в котором передаются учетные данные пользователя. Обычно двумя основными экземплярами являются основная функция входа в систему и функция смены пароля. Последний обычно является допустимой целью для атак с угадыванием пароля, только если можно указать произвольное имя пользователя.

В каждом месте, используя учетную запись, которую вы контролируете, вручную следует отправить несколько запросов, содержащих действительное имя пользователя, но другие недействительные учетные данные. Следите за ответами приложения, чтобы выявить любые различия. Примерно после 10 неудачных входов в систему, если в приложении и не вернулось сообщение о блокировке учетной записи, следует отправить запрос, содержащий действительные учетные данные. Если этот запрос будет выполнен успешно, политика блокировки учетной записи, вероятно, не действует.

Если не контролируется какие-либо учетные записи, необходимо попытаться перечислить или угадать действительное имя пользователя и сделайте несколько недействительных запросов, используя это предположение, отслеживая любые сообщения об ошибках о блокировке учетной записи. Конечно, вы должны знать, что этот тест может привести к приостановке или отключению учетной записи, принадлежащей другому пользователю.

2.4.5 Тестирование любой функции восстановления учетной записи

Необходимо определить, содержит ли приложение какую-либо возможность для пользователей восстановить контроль над своей учетной

записью, если они забыли свои учетные данные. На это часто указывает ссылка "Забыли пароль" рядом с основной функцией входа в систему.

Необходимо определить, как работает функция восстановления учетной записи, выполнив полный обзор процесса восстановления с помощью учетной записи, которую вы контролируете.

• Если функция использует вызов, такой как секретный вопрос, необходимо определить, могут ли пользователи устанавливать или выбирать свой собственный вызов во время регистрации. Если это так, необходимо использовать список перечисленных или общих имен пользователей, чтобы собрать список проблем, и следует просмотреть его для любых, которые кажутся легко угадываемыми.

• Если функция использует подсказку пароля, Следует выполнить те же действия, чтобы собрать список подсказок пароля и определить те, которые кажутся легко угадываемыми.

Следует выполнить те же тесты для любых задач восстановления учетной записи, которые выполнив в основной функции входа в систему, чтобы оценить уязвимость к атакам автоматического угадывания.

Если функция включает в себя отправку электронной почты пользователю для завершения процесса восстановления, нужно найти любые слабые места, которые могут позволить вам взять под контроль учетные записи других пользователей. Необходимо определить, возможно ли контролировать адрес, на который отправляется электронное письмо. Если сообщение содержит уникальный URL-адрес восстановления, получите несколько сообщений, используя адрес электронной почты, который вы контролируете, и необходимо попытаться определить любые шаблоны, которые могут позволить вам предсказать URL-адреса, выданные другим пользователям.

2.4.6 Тестирование функции «Запомнить меня»

Если основная функция входа в систему или ее вспомогательная логика содержит функцию Remember Me, активируйте ее и следует просмотреть ее эффекты. Если эта функция позволяет пользователю входить в систему в последующих случаях без ввода каких-либо учетных данных, вы должны внимательно изучить ее на наличие любых уязвимостей.

Внимательно нужно проверить все постоянные файлы cookie, которые устанавливаются при активации функции " Remember Me". Следует искать любые данные, которые явно идентифицируют пользователя или, по-видимому, содержат какой-либо предсказуемый идентификатор пользователя.

Даже в тех случаях, когда сохраненные данные кажутся сильно закодированными или запутанными, внимательно изучите это и сравните результаты запоминания нескольких очень похожих имен пользователей и/или

паролей, чтобы определить любые возможности для обратного проектирования исходных данных.

В зависимости от ваших результатов, следует изменить содержимое вашего файла cookie подходящим образом, пытаясь замаскироваться под других пользователей приложения.

2.4.7 Тестирование функции имперсонации

Если приложение содержит какие-либо явные функциональные возможности, позволяющие одному пользователю выдавать себя за другого, внимательно нужно проверить это на наличие любых уязвимостей, которые могут позволить вам выдавать себя за произвольных пользователей без надлежащей авторизации.

Поиск любых предоставленных пользователем данных, которые используются для определения цели имперсонации. Необходимо попытаться манипулировать этим, чтобы выдавать себя за других пользователей, особенно пользователей с правами администратора, что может позволить повысить привилегии.

2.4.8 Проверка уникальности имени пользователя

Если в приложении есть функция регистрации, позволяющая указать желаемое имя пользователя, следует попробовать дважды зарегистрировать одно и то же имя пользователя с разными паролями.

Если приложение блокирует вторую попытку регистрации, можно использовать это поведение для перечисления зарегистрированных имен пользователей.

Если приложение регистрирует обе учетные записи, следует выполнить дальнейшие исследования, чтобы определить его поведение при столкновении имени пользователя и пароля. Необходимо попытаться изменить пароль одной из учетных записей, чтобы он соответствовал паролю другой. Кроме того, следует попробовать зарегистрировать две учетные записи с одинаковыми именами пользователей и пароли.

Если приложение предупреждает или выдает ошибку при столкновении имени пользователя и пароля, вы, вероятно, можете использовать это для выполнения автоматической атаки угадывания, чтобы обнаружить пароль другого пользователя. Необходимо выбрать перечисленное или угаданное имя пользователя и необходимо попытаться создать учетные записи с этим именем пользователя и другими паролями. Когда приложении еотклоняет указанный пароль, вы, вероятно, нашли существующий пароль для целевой учетной записи.

Если приложение, по-видимому, допускает столкновение имени пользователя и пароля без ошибок войдите в систему, используя конфликтующие учетные данные. Необходимо определить, что происходит и можно ли

использовать поведение приложения для получения несанкционированного доступа к учетным записям других пользователей.

2.4.9 Проверка предсказуемости автоматически сгенерированных учетных данных

Если приложение автоматически генерирует имена пользователей или пароли, следует попробовать получить несколько значений в быстрой последовательности и определить любые обнаруживаемые последовательности или шаблоны.

Если имена пользователей генерируются предсказуемым образом, экстраполируйте их в обратном направлении, чтобы получить список возможных допустимых имен пользователей. Можно использовать это в качестве основы для автоматического подбора паролей и других атак.

Если пароли генерируются предсказуемым образом, экстраполируйте шаблон, чтобы получить список возможных паролей, выданных другим пользователям приложения. Это может быть объединено с любыми списками имен пользователей, которые вы получите, чтобы выполнить атаку на угадывание пароля.

2.4.10 Проверка на небезопасную передачу учетных данных

Нужно пройти через все функции, связанные с проверкой подлинности, которые включают передачу учетных данных, включая основной логин, регистрацию учетной записи, смену пароля и любую страницу, которая позволяет просматривать или обновлять информацию о профиле пользователя. Контролируйте весь трафик, проходящий в обоих направлениях между клиентом и сервером, используя перехватывающий прокси-сервер.

Идентифицировать каждый случай, в котором учетные данные передаются в любом направлении. Можно установить правила перехвата в своем прокси-сервере для отправки сообщений, содержащих определенные строки.

Если учетные данные когда-либо передаются в строке запроса URL-адреса, они потенциально уязвимы для раскрытия в истории браузера, на экране, в журналах сервера и в заголовке *Referer* при переходе по ссылкам третьих лиц.

Если учетные данные когда-либо хранятся в файле cookie, они потенциально уязвимы для раскрытия с помощью атак XSS или локальных атак на конфиденциальность.

Если учетные данные когда-либо передаются с сервера клиенту, они могут быть скомпрометированы с помощью любых уязвимостей в управлении сеансами или контроле доступа, а также в результате атаки XSS.

Если учетные данные когда-либо передаются по незашифрованному соединению, они уязвимы для перехвата подслушивающим устройством.

Если учетные данные передаются по протоколу HTTPS, но сама форма входаз агружается по протоколу HTTP, приложение уязвимо для атаки «man-in-the-middle» ("человек посередине"), которая может быть использована для захвата учетных данных.

2.4.11 Проверка на небезопасное распространение учетных данных

Если учетные записи создаются через какой-либо внеполосный канал или приложение имеет функцию саморегистрации, которая сама по себе не определяет все исходные учетные данные пользователя, Следует установить средства, с помощью которых учетные данные распространяются новым пользователям. Распространенные методы включают отправку сообщения на адрес электронной почты или почтовый адрес.

Если приложение генерирует URL-адреса активации учетных записей, которые распространяются вне диапазона, Следует попробовать зарегистрировать несколько новых учетных записей в тесной последовательности и определить любую последовательность в полученных вами URL-адресах. Если шаблон можно определить, следует попробовать предсказать URL-адреса, отправленные недавним и будущим пользователям, и Необходимо попытаться использовать эти URL-адреса, чтобы стать владельцем их учетных записей.

Следует попробовать повторно использовать один URL-адрес активации несколько раз и необходимо посмотреть, позволяет ли это приложение. Если это не так, следует попробовать заблокировать целевую учетную запись до повторного использования URL-адреса и Необходимо посмотреть, работает ли оно по-прежнему. Необходимо определить, позволяет ли это установить новый пароль для активной учетной записи.

2.4.12 Проверка на наличие небезопасного хранилища

Если были получены доступ к хэшированным паролям, нужно проверить наличие учетных записей с одинаковым значением хэшированного пароля. Следует попробовать войти в систему с помощью общих паролей для наиболее распространенного хэшированного значения.

Необходимо использовать автономную радужную таблицу для рассматриваемого алгоритма хеширования, чтобы восстановить значение открытого текста.

2.4.13 Тест на логические ошибки

Для каждой функции, в которой приложение проверяет учетные данные пользователя, включая функции смены логина и пароля, Следует выполнить процесс обычным способом, используя учетную запись, которую вы контролируете. Нужно обратить внимание на каждый параметр запроса, отправленный в приложение.

Повторите процесс несколько раз, изменяя каждый параметр по очереди различными неожиданными способами, предназначенными для вмешательства вл огику приложения. Для каждого параметра включите следующие изменения:

Следует отправить пустую строку в качестве значения.

Удалите пару имя/значение.

Отправляйте очень длинные и очень короткие значения.

Отправляйте строки вместо чисел, и наоборот.

Отправляйте один и тот же именованный параметр несколько раз с одинаковыми и разными значениями.

Внимательно изучите ответы приложения на предыдущие запросы. Если возникнут какие-либо неожиданные расхождения с базовым случаем, верните это наблюдение в свою структуру дальнейших тестовых случаев. Если одна модификация вызывает изменение в поведении, Следует попробовать объединить ее с другими изменениями, чтобы довести логику приложения до предела.

2.4.13.2 Проверка многоступенчатого механизма

Если какая-либо функция, связанная с проверкой подлинности, включает в себя отправку учетных данных в серии различных запросов, Необходимо определить очевидную цель каждого отдельного этапа и Нужно обратить внимание на параметры, представленные на каждом этапе.

Повторите процесс несколько раз, изменяя последовательность запросов таким образом, чтобы нарушить логику приложения, включая следующие тесты.

Нужно пройти все этапы, но в другой последовательности, чем предполагалось.

Переходите непосредственно к каждому этапу по очереди и продолжайте нормальную последовательность оттуда.

Нужно пройти через нормальную последовательность несколько раз, пропуская каждый этап по очереди и продолжая нормальную последовательность со следующего этапа.

На основе наблюдений и очевидной цели каждого этапа механизма Следует попробовать подумать о дальнейших способах изменения последовательности и доступа к различным этапам, которые разработчики, возможно, не ожидали.

Необходимо определить, отправлена ли какая-либо отдельная информация (например, имя пользователя)более чем на одном этапе, либо возможно она захватывается более одного раза от пользователя, либо потому, что она передается через клиента в скрытом поле формы, файле cookie или заданном параметре строки запроса. Если это так, Следует попробовать представить разные значения на разных этапах (как действительные, так и недействительные)и наблюдайте за эффектом. Необходимо попытаться определить, является ли отправленный элемент иногда сверхсложным, или проверен на одном этапе, а затем доверен впоследствии, или проверен на разных этапах с помощью разных проверок.

Необходимо попытаться использовать поведение приложения для получения несанкционированного доступа или снижения эффективности контроля, введенного механизмом.

Следует искать любые данные, передаваемые через клиента, которые не были получены от пользователя в какой-либо момент. Если скрытые параметры используются для отслеживания состояния процесса на последовательных этапах, можно вмешаться в логику приложения, изменив эти параметры различными способами.

Если какая-либо часть процесса связана с тем, что приложение представляет случайно изменяющуюся проблему, Нужно проверить наличие двух распространенных дефектов.

Если параметр, определяющий вызов, отправляется вместе с ответом пользователя, Необходимо определить, можете ли вы эффективно выбрать свой собственный вызов, изменив это значение.

Следует попробовать несколько раз перейти к одному и тому же вызову с одним и тем же именем пользователя и определить, представлен ли другой вызов. Если это так, можно эффективно выбрать свой собственный вызов, переходя к этому этапу несколько раз, пока не будет представлен желаемый вызов.

2.4.14 Необходимо использовать любые уязвимости для получение несанкционированного доступа

Следует просмотреть все выявленные вами уязвимости в различных функциях аутентификации и Необходимо определить все, которые можно использовать для достижения своих целей при атаке на приложение. Обычно это включает в себя попытку аутентификации от имени другого пользователя — если это возможно, пользователя с правами администратора.

Перед установкой любого вида автоматической атаки обратите внимание на все средства защиты от блокировки учетной записи, которые вы идентифицировали. Например, при выполнении перечисления имени пользователя для функции входа в систему отправляйте общий пароль с каждым запросом, а не полностью произвольное значение чтобы не тратить впустую неудачную попытку входа в систему на каждое обнаруженное имя пользователя. Аналогично, выполняйте любые атаки с угадыванием пароля. Начните свой список слов с наиболее распространенных слабых паролей и нужно пройти по этому списку, пробуя каждый элемент против каждого перечисленного имени пользователя.

Учитывайте правила качества паролей и полноту проверки паролей при построении списков слов для использования в любой атаке с использованием пароля, чтобы избежать невозможных или сверхсложных тестовых случаев.

4.14.4 Необходимо использовать методы, описанные выше, чтобы максимально автоматизировать процесс, работайте как можно усерднее и максимизируйте скорость и эффективность ваших атак.

2.4 Проверка контроля доступа

Проверка контроля доступа осуществляется в несколько этапов как показано на рисунке 2.5.

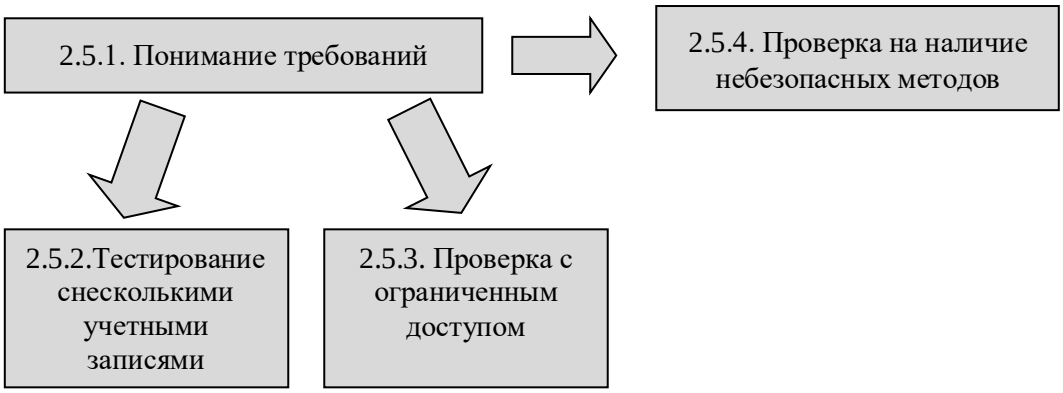

Рисунок 2.5 – Тестирование средств контроля доступа

2.5.1 Понимание требований к контролю доступа

Основываясь на основных функциях, реализованных в приложении, изучите широкие требования к контролю доступа с точки зрения вертикальной сегрегации (разные уровни пользователей имеют доступ к различным типам функций) и горизонтальной сегрегации (пользователи с одинаковым уровнем привилегий имеют доступ к различным подмножествам данных) Часто присутствуют оба типа сегрегации. Например, обычные пользователи могут иметь доступ к своим собственным данным, в то время как администраторы могут получить доступ к данным всех пользователей.

Следует просмотреть результаты сопоставления приложений, чтобы определить области функциональности и типы ресурсов данных, которые представляют собой наиболее эффективные цели для атак с повышением привилегий.

Для наиболее эффективного тестирования уязвимостей контроля доступа в идеале необходимо получить несколько различных учетных записей с различными вертикальными и горизонтальными привилегиями. Если возможна самостоятельная регистрация, вы, вероятно, можете получить последнюю непосредственно из приложения. Чтобы получить первое, вам, вероятно, потребуется сотрудничество владельца приложения(или вам нужно будет использовать некоторую уязвимость, чтобы получить доступ к высоко

привилегированному счету). Доступность различных типов учетных записей повлияет на типы тестирования, которые можно выполнить, как описано ниже.

2.5.2 Тестирование с несколькими учетными записями

Если приложение применяет вертикальное разделение привилегий, сначала необходимо использовать привилегированную учетную запись, чтобы найти все функции, к которым оно может получить доступ. Затем необходимо использовать менее привилегированную учетную запись и необходимо попытаться получить доступ к каждому элементу этой функции.

Используя Burp, просматривайте все содержимое приложения в одном пользовательском контексте.

Следует просмотреть содержимое карты сайта Burp, чтобы убедиться, что вы определили все функции, которые хотите протестировать. Затем необходимо выйти из приложения и снова войдите в систему, используя другой пользовательский контекст. Необходимо использовать контекстное меню, чтобы выбрать функцию “compare site maps”, чтобы определить, какие запросы с высоким уровнем привилегий могут быть доступны пользователю с более низким уровнем привилегий.

Если приложение применяет горизонтальное разделение привилегий, Следует выполнить эквивалентный тест с использованием двух разных учетных записей на одном и том же уровне привилегий, пытаясь использовать одну учетную запись для доступа к данным, принадлежащим другой учетной записи. Обычно это включает замену идентификатора (например, идентификатора документа) в запросе на указание ресурса, принадлежащего другому пользователю.

Выполнить ручную проверку логики управления доступом к ключам.

Для каждой привилегии пользователя Следует просмотреть ресурсы, доступные пользователю. Необходимо попытаться получить доступ к этим ресурсам из неавторизованной учетной записи пользователя, повторив запрос с использованием маркера сеанса неавторизованного пользователя.

При выполнении любого вида проверки контроля доступа обязательно проверяйте каждый шаг многоступенчатых функций по отдельности, чтобы убедиться, что контроль доступа был должным образом реализован на каждом этапе или приложение предполагает, что пользователи, которые получают доступ на более позднем этапе, должны пройти проверки безопасности, реализованные на более ранних этапах. Например, если административная страница, содержащая форму, должным образом защищена, нужно проверить, подвергается ли фактическая отправка формы также надлежащему контролю доступа.

2.5.3 Проверка с ограниченным доступом

Если у вас нет предварительного доступа к учетным записям с разными уровнями привилегий или к нескольким учетным записям с доступом к разным данным, тестирование на наличие нарушенных элементов управления без доступа не так просто. Многие распространенные уязвимости будет гораздо труднее обнаружить, поскольку вы не знаете имен URL-адресов, идентификаторов и параметров, необходимых для использования слабых мест.

В упражнениях по сопоставлению приложений, использующих учетную запись с низким уровнем привилегий, вы, возможно, идентифицировали URL-адреса для привилегированных функций, таких как административные интерфейсы. Если они не защищены должным образом, вы, вероятно, уже знаете об этом.

Следует декомпилировать все скомпилированные клиенты, которые присутствуют, и извлеките любые ссылкин а функциональность на стороне сервера.

Доступ к большинству данных, подлежащих горизонтальному контролю доступа, осуществляется с помощью идентификатора, такого как номер счета или ссылка на заказ. Чтобы проверить, эффективен ли контроль доступа с использованием только одной учетной записи, необходимо попытаться угадать или обнаружить идентификаторы, связанные с данными других пользователей. Если возможно, создайте серию идентификационных данных в быстрой последовательности(например, путем создания нескольких новых заказов). Необходимо попытаться идентифицировать любые шаблоны, которые могут позволить вам предсказать идентификаторы, выданные другим пользователям. Если нет способа генерировать новые идентификаторы, вы, вероятно, ограничены анализом тех, которые у вас уже есть, и предположением на этой основе.

Если вы нашли способ предсказать идентификаторы, выданные другим пользователям, Необходимо использовать методы, чтобы организовать автоматическую атаку для сбора содержательных данных, принадлежащих другим пользователям. Необходимо использовать функцию Извлечения Grep в Burp Intruder, чтобы захватить соответствующую информацию с ответов приложения.

2.5.4 Тест на небезопасные методы контроля доступа

Некоторые приложения реализуют управление доступом на основе параметров запроса изначально небезопасным способом. Следует искать такие параметры, как edit=false или access=read в любых запросах ключей, и изменять их в соответствии с их очевидной ролью, чтобы попытаться вмешаться в логику управления доступом приложения.

Некоторые приложения основывают решения по управлению доступом на заголовке HTTP Referer. Например, приложение может правильно контролировать доступ к/admin.jsp и принимать любой запрос, показывающий это в качестве своего Референта. Чтобы проверить это поведение, Следует выполнить некоторые

привилегированные действия, к которым вы авторизованы и отправляете отсутствующий или измененный заголовок Referer. Если это изменение приведет к тому, что приложение заблокирует ваш запрос, это может быть небезопасное использование заголовка Referer. Следует выполнить то же действие, что и неавторизованный пользователь, но укажите исходный заголовок Referer и необходимо посмотреть, удастся ли выполнить это действие.

Если на сайте разрешен метод HEAD, Нужно проверить наличие небезопасного контроля доступа к URL-адресам с использованием контейнеров. Следует сделать запрос, используя метод HEAD, чтобы определить, разрешает ли это приложение.

2.6 Проверка уязвимостей на основе входных данных

Многие важные категории уязвимостей вызываются неожиданным вводом данных пользователем и могут появляться в любом месте приложения. Эффективный способ для проверки приложение на наличие этих уязвимостей- это сопоставить каждый параметр с каждым запросом с помощью набора строк атаки(как показано на рисунке 2.6).

Рисунок 2.6 – Тестирование уязвимостей на основе входных данных

2.6.1 Размытие всех параметров запроса

Следует просмотреть результаты ваших действий по сопоставлению приложений и Необходимо определить каждый отдельный запрос клиента, который отправляет параметры, обрабатываемые приложением на стороне сервера. Соответствующие параметры включают элементы в URL-адресе строки запроса, параметры в теле запроса и файлы cookie HTTP. Также включить любые другие элементы пользовательского ввода, которые, как было замечено, влияют на поведение приложения, такие как заголовки Referer или User-Agent.

Для размытия параметров можно использовать свои собственные скрипты или готовый инструмент для размытия. Например, чтобы использовать Burp Intruder, загрузите каждый запрос в инструмент. Простой способ сделать это- перехватить запрос в Burp прокси-сервер и выбирать действие отправить

злоумышленнику или щелкнуть правой кнопкой мыши элемент в истории прокси-сервера Burp и выбрать этот параметр. Использование этой опции гарантирует Burp Intruder с содержимым запроса, а также правильным целевым хостом и портом. Он также автоматически помечает значения всех параметров запроса как позиции полезной нагрузки, готовые к фаззингу.

Используя вкладку полезные нагрузки, Следует настроить подходящий набор полезных нагрузок атаки для поиска уязвимостей в приложении. Можно ввести полезную нагрузку вручную, загрузить ее из файла или выбрать один из предустановленных списков полезной нагрузки. Размывание каждого параметра запроса в приложении обычно влечет за собой выдачу большого количества запросов и проверку результатов на наличие аномалий. Если ваш набор строк атаки слишком велик, это может быть контрпродуктивно и привести к чрезмерно большому объему выходных данных для просмотра. Следовательно, разумный подход заключается в том, чтобы нацелиться на ряд распространенных уязвимостей, которые часто могут быть легко обнаружены в аномальных ответах на конкретны обработанные входные данные и которые часто проявляются в любом месте приложения, а не в конкретных типах функций. Вот подходящий набор полезных нагрузок, которые можно использовать для тестирования некоторых распространенных категорий уязвимостей:

SQL-инъекция
‘
‘--
‘; waitfor delay ‘0:30:0’--
1; waitfordelay ‘0:30:0’—

XSS и Инъекция заголовка
xsstest
“><script>alert(‘xss’)</script>

Внедрение команд ОС
|| ping -i 30 127.0.0.1 ; x || ping -n 30 127.0.0.1 &
| ping –i 30 127.0.0.1 |
| ping –n 30 127.0.0.1 |
& ping –i 30 127.0.0.1 &
& ping –n 30 127.0.0.1 &
; ping 127.0.0.1 ;
%0a ping –i 30 127.0.0.1 %0a
` ping 127.0.0.1 `

Обход Пути

../../../../../../../../../../etc/passwd
../../../../../../../../../../boot.ini
..\..\..\..\..\..\..\..\..\..\..\etc\passwd
..\..\..\..\..\..\..\..\..\..\..\boot.ini

Инъекция скрипта
;echo 111111
echo 111111
response.write 111111
:response.write 111111

Включение файла
http://<your server name>/
http://<nonexistentIPaddress>/

Все предыдущие полезные нагрузки показаны в их буквальном виде. Персонажи?, ;, &, +, =, и пространство должно быть закодировано URL-адресом, потому что они имеют особое значение в HTTP-запросах. По умолчанию Burp Intruder выполняет необходимую кодировку этих символов, поэтому Следует убедиться, что эта опция нео тключена. (Чтобы восстановить все параметры по умолчанию после предыдущей настройки, Необходимо выбрать Burp ' Восстановить значения по умолчанию.)

В функции Grep Burp Intruder сконфигурируйте подходящий набор строк для удаления некоторых распространенных сообщений об ошибках в ответах. Например:

error
exception
illegal
invalid
fail
stack
access
directory
file
not found
varchar
ODBC
SQL
SELECT
111111

Нужно обратить внимание, что строка 111111 включена для проверки успешных атак внедрения скриптов. Полезные нагрузки на шаге 7.1.3 включают запись этого значения в ответ сервера.

Также Необходимо выбрать опцию Payload Grep, чтобы откликаться на ответы, содержащие саму полезную нагрузку, указывая на потенциальную уязвимость XSS или инъекции заголовка.

Следует настроить веб-сервер или прослушиватель netcat на хосте, указанном в полезной нагрузке включения первого файла. Это поможет вам следить за подключением попытки, полученные от сервера в результате успешной атаки удаленного включения файла.

Начать атаку. Когда атака будет завершена, Следует просмотреть результаты на предмет аномальных ответов, указывающих на наличие уязвимостей. Нужно проверить наличие расхождений в коде состояния HTTP, длине ответа, времени ответа, появлении настроенных выражений и появлении самой полезной нагрузки. Можно щелкнуть заголовок каждого столбца в таблице результатов, чтобы отсортировать результаты по значениям в этом столбце (и щелкнуть правой кнопкой мыши, чтобы выполнить обратную сортировку результатов). Это позволяет быстро идентифицировать любые аномалии, которые выделяются на фоне других результатов.

Для каждой потенциальной уязвимости, указанной в результатах тестирования fuzz, обратитесь к следующим разделам этой методологии. Они описывают подробные шаги, которые вы должны предпринять в отношении каждой категории проблем, чтобы проверить наличие уязвимости и успешно использовать ее.

После того, как вы убедили Burp Intruder выполнить тест на нечеткость одного запроса, можно быстро повторить тот же тест для других запросов в приложении. Просто Необходимо выбрать каждый целевой запрос в прокси-сервере Burp и необходимо выбрать опцию Отправить злоумышленнику. Затем немедленно следует запустить атаку внутри Злоумышленника, используя существующую конфигурацию атаки. Таким образом, вы можете запускать большое количество тестов одновременно в отдельных окнах и вручную просматривать результаты по мере завершения работы каждого теста.

Если ваши действия по картографированию идентифицировали какие-либо внеполосные входные каналы таким образом, что управляемый пользователем ввод может быть введен в обработку приложения, вы должны выполнить аналогичное упражнение по размыванию на этих входных каналах. Отправляйте различные обработанные данные, предназначенные для запуска распространенных уязвимостей при обработке в веб-приложении. В зависимости от характера входного канала для этой цели может потребоваться создать пользовательский сценарий или другой жгут проводов.

В дополнение к вашей собственной обработке запросов приложений, если у вас есть доступ к автоматизированному сканеру уязвимостей веб-приложений, вы должны запустить это против целевого приложения, чтобы обеспечить основу для сравнения с вашими собственными результатами.

2.6.2 Тест на SQL-инъекцию

Если строки атаки SQL, перечисленные в шаге 7.1.3, приводят к каким-либо аномальным ответам, Нужно проверить обработку приложением соответствующего параметра вручную, чтобы определить, присутствует ли уязвимость внедрения SQL.

Если были возвращены какие-либо сообщения об ошибках базы данных, изучите их значение. Необходимо использовать раздел синтаксис SQL,чтобы помочь интерпретировать сообщения об ошибках на некоторых распространенных платформах баз данных.

Если отправка одинарной кавычки в параметре вызывает ошибку или другое аномальное поведение, следует отправить две одинарные кавычки. Если этот ввод приводит к исчезновению ошибки или аномального поведения, приложение, вероятно, уязвимо для инъекции SQL.

Следует попробовать использовать общие функции конкатенатора строк SQL для построения строки, эквивалентной доброкачественному входу. Если это вызывает тот же ответ, что и исходный доброкачественный ввод, приложение, вероятно, уязвимо. Например, если исходными входными данными является выражение FOO, можно выполнить этот тест, используя следующие элементы (в третьем примере Нужно обратить внимание на пробел между двумя кавычками):

‘||’FOO
‘+’FOO
‘ ‘FOO

Обязательно кодируйте символы URL, такие как + и пробел, которые имеют особое значение в HTTP-запросах.

Если исходные входные данные являются числовыми, Следует попробовать использовать математическое выражение, эквивалентное исходному значению. Например, если исходное значение было2, Следует попробовать отправить 1+1 или 3-1. Если приложение реагирует таким же образом, оно может быть уязвимым, особенно если значение числового выражения оказывает систематическое влияние на поведение приложения.

Если предыдущий тест прошел успешно, можно получить дополнительную уверенность в том, что уязвимость SQL-инъекции связана с использованием математических выражений, специфичных для SQL, для построения определенного значения. Если логикой приложения можно систематически манипулировать таким образом, оно почти наверняка уязвимо для SQL-инъекции. Например, оба следующих элемента эквивалентны числу 2:

67-ASCII(‘A’)

51-ASCII(1)

Если какой-либо из тестовых случаев fuzz с использованием команды waitfor привел к ненормальной задержке времени до того, как приложение ответило, это являетс ясильным показателем того, что тип базы данных MS-SQL и приложение уязвимо для SQL-инъекций. Повторите тест вручную, указав различные значения в параметре waitfor, и Необходимо определить, систематически ли время, необходимое для ответа, зависит от этого значения. Нужно обратить внимание, что полезная нагрузка атаки может быть вставлена в несколько SQL-запросов, поэтому наблюдаемая временная задержка может быть фиксированной кратной указанному значению.

Если приложение уязвимо для внедрения SQL, подумайте, какие виды атак возможны и, вероятно, помогут вам достичь ваших целей. Подробные шаги, необходимые для выполнения любой из следующих атак

Следует изменить условия в предложении WHERE, чтобы изменить логику приложения (например, путем инъекции или 1=1-для обхода входа в систему).

Необходимо использовать оператор UNION для ввода произвольного запроса SELECT и объединения результатов с результатами исходного запроса приложения.

Введите тип базы данных, используя специфичный для базы данных синтаксис SQL.

Если тип базы данных-MS-SQL и приложение возвращает сообщения об ошибках ODBC в своих ответах, необходимо использовать их для перечисления структуры базы данных и извлечения произвольных данных.

Если не получается найти способ прямого извлечения результатов с произвольного запроса, необходимо использовать следующие расширенные методы для извлечения данных:

- Извлекать строковые данные в числовой форме по одному байту за раз.
- Необходимо использовать внеполосный канал.

Если можно вызвать различные ответы приложений на основе одного произвольного условия, необходимо использовать Absinthe для извлечения произвольных данных по одному биту за раз.

- Если можно вызвать временные задержки на основе одного произвольного условия, Необходимо использовать их для извлечения данных по одному биту за раз.

- Если приложение блокирует определенные символы или выражения, необходимые для выполнения конкретной атаки, Следует попробовать различные методы обхода, чтобы обойти входной фильтр.

- По возможности усиливайте атаку на базу данных, используя любые уязвимости или мощные функции в базе данных.

2.6.3 Тест на XSS и другие ответные инъекции

Идентификация отраженных параметров Запроса

Необходимо отсортировать результаты тестирования fuzz, щелкнув столбец Grep полезной нагрузки, и Необходимо определить любые совпадения, соответствующие полезным нагрузкам XSS, перечисленным вшаге 7.1.3. Это случаи, когда строки теста XSS были возвращены в неизмененном виде в ответах приложения.

В каждом из этих случаев Следует просмотреть ответ приложения, чтобы найти местоположение предоставленного ввода. Если это появляется в теле ответа, Нужно проверить наличие уязвимостей XSS. Если входные данные отображаются в любом заголовке HTTP, Нужно проверить для уязвимостей инъекции заголовка. Если они используются в заголовке местоположения ответа 302 или используются для указания перенаправления каким-либо другим способом, Нужно проверить наличие уязвимостей перенаправления. Нужно обратить внимание, что один и тот же ввод может быть скопирован в несколько мест в ответе и что может присутствовать более одного типа отраженной уязвимости.

Для каждого места в теле ответа, гдео тображается значение параметра запроса, Следует просмотреть окружающий HTML-код, чтобы определить возможные способы обработки вашего ввода, чтобы вызвать выполнение произвольного JavaScript. Например, можно ввести теги <script>, внедрить их в существующий сценарий или поместить созданное значение в атрибут тега.

Необходимо использовать различные методы отбивания фильтров на основе сигнатур, в качестве ссылки на различные способы, с помощью которых созданный ввод может быть использован для выполнения JavaScript.

Следует попробовать представить приложению различные возможные эксплойты и отслеживать его ответы, чтобы определить, выполняется ли какая-либо фильтрация или очистка входных данных. Если ваша строка атаки возвращается без изменений, необходимо использовать браузер, чтобы окончательно убедиться, что вам удалось выполнить произвольный JavaScript (например, создав диалоговое окно предупреждения).

Если вы обнаружите, что приложение блокирует ввод, содержащий определенные символы или выражения, которые вам необходимо использовать, или кодирует определенные символы в формате HTML, Следует попробовать различные обходы фильтра.

Если вы обнаружите уязвимость XSS в запросе POST, она все равно может быть использована через вредоносный веб-сайт, содержащий форму с требуемыми параметрами и скрипт для автоматической отправки формы. Тем не менее, более широкий спектр механизмов доставки атак доступен, если эксплойт может быть доставлен с помощью запроса GET. Следует попробовать отправить те же параметры в запросе GET и необходимо посмотреть, удастся ли атака.

Можно использовать действие Change Request Method в прокси-сервере Burp для преобразования запроса для вас.

Для каждого места в заголовках ответов, гдео тображается значение параметра запроса, Нужно проверить, принимает ли приложение данные, содержащие кодированные URL-адресом символы возврата carriage-return (%0d) и перевода строки line-feed (%0a), а также Нужно проверить, возвращаются ли они в его ответе без обработки. (Нужно обратить внимание, что вы ищете сами символы новой строки, которые появятся в ответе сервера, а не их эквиваленты, закодированные в URL-адресе.)

Если новая строка появляется в заголовках ответов сервера при вводе обработанного ввода, значит приложение уязвимо для инъекции заголовка HTTP.Это может быть использовано для выполнения различных атак.

Если обнаружится, что в ответах сервера возвращается только один из двух символ в новой строки, все еще может быть возможно создать рабочий эксплойт, в зависимости от контекста и браузера целевого пользователя.

Если вы обнаружите, что приложение блокирует ввод, содержащий символы новой строки, или очищает эти символы в своем ответе, следует попробовать следующие элементы ввода для проверки эффективности фильтра:

foo%00%0d%0abar

foo%250d%250abar

foo%%0d0d%%0a0abar

Если отраженные входные данные используются для указания цели какого-либо перенаправления, Нужно проверить, можно ли предоставить обработанные входные данные, которые приводят к произвольному перенаправлению на внешний веб-сайт. Если это так, то такое поведение может быть использовано для придания достоверности атаке в стиле фишинга.

Если приложение обычно передает абсолютный URL-адрес в качестве значения параметра, Следует изменить доменное имя в URL-адресе и нужно проверить, перенаправляет ли приложение вас в другой домен. Если параметр обычно содержит относительный URL-адрес, следует изменить его на абсолютный URL-адрес для другого домена и Нужно проверить, перенаправляет ли приложение вас на этот домен.

Если приложение выполняет некоторую проверку параметра перед выполнением перенаправления, чтобы предотвратить внешнее перенаправление, это часто уязвимо для обхода. Следует попробовать различные атаки чтобы проверить надежность фильтров.

Если приложение хранит элементы ввода, предоставленные пользователем, и позже отображает их на экране, после того, как вы размыли все приложение, можно наблюдать, как некоторые из ваших строк атаки возвращаются в ответ на запросы, которые сами по себе не содержали этих строк. Нужно обратить

внимание на любые случаи, когда это происходит, и необходимо определить исходную точку входа для сохраняемых данных.

- В некоторых случаях предоставленные пользователем данные успешно сохраняются только в том случае, если вы завершаете многоступенчатый процесс, который не выполняется при базовом тестировании fuzz. Если ваши действия по сопоставлению приложений идентифицируют любую функциональность такого рода, нужно вручную исполнить соответствующий процесс и проверить сохраненные данные на наличие уязвимостей XSS.

- Если у вас есть достаточный доступ для его тестирования, внимательно изучите любую административную функциональность, в которой данные, полученные от пользователей с низкими привилегиями, в конечном итоге отображаются на экране в сеансе более привилегированных пользователей. Любые сохраненные уязвимости XSS в функциях такого рода обычно приводят непосредственно к повышению привилегий.

- Проверить каждый экземпляр, в котором хранятся и отображаются предоставленные пользователем данные для пользователей. Нужно проверить их на наличие XSS и других атак с ответной инъекцией, описанных ранее.

- Если вы обнаружите уязвимость, при которой ввод, предоставленный одним пользователем, отображается другим пользователям, Необходимо определить наиболее эффективную полезную нагрузку атаки, с помощью которой можно достичь своих целей, таких как захват сеанса или подделка запроса. Если сохраненные данные отображаются только для того же пользователя, от которого они были получены, Следует попробовать найти способы соединения любых других обнаруженных уязвимостей (например, нарушенных элементов управления доступом), чтобы внедрить атаку в сеансы других пользователей.

- Если приложение позволяет загружать и скачивать файлы, всегда проверяйте эту функцию на наличие сохраненных атак XSS. Если приложение допускает HTML, JAR или текстовые файлы и не проверяет или не очищает их содержимое, оно почти наверняка уязвимо. Если оно допускает файлы JPEG и не проверяет, содержат ли они допустимые изображения, вероятно приложение уязвимо для атак против пользователей Internet Explorer. Нужно проверить, как приложение обрабатывает каждый тип файла, который оно поддерживает, и подтвердите, как браузеры обрабатывают ответы, содержащие HTML вместо обычного типа контента.

- В каждом месте, где данные, предоставленные одним пользователем, отображаются другим пользователям, но где фильтры приложения не позволяют вам выполнить сохраненную атаку XSS, Нужно проверить, не делает ли поведение приложения уязвимым для подделки запросов на месте.

2.6.4 Тест на внедрение команд ОС

Если какая-либо из строк атаки внедрения команд, привела к ненормальной задержке времени до того, как приложение ответило, это является сильным показателем того, что приложение уязвимо для внедрения команд ОС. Повторите тест, вручную указав различные значения в-i или-параметр n и Необходимо определить, систематически ли изменяется время, затраченное на ответ, в зависимости от этого значения.

Используя любую из строк ввода, которая была признана успешной, Следует попробовать ввести более интересную команду (например, ls или dir) и Необходимо определить можете ли вы получить результаты команды в свой браузер.

Если вы не можете получить результаты напрямую, вам доступны другие варианты.

Можно попытаться открыть внеполосный канал обратно на свой компьютер. Следует попробовать использовать TFTP для копирования инструментов на сервер, используя telnet или netcat для создания обратной оболочки обратно на компьютер и используя команду mail для отправки выходных данных команды через SMTP.

Можно перенаправить результаты ваших команд в файл в корневом каталоге веб-сайта, который затем можно получить непосредственно с помощью браузера. Например:

dir > c:\inetpub\wwwroot\foo.txt

Если был найден способ вводить команды и получать результаты, следует определить свой уровень привилегий (используя whoami или аналогичную команду или пытаясь записать безвредный файл в защищенный каталог).Затем можно попытаться повысить привилегии, получить бэкдор-доступ к конфиденциальным данным приложений или атаковать другие хосты, доступ к которым можно получить с взломанного сервера.

Если кажется, что ввод передается какой-либо команде операционной системы, но перечисленные строки атаки не увенчались успехом, Необходимо посмотреть, возможно ли использовать символ < или > для направления содержимого файла на вход команды или для направления вывода команды на файл. Это может позволить читать или записывать произвольное содержимое файла. Если вы знаете или можете угадать фактическую выполняемую команду, Следует попробовать ввести параметры командной строки, связанные с этой командой, чтобы изменить ее поведение полезными способами(например, указав выходной файл в корневом каталоге веб-сайта).

Если обнаружится, что приложение экранирует определенные ключевые символы, необходимые для выполнения атаки с введением команд, Следует попробовать поместить экранирующий символ перед каждым таким символом. Если приложение не экранирует сам экранирующий символ, это обычно приводит

к обходу этой защитной меры. Если вы обнаружите, что пробелы заблокированы или очищены, можно использовать $IFS вместо пробелов на платформах UNIX.

2.6.5 Проверка прохождения пути

Для каждого выполненного теста fuzz следует просмотреть результаты, полученные с помощью строк атаки обхода пути, перечисленных в шаге 7.1.3. Можно щелкнуть верхнюю часть столбца полезной нагрузки в Burp Intruder, чтобы отсортировать таблицу результатов по полезной нагрузке и сгруппировать результаты для этих строк. В любых случаях, когда было получено необычное сообщение об ошибке или ответ с ненормальной длиной, следует просмотреть ответ вручную, чтобы определить, содержит ли он содержимое указанного файла или другие доказательства того, что произошла аномальная операция файла.

При отображении поверхности атаки приложения нужно отметить любую функциональность, которая конкретно поддерживает чтение и запись файлов на основе предоставленных пользователем входных данных. В дополнение к общему размыванию всех параметров, вы должны вручную очень тщательно протестировать эту функциональность, чтобы выявить любые существующие уязвимости при обходе пути.

Если параметр содержит имя файла, часть имени файла или каталог, следует изменить существующее значение параметра, чтобы вставить произвольный подкаталог и одну последовательность обхода. Например, если приложение отправляет этот параметр:

file=foo/file1.txt

Следует попробовать отправить это значение:

file=foo/bar/../file1.txt

Если поведение приложения идентично в обоих случаях, оно может быть уязвимым, и вам следует перейти к следующему шагу. Если поведение отличается, приложение может блокировать, удалять или очищать последовательности обхода, что приводит к недопустимому пути файла. Следует попробовать использовать кодирование и другие атаки, в попытке обойти фильтры.

Если предыдущий тест использования последовательностей обхода в базовом каталоге прошел успешно, Следует попробовать использовать дополнительные последовательности, чтобы подняться над базовым каталогом и получить доступ к известным файлам в операционной системе сервера. Если эти попытки не увенчаются успехом, приложение может вводить различные фильтры или проверки перед предоставлением доступа к файлам. Вам следует продолжить исследование, чтобы понять, какие элементы управления реализованы и существуют ли какие-либо обходы.

Приложение может проверять запрашиваемое расширение файла и разрешать доступ только к определенным типам файлов. Следует попробовать

использовать нулевой байт или атака новой строки вместе с известным принятым расширением файла в попытке обойти фильтр. Например:

../../../../../boot.ini%00.jpg

../../../../../etc/passwd%0a.jpg

Приложение может проверять, что путь к файлу, предоставленный пользователем, начинается с определенного каталога или файла. Следует попробовать добавить последовательности обхода после известного принятого стержня в попытке обойти фильтр. Например:

/images/../../../../../../../etc/passwd

Если эти атаки не увенчались успехом, следует попробовать объединить несколько обходов, работайте изначально полностью в базовом каталоге в чтобы выявить фильтры и способы, которыми приложение обрабатывает неожиданные входные данные.

Если удастся получить доступ на чтение к произвольным файлам на сервере, необходимо попытаться получить любой из следующих файлов, что может позволить вам усилить атаку.

Файлы паролей для операционной системы и приложения

Файлы конфигурации сервера и приложений для обнаружения других уязвимостей или точной настройки другой атаки

Включить файлы, которые могут содержать учетные данные базы данных

Источники данных, используемые приложением, такие как файлы базы данных MySQL или XML-файлы

Исходный код на исполняемые на сервере страницы, чтобы выполнить проверку кода в поисках ошибок

Файлы журнала приложений, которые могут содержать такую информацию, как имена пользователей и маркеры сеанса

Если удастся получить доступ на запись к произвольным файлам на сервере, Нужно проверить, возможны ли какие-либо из следующих атак для эскалации вашей атаки:

Создание сценариев в папках запуска пользователей

Изменение файлов, таких как in.ftpd, для выполнения произвольных команд при следующем подключении пользователя

Запись сценариев в веб-каталог с разрешениями на выполнение и вызов их из браузера

2.6.6 Тест на внедрение скрипта

Для каждого выполненного теста fuzz следует просмотреть результаты для строки111111 (то есть которая не предшествует остальной части тестовой строки).Можно быстро идентифицировать их в Burp Intruder, щелкнув правой кнопкой мыши заголовок строки 111111 Grep, чтобы сгруппировать все результаты, содержащиеэ ту строку. Следует искать те, у которых нет проверки в

столбце Grep полезной нагрузки. Любые выявленные случаи, вероятно, будут уязвимы для внедрения команд сценариев.

•Следует просмотреть все тестовые случаи, в которых использовались строки внедрения скриптов, и Необходимо определить любые содержащие сообщения об ошибках сценариев, которые могут указывать на то, что ваш ввод выполняется, но вызвали ошибку. Возможно, их потребуется настроить, чтобы выполнить успешную инъекцию скрипта.

Если приложение кажется уязвимым, Нужно проверить это, введя дополнительные команды, указанные в используемой платформе сценариев. Например, можно использовать полезные нагрузки атаки, аналогичные тем, которые используются при фаззинге для внедрения команд ОС:

system(‘ping%20127.0.0.1’)

2.6.7 Проверка на включение файла

Если был получен какие-либо входящие HTTP-соединения от инфраструктуры целевого приложения во время фаззинга, приложение почти наверняка уязвимо для удаленного включения файлов. Повторите соответствующие тесты однопоточным и регулируемым по времени способом, чтобы точно определить, какие параметры заставляют приложение выдавать HTTP-запросы.

Следует просмотреть результаты тестов включения файлов и необходимо определить все, что вызвало аномальную задержку в ответе приложения. В этих случаях, возможно, само приложение уязвимо, но в результате время ожидания HTTP-запросов истекает из-за фильтров сетевого уровня.

Если обнаружится уязвимость удаленного включения файлов, разверните веб-сервер, содержащий вредоносный сценарий, специфичный для языка, на который вы ориентируетесь, и необходимо использовать команды, подобные тем, которые используются для проверки внедрения сценария, чтобы убедиться, что ваш сценарий выполняется.

3 РАЗРАБОТКА СКАНЕРА ВЕБ-УЯЗВИМОСТЕЙ

3.1 Используемые технологии

После анализа проблем и исследования технической части проблемы было решено разработать программный комплекс, отвечающий следующим технико-технологическим требованиям:

- комплекс должен быть написан на простом языке программирования, желательно скриптовом;
- должны быть использованы простые в написании и понимании структуры и модели;

- необходимо использовать микросервисный подход при разработке комплекса;

- по возможности использовать методы защиты информации такие как JWT, SSL/TLS;

- взаимодействие с пользователем и другими программами должно осуществляться через API;

- входные и выходные данные должны представляться в универсальном виде JSON;

- комплекс в приоритете должен разворачиваться под управлением операционной системы LINUX или другой UNIX подобной

Под простым языком программирования понимается язык, легкий в изучении и чтении, распространённый в IT среде, для максимально дешевой и быстрой разработки, а также внедрения и сопровождения программного комплекса [10]. В качестве такого языка программирования предложено использовать язык Python, в связи с его легко читаемым синтаксисом и распространенностью.

Под простыми структурами и моделями понимаются элементы программ, написанные по стандартным шаблонам, которые позволяют быстро и просто разрабатывать программные средства, а также легко исправлять ошибки на этапе отладки, из-за понятной и простой структуры всего проекта.

При передаче данных важно обеспечивать их целостность и конфиденциальность, чтобы не допускать утечек и компрометации, в любых ее видах. При передаче данных через протокол HTTP не гарантируется их целостность, потому что данные не шифруются и передаются через всю сеть в открытом видел, что позволяет проводить атаки ManInTheMiddle, цель которых получение исходных данных, их искажение, либо сбор. Для этого были разработаны две технологии SSL/TLS, которые позволяют передавать данные по протоколу HTTPS, с асинхронным двусторонним шифрованием. Задачи SSL/TLS соединения:

- обеспечение приватности с помощью шифрования передаваемых данных

- аутентификация с помощью сертификатов

- обеспечение достоверности данных с помощью проверки целостности данных

Сертификат открытого ключа (public key certificate, digital certificate или identity certificate, а также сертификат, сертификат ключа подписи, сертификат ключа проверки электронной подписи) - электронный документ, подтверждающий принадлежность публичного ключа его владельцу.

Сертификат выдаётся и подписывается Certificate Authorities, которые подписывают выданные сертификаты своей цифровой подписью, подтверждая, что владелец сертификата (и публичного ключа, включённого в этот сертификат) был проверен:

- TLS сертификат как правило включает в себя поляVersionNumber: версия сертификата (см X.509)

- Serial Number: используется для идентификации сертификата его Certificate Authorities

- Signature Algorithm: алгоритм, использованный для подписи публичного ключа

- Issuer: кем выдан и подписан сертификат (Certificate Authority)

- Not Before и Not After: срок действия (валидности) сертификата

- Subject name: комупринадлежитсертификат

- Subject Public Key Info

- Public Key Algorithm: алгоритм, использованный для подписи публичного ключа

- Subject Public Key: сам публичный ключ владельца сертификата

- Certificate Signature Algorithm: алгоритм, использованный для подписи сертификата

- Certificate Signature: цифровая подпись сертификата

- Asymmetric encryption (также Public Key Cryptography - криптография по открытому ключу) использует пары ключей - публичный и приватный. Эти ключи связаны друг с другом таким образом, что данные, зашифрованные с помощью одного ключа (как правило публичного) могут быть расшифрованы другим (приватным).

Любое SSL-соединение использует набор протоколов, описанных в RFC 5246 (тут и ниже рассматривается TLS v 1.2 и этот RFC, как последняя актуальная версия на момент написания этого поста - март 2018 г.).

В RFC 5246 они описаны в обратном порядке, но тут рассмотрим их начиная с Handshake Protocol, т.к. именно в нём описывается процесс установления сессии и создания ключей, которые далее используются в TLS Record Protocol.

Структура получается следующая:

1. The TLS Handshaking Protocols
2. Handshake Protocol
3. Change Cipher Spec Protocol
4. Alert Protocol
5. The TLS Record Protocol
6. RecordLayer

Веб-токен JSON (JWT) - это открытый стандарт (RFC 7519) для создания токенов доступа на основе формата JSON. Обычно используется для передачи аутентификационных данных в клиент-серверных приложениях. Токены создаются сервером, подписываются секретным ключом и передаются клиенту, который затем использует этот токен для проверки своей личности.

APT-токен состоит из трех частей: заголовка, полезной нагрузки и подписи или данных шифрования. Первые два элемента-это объекты JSON определенной структуры. Третий элемент вычисляется на основе первого и зависит от выбранного алгоритма (в случае использования беззнакового JWT он может быть опущен). Токены могут быть перекодированы в компактное представление (JWS / JWE Compact Serialization): алгоритм кодирования Base64-URL применяется к заголовку и полезной нагрузке, после чего добавляется подпись и все три элемента разделяются точками (".").

Программный комплекс в первую очередь должен выполнять свой непосредственный функционал и оправдывать разработку, а именно:

- проводить тестирование инфраструктуры на открытые порты;
- перебирать поддоменные имена сайтов;
- проводить сканирование на наличие уязвимостей типа SQLi;
- по возможности определять тип используемой ОС;
- по возможности проводить BrutForce (Перебор) паролей, для поиска ненадежных и распространённых паролей.

Программный комплекс должен разрабатываться по итерационному подходу жизненного цикла ПО, с возможностью на любом этапе производства внести корректировки. Пример итерационного цикла ПО показан на рисунке 3.1:

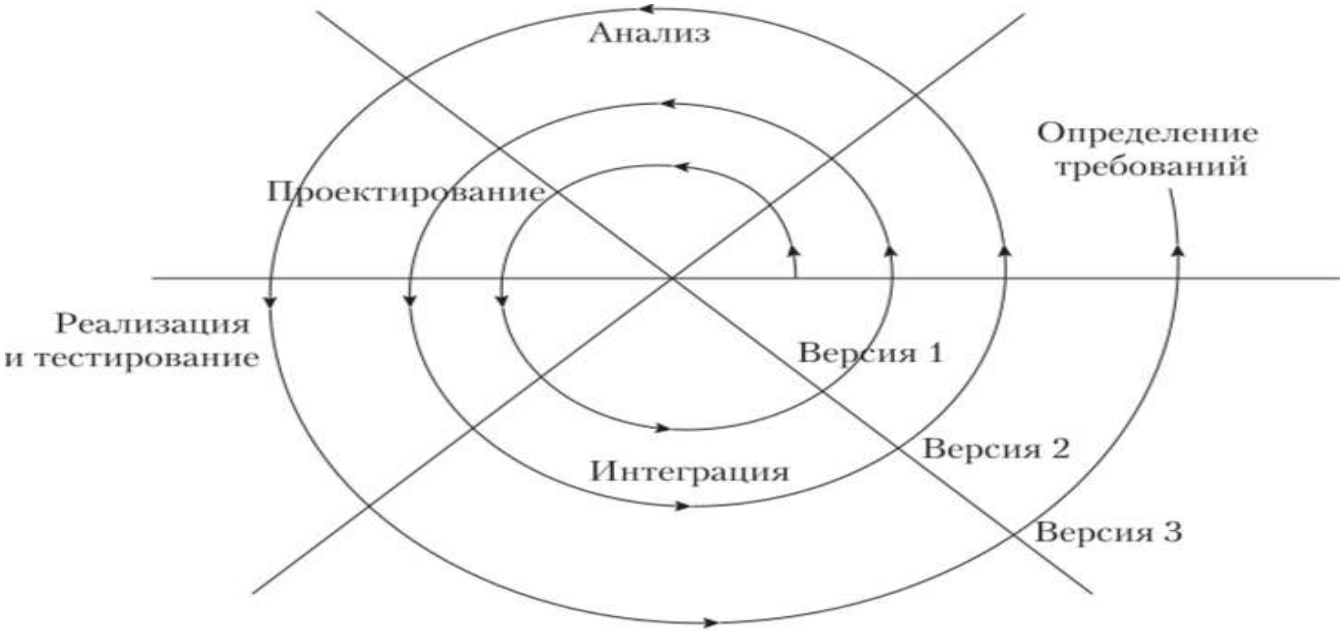

Рисунок 3.1 - Итерационный жизненный цикл ПО

3.2 Алгоритм сканирования системы

В дипломной работе использовался большой перечень технологий, ИС и решений, который обеспечивает правильное и надежное функционирование ПО. Список используемых технологий:

- ОС Linux Ubuntu Server 20.04
- графическая среда рабочего стола KDE
- средство контейнеризации Docker
- веб сервер NGINX
- WSGI сервер JUNICORN
- язык программирования Python
- фреймворк Flask для Python
- JWT
- интерактивная оболочка BASH

Linux - это семейство Unix-подобных операционных систем ("ОС"), использующих ядро Linux, разработанное финско-американским программистом Линусом Торвальдсом. ОС, использующие ядро Linux, называются дистрибутивами Linux, и это те же операционные системы, что и Microsoft Windows, Apple macOS, iOS, но с одной очень важной особенностью, а именно-их исходные коды открыты, так как они распространяются под лицензией GNU GPL, что подразумевает создание свободного и открытого программного обеспечения (open - source software). Это означает, что любой пользователь имеет право изучать и изменять исходный код.

Она передает процессору необходимые для обработки входные данные и затем, при помощи аппаратного обеспечения компьютера, отображает выходной результат вычислений. Описанный процесс является основной функцией операционной системы. Стоит отметить, что она также выполняет и множество других не менее важных задач (но это уже выходит за рамки данной статьи).

Linux существует вокруг нас с середины 1990-х годов. Можно встретить его повсюду: в наших телефонах, ноутбуках, в наручных часах, в суперкомпьютерах, автомобилях и даже в холодильниках. Он обрел известность, как среди разработчиков, так и среди обычных пользователей компьютеров. В сообществе программистов существует спор об именовании операционных систем, использующих ядро Linux и программное обеспечение, разработанное под лицензией GNU GPL. Поскольку ядро Linux само по себе не является работающей операционной системой, то многие предпочитают использовать термин «GNU/Linux». Структура Linux приведена на рисунке 3.2:

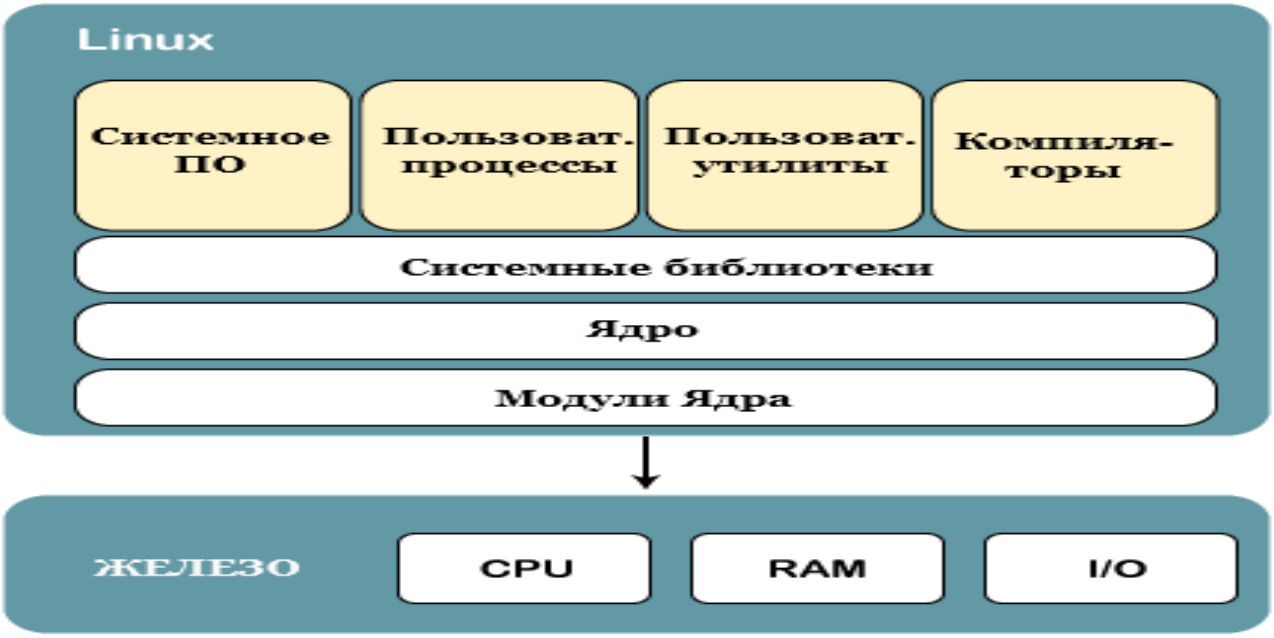

Рисунок 3.2 - Строение Linux

Ядро - это своего рода главная программа, являющаяся основной частью операционной системы. Оно выступает в роли посредника между устройствами компьютера (процессором, видеокартой, оперативной памятью и т.д.) и его программным обеспечением, абстрагируя от обычных программ и пользователей сложную, низкоуровневую работу с «железом» компьютера, предоставляя взамен простой, понятный и удобный в использовании интерфейс. Для этого в код ядра были включены драйверы устройств, которые могут, как загружаться в память вместе с ядром ОС, так и подключаться по мере возникновения потребности в ресурсах необходимого устройства.

Микроядро - это ядро, состоящее из нескольких подгружаемых в память по мере надобности независимых модулей, выполняющихся в отдельных адресных пространствах. По сути, грубо говоря, в таком варианте исполнения оно не сильно отличается от обычных прикладных программ. К достоинствам данного ядра можно отнести теоретически большую надежность в сравнении с другими архитектурами (в действительности же - не всё так радужно и гладко) и его модульность (легкость в подключении дополнительных частей ядра). К минусам же микроядерной архитектуры относится то, что ядро, построенное по такой схеме, получается очень медленным (ведь ему нужно постоянно переключаться между отдельными частями).

Монолитное ядро - это полная противоположность микроядра, т.к. в памяти компьютера всегда находится весь (или почти весь) код ядра. Вследствие чего, скорость его работы выше в сравнении с микроядром [11].

Гибридное ядро - это ядро, сочетающее в себе элементы как монолитной, так и микроядерной архитектур. Ядро Linux хоть и относится к монолитным ядрам, но оно также заимствует и некоторые идеи из микроядерной архитектуры, что означает, что вся операционная система работает в пространстве ядра, а драйвера устройств (в виде модулей) могут быть легко загружены (или выгружены) прямо во время работы операционной системы. На следующем рисунке 3.3 показана архитектура системы Linux:

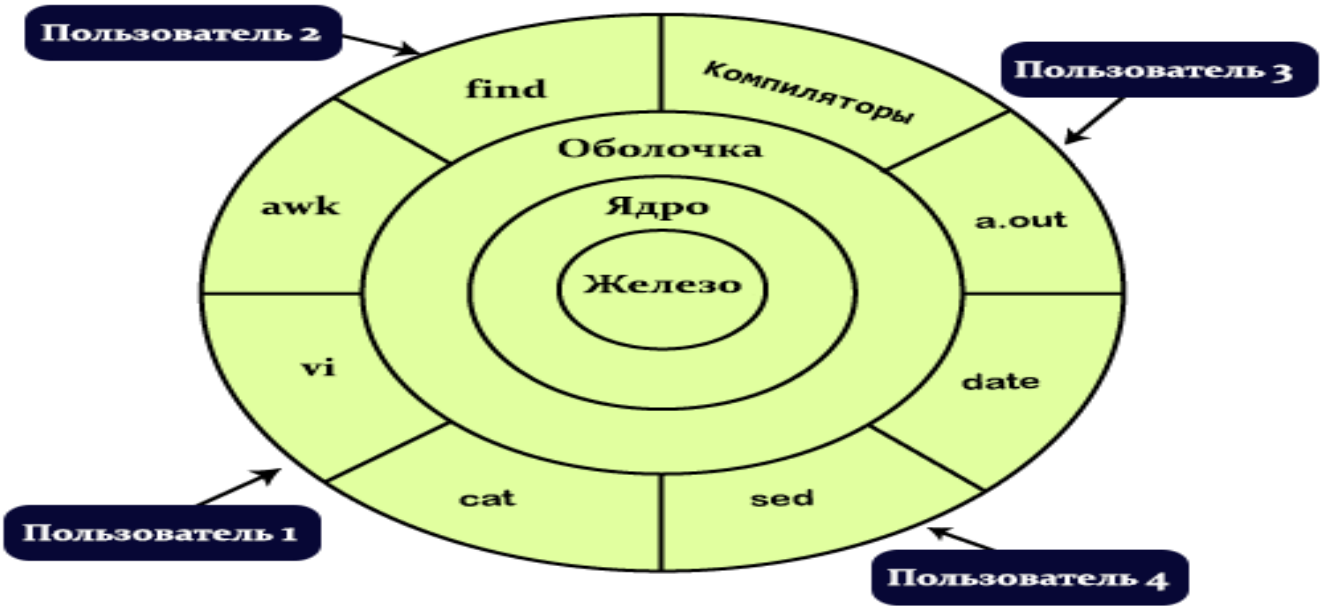

Рисунок 3.3 - Архитектура Linux

«Железо» - аппаратное обеспечение компьютера (процессор, видеокарта, оперативная память и т.д.) со всеми его периферийными устройствами.

Ядро - является основным компонентом операционной системы, взаимодействует непосредственно с аппаратным обеспечением, играя роль посредника между низкоуровневым «железом» и компонентами верхнего уровня.

Оболочка (или «командный интерпретатор») - интерфейс для взаимодействия между пользователями системы и ядром ОС, абстрагирующий внутреннее устройство системы. Принимает команды от пользователей и запускает на выполнение соответствующие функции.

Утилиты (vi, cat, sed, date и т.д.) - служебные программы, которые предоставляют пользователю большую часть функциональных возможностей операционной системы.

KDE-это международное сообщество, которое разрабатывает бесплатную среду рабочего стола KDE Plasma, набор тесно интегрированных программ для выполнения повседневной работы, а также несколько веб-сервисов. До начала 2010 года название KDE означало само программное обеспечение (сокращение от K Desktop Environment).

Программное обеспечение KDE построено на основе кросс-платформенного инструментария разработки пользовательского интерфейса Qt. Он работает в основном под UNIX-подобными операционными системами, использующими X Window System и графические подсистемы Wayland. KDE 4 частично работал на Microsoft Windows и Mac OS. На данный момент многие приложения KDE доступны как на этих платформах, так и на Android.

Будучи одним из самых известных проектов, KDE Plasma является основной настольной средой для многих дистрибутивов Linux, например openSUSE, Manjaro, Mageia, Netrunner, OpenMandriva, Chakra, Kubuntu, KaOS и PCLinuxOS.

KDE Plasma 5-это пятая и текущая настольная среда, созданная сообществом KDE для систем Linux. KDE Plasma 5 является преемником KDE Plasma 4 и впервые был выпущен в июле 15, 2014. Он содержит новую тему по умолчанию, известную как "Бриз", а также увеличенную конвергенцию между различными устройствами. Графический интерфейс был полностью портирован на QML, который использует OpenGL для аппаратного ускорения, что привело к повышению производительности и снижению энергопотребления.

LXC - это метод виртуализации на уровне ОС предназначенный для того, чтобы запускать множество изолированных систем Linux на одном хосте.

Контейнеры отделяют приложения от операционных систем. Это значит, что у пользователей есть чистая минимальная Linux ОС, и можно запускать все процессы в одном или нескольких изолированных контейнерах.

Так как операционная система отделена от контейнеров, можно перемещать контейнер на любой Linux-сервер, который поддерживает операционную среду контейнера.

Docker - это открытая платформа для разработки, доставки и эксплуатации приложений. Docker, который начался как проект, чтобы строить LXC-контейнеры под одно приложение, серьезно изменил LXC и сделал контейнеры более портативными и гибкими.

Используя контейнеры Docker, можно развертывать, копировать, переносить и делать резервные копии информации быстрее и легче, чем при помощи виртуальной машины. В принципе Docker привносит облакоподобную гибкость в любую инфраструктуру, которая может работать на контейнерах.

Хотя Docker начался как проект с открытым кодом для того, чтобы строить специализированную LXC, он позже превратился в собственную контейнерную среду исполнения. Docker - это инструмент Linux, который эффективно создает, отправляет и запускает контейнеры.

Контейнеры Docker и LXC - легковесные механизмы виртуализации в пользовательском пространстве, которые применяют контрольные группы и пространства имен, чтобы управлять изолированием ресурсов. Но между Docker и LXC есть несколько фундаментальных отличий - рассмотрим их на рисунке 3.4

VM	Docker
Аппаратная виртуализация	Виртуализация на уровне операционной системы
Каждой машине соответствует отдельная операционная система	Контейнеры работают как отдельный процесс на хосте, разделяя ядро системы
Есть возможность распределить между собой ресурсы хоста: процессор, оперативную память, дисковое пространство	С помощью cgroups можно ограничить отдельно каждый контейнер в потреблении ресурсов
Линейный расход дискового пространства, каждой машине необходимо место для гостевой операционной системы	Layed File System позволяет сэкономить место на диске, за счет повторного использования слоев
	Файловую систему одного образа, можно использовать в качестве основы для формирования новых образов

Рисунок 3.4 - Различие виртуальных машин и контейнеров

Docker ограничивает контейнеры, заставляя их работать как единый процесс. Если ваша среда приложения состоит из X одновременных процессов, Docker запустит X контейнеров, каждый со своим процессом. В отличие от Docker, LXC контейнеры могут запускать множество процессов[12].

Чтобы запустить простое многоуровневое веб-приложение в Docker, вам понадобится PHP контейнер, Nginx контейнер (веб-сервер), MySQL контейнер

(для процесса базы данных) и несколько контейнеров данных для того, чтобы хранить таблицы баз данных и другую информацию приложения.

У однопроцессных контейнеров много преимуществ, включая простые и более мелкие обновления. Вам не нужно убивать процесс баз данных, когда вы хотить обновить только веб-сервер. Также у однопроцессных контейнеров эффективная архитектура для того, чтобы строить приложения, основанные на микросервисах.

У однопроцессных контейнеров также есть ограничения. Например, вы не можете запускать агенты, скрипты регистрации или автоматически запускаемые SSH-процессы внутри контейнера. Также нелегко незначительно обновлять однопроцессный контейнер на уровне приложения. Вам придется запускать новый обновленный контейнер.

Контейнеры Docker сделаны так, чтобы быть более бесструктурными, чем LXC:

Во-первых, Docker не поддерживает внешнее хранилище. Docker обходит это тем, что позволяет вам подключать хранилище хоста в качестве тома Docker из ваших контейнеров. Так как тома подключаются, они не считаются частью среды контейнера.

Во-вторых, контейнеры Docker состоят из слоев в режиме чтения. Это значит, что как только создается образ контейнера, он не меняется. Во время выполнения программы, если процесс в контейнере меняет свое внутреннее состояние, создается разница между внутренним состоянием и образом, из которого был создан контейнер.

Если вы выполняете команду docker commit, разница между двумя версиями становится частью нового образа - не оригинального, а нового, из которого можно создавать новые контейнеры. Если вы удалите контейнер, разница версий исчезнет.

Бесструктурный контейнер - интересная сущность. Можно обновлять контейнер, но серия обновлений создаст серию новых образов контейнеров, поэтому в системе так легко откатываться.

Пожалуй, это самое важное преимущество Docker над LXC. Docker больше отделяет сетевые ресурсы, хранилище и детали ОС, чем LXC. С Docker приложение действительно не зависит от настроек этих низкоуровневых ресурсов. Когда вы перемещаете контейнер Docker от одного хоста Docker к другой машине с Docker, Docker гарантирует, что среда для приложения останется неизменной.

Прямое преимущество этого подхода - это то, что Docker помогает программистам создавать локальные среды разработки, которые выглядят как продакшн-сервер. Когда программист заканчивает писать и начинает тестировать код, он может обернуть его в контейнер, опубликовать напрямую на сервере или в приватном облаке, и он сразу будет работать, так как это одна и та же среда.

CLXC программист может запустить что-то на своей машине, но обнаружить, что код работает неправильно при разворачивании на сервере. Среда сервера будет другой, и программисту придется потратить много времени, чтобы починить эту разницу и исправить проблему. С Docker этих проблем нет.

Docker состоит из нескольких компонентов, (представленных на рисунке 3.5).

- Docker Daemon - то самое Container Engine; запускает контейнеры
- Docker CII - утилита по управлению Docker
- Dockerfile - инструкция по тому, как собирать образ
- Image - образ, из которого раскатывается контейнер
- Container
- Docker registry - хранилищеобразов.

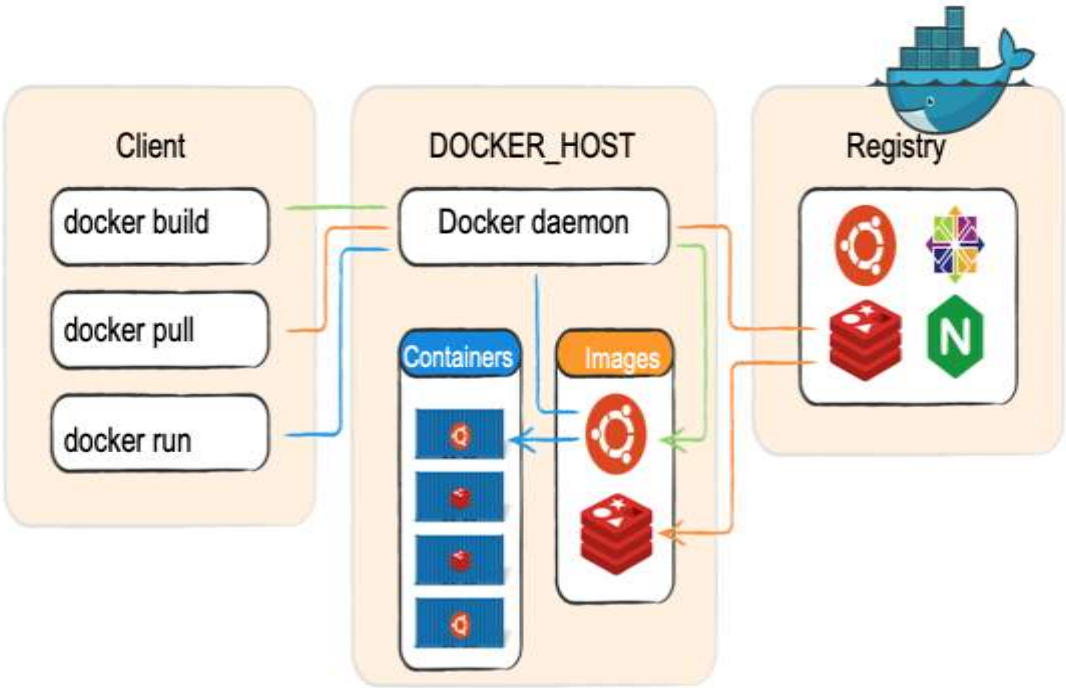

Рисунок 3.5 - Схематичное обозначение Docker контейнера

Nginx - мощный инструмент для развертывания веб-сервера, который при правильной настройке превосходит Apache. Области применения Nginx весьма обширны - от кэширования HTTP до создания инвертированного прокси-сервера.

Сейчас Nginx обслуживает примерно 30, 8% всех существующих сайтов мира, о чьих веб-серверах есть информация в открытом доступе. Понимание, что из себя представляет Nginx и как этот программный продукт можно применять на практике, помогает эффективно решать задачи во многих областях IT-индустрии.

Nginx (NGINX, Engine-X, «Энжин-кс») - это бесплатный веб- и почтовый прокси-сервер с непоточной (асинхронной) архитектурой и открытым кодом.

Разработку Nginx начали в 2002 году для Rambler, а в 2004 году стал доступен широкому кругу пользователей. Спустя 2 года в 2011 году серверное ПО запустила расширенную платную версию продукта (Nginx Plus). Весной 2019 года Nginx была выкуплена крупным американским девелопером F5 Networks.

Nginx работает на ОС Unix-типа и был успешно протестирован на OpenBSD, FreeBSD, Linux, Mac OS X, Solaris. На ОС Windows он стал доступен после выпуска бинарной сборки 0, 7.52.

На данный момент функционалом пользуются такие известные платформы: Rambler, Begun, Yandex, SourceForge.net, WordPress.com, vkontakte.ru. Статистика показывает, что Nginx используют 22, 3 млн веб-сайтов и 2, 03 млн дополнительных активных сайтов.

В отличие от обычного веб-сервера, Nginx не создаёт один поток под каждый запрос, а разделяет его на меньшие однотипные структуры, называемые рабочими соединениями. Каждое такое соединение обрабатывается отдельным рабочим процессом, а после выполнения они сливаются в единый блок, возвращающий результат в основной процесс обработки данных. Одно рабочее соединение может обрабатывать до 1024 запросов одного вида одновременно.

Веб-сервер Nginx по сравнению с Apache работает быстрее при отдаче статики и потребляет меньше серверных ресурсов. Его использует вместо или совместно с Apache для ускорения обработки запросов и уменьшения нагрузки. Это обуславливается тем, что большая часть тех возможностей, которые предлагает Apache, большинству обычных пользователей не нужно.

Несмотря на плюсы Nginx и его асинхронного алгоритма, есть один минус - блокирующие операции. Бывает так, что среди параллельно выполняющихся маленьких операций есть одна, которая задерживает весь поток. Например, весь поток может ждать ответа от жёсткого диска для одного из этапов. Это похоже на очередь в кассе супермаркета. Какое-то время вся система идёт чётко. Один человек оплачивает, второй уже выкладывает товар на ленту, кассир пробивает товар, параллельно покупатель складывает пробитый товар в пакет. Вдруг на пакете с помидорами нет штрих-кода. Покупатель бежит взвешивать. Вся очередь стоит. Процесс остановлен.

Для решения этой проблемы в версии Nginx 1.7.11 был создан новый механизм - пул потоков. Когда в процессе нужно выполнить потенциально долгую операцию, она помещается в отдельный поток, чтобы не задерживать все остальные. В нашем примере с кассой не покупатель бы побежал взвешивать помидоры, а работник супермаркета. Пока он взвешивал помидоры, кассир продолжал бы пробивать товары, а покупатель складывать их.

Раньше традиционный веб-сервер не имел возможности запускатьприложения, написанные на языке программирования Python. В

конце 1990-х был предложен модуль для веб-сервера Apache под названием mod_python для выполнения произвольного кода Python. В течение нескольких лет в конце 1990-х и начале 2000-х Apache, настроенный с помощью mod_python, запускал большинство веб-приложений Python. Были и другие средства для связи веб-сервера, и веб-приложения. Но в Python-приложениях был один существенный недостаток - отсутствие совместимости. Он был разработан только для одного FastCGI, mod_Python, CGI или других API конкретного веб-сервера, и они не допускали взаимозаменяемости. Модуль Apache mod_Python не был официальной спецификацией, и в нем были проблемы с безопасностью. Здесь и появился WSGI. У него был стандартный интерфейс для маршрутизации веб-приложений и фреймворков на веб-серверы. Фреймворк берет свое начало от CGI, или Common Gateway Interface, и использовался на заре Интернета. Успех CGI был в том, что он мог работать со многими языками, но недостатком было то, что он был медленным и ограниченным. Стандарт WSGI, созданный Филиппом Дж. Эби и опубликованный 7 декабря 2003 г., представляет собой интерфейс шлюза веб-сервера, который объясняет, как веб-сервер взаимодействует с веб-приложениями и как приложения могут быть объединены в цепочку для генерации запросов.

У стандарта WSGI имеется ряд преимуществ перед предшественниками. Ниже представлен их список с пояснением:

- гибкость. Одним из самых больших преимуществ, которые дает WSGI, является гибкость. Фактически можно изменить компоненты вебстека, не меняя код и даже не меняя приложение, на котором работают серверы WSGI;
- масштабируемость. Фреймворки не могут обрабатывать слишком много запросов. Но серверы WSGI могут, и они могут одновременно обрабатывать тысячи запросов и направлять их с веб-сервера наилучшим из возможных способов;
- скорость. Стандарт WSGI помогает ускорить разработку вебприложений на Python, потому что достаточно знать основы работы интерфейса. Многие веб-фреймворки Python поставляются с адаптером WSGI, а серверные технологии, такие как Apache, mod_python, FastCGI, CGI и т.д., могут запускать WSGI-приложение;
- простота. WSGI прост в изучении, что облегчает его освоение, не требуя настройки или установки. Это одно из самых больших преимуществ WSGI, и разработчики искренне поддерживают его;
- многоразовое ПО. Можно улучшить функциональность WSGI с помощью существующих компонентов промежуточного программного обеспечения, таких как аутентификация/авторизация, кэширование, фильтрация и т.д. Функция повторного использования экономит время.

В структуре WSGI есть две основные стороны:

- приложение WSGI, созданное из скрипта Python;
- сервер/шлюз WSGI, такой как Apache или Nginx.

Будучи вызываемым объектом, приложение-WSGI принимает два параметра. Это: среда окружения WSGI, название функции, которая запускает ответ. Каждый раз, когда оно получает запрос от HTTP-клиентов, направленный на приложение, сервер/шлюз вызывает приложение. На рисунке 3.6 показана диаграмма вызова функции.

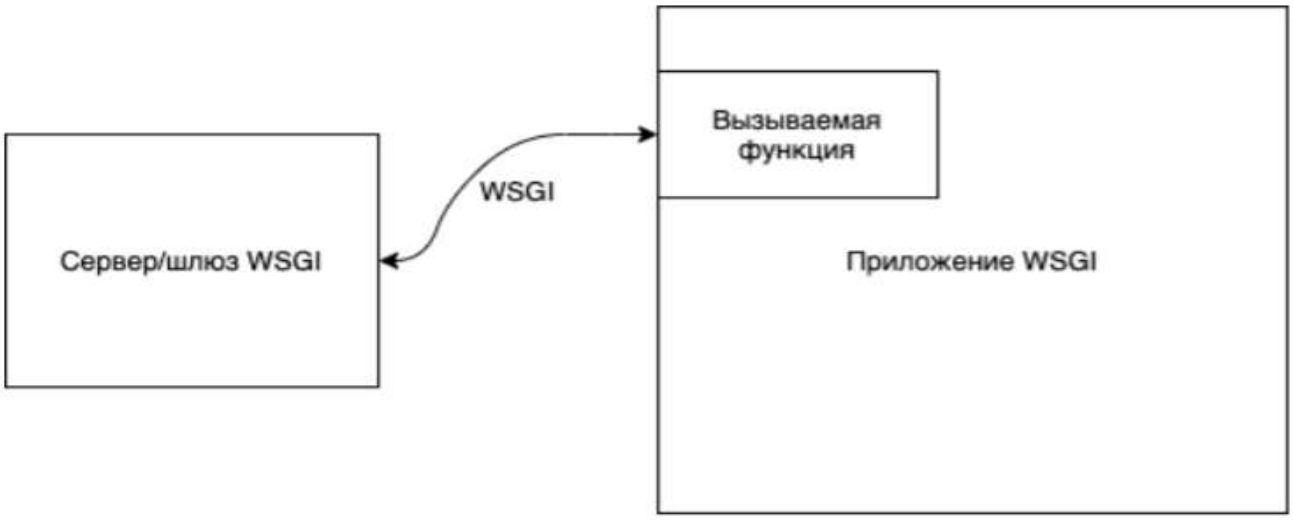

Рисунок 3.6 - Диаграмма вызова функции в стандарте WSGI

Шлюз WSGI - это сервер, который запускает веб-приложение и передает соответствующую информацию с помощью функции обратного вызова в приложение. Обработка запроса происходит на стороне приложения, в то время как сервер получает ответ, используя функцию обратного вызова. Фреймворками Python, поддерживающими WSGI, являются Django, web2py, Flask, TurboGears и CherryPy. Между веб-приложением и веб-сервером может быть несколько промежуточных программ WSGI. Их функция будет заключаться в том, чтобы направлять запросы к различным объектам приложения, предварительной обработке контента, балансировке нагрузки и так далее.

Gunicorn, созданный для использования в UNIX, представляет собой Python WSGI HTTP Server. Название является сокращенной и комбинированной версией слов «Зеленый Единорог». Сам сайт проекта на самом деле имеет зеленый единорог. Gunicorn был перенесен из проекта Unicorn из Ruby. Он относительно быстрый, требует мало ресурсов, прост в реализации и работает с различными вебфреймворками. Команда Gunicorn рекомендует вам использовать Nginx. Сервер Gunicorn работает на локальном порте 8000, а Nginx обычно используется в качестве обратного прокси-сервера. Gunicorn не имеет зависимостей.

Python - интерпретируемый, объектно-ориентированный высокоуровневый язык программирования с динамической семантикой. Встроенные высокоуровневые структуры данных в сочетании с

динамическими типизацией и связыванием делают язык привлекательным для быстрой разработки приложений (RAD, Rapid Application Development). Кроме того, его можно использовать в качестве сценарного языка для связи программных компонентов. Синтаксис Python прост в изучении, в нем придается особое значение читаемости кода, а это сокращает затраты на сопровождение программных продуктов. Python поддерживает модули и пакеты, поощряя модульность и повторное использование кода. Интерпретатор Python и большая стандартная библиотека доступны бесплатно в виде исходных и исполняемых кодов для всех основных платформ и могут свободно распространяться.

Создание Python было начато Гвидо ван Россумом (Guido van Rossum) в 1991 году, когда он работал над распределенной ОС Амеба. Ему требовался расширяемый язык, который бы обеспечил поддержку системных вызовов. За основу были взяты ABC и Модула-3. В качестве названия выбрали Python..

Flask является микрофреймворком для создания вебсайтов на языке Python. В основу статьи положен перевод из официальной документации Flask. Поэтому в ней имеется обращение от первого лица, то есть от создателя фреймворка Армина Ронахера.

Flask в данном контексте переводится как пороховой рожок, на это указывает официальное лого.

Одним из проектных решений во Flask является то, что простые задачи должны быть простыми; они не должны занимать много кода, и это не должно ограничивать. Flask использует локальные треды внутри объектов, так что вы не должны передавать объекты в пределах одного запроса от функции к функции, оставаясь в безопасном треде. Хоть это и очень простой подход, который позволяет сэкономить время, такое решение может вызвать некоторые проблемы для слишком больших приложений, поскольку изменения в этих локальных тредах-объектах могут произойти где угодно в этом треде.

Во Flask многие вещи предварительно сконфигурированы, на основе общей базовой конфигурации. Например, шаблоны и статические файлы сохранены в подкаталогах в пределах исходного дерева.

Основная причина почему Flask называется «микрофреймворком» - это идея сохранить ядро простым, но расширяемым. В нем нет абстрактного уровня базы данных, нет валидации форм или всего того, что уже есть в других библиотеках. Однако, Flask поддерживает расширения, которые могут добавить необходимую функциональность и имплементирует их так, как будто они уже были встроены изначально. В настоящее время уже есть расширения: формы валидации, поддержка закачки файлов, различные технологии аутентификации и многие другие.

Flask защищает вас от наиболее распространенных и известных способов взлома, такие как XSS (cross-site scripting). Flask и шаблонизатор Jinja2 защищают, но все равно могут найтись способы взломать сайт.

JSON Web Token (JWT) - это открытый стандарт (RFC 7519) для создания токенов доступа, основанный на формате JSON. Как правило, используется для передачи данных для аутентификации в клиент-серверных приложениях. Токены создаются сервером, подписываются секретным ключом и передаются клиенту, который в дальнейшем использует данный токен для подтверждения своей личности.

В простом понимании - это строка в специальном формате, которая содержит данные, например, ID и имя зарегистрированного пользователя. Она передается при каждом запросе на сервер, когда необходимо идентифицировать и понять, кто прислал этот запрос.

После того, как посетитель прошел авторизацию в нашей системе, указав свой логин и пароль, система выдает ему 2 токена: access token и refresh токен.

После чего посетитель, когда хочет получить с сервера данные, например, свой профиль, вместе с запросом он передает Access токен, как на примере выше. Сервер, получив его проверяет, что он действительный (об этом чуть ниже), вычитывает полезные данные из него (тот же user_id) и, таким образом, может идентифицировать пользователя.

Токен разделен на три основные группы: заголовок, полезные данные и сигнатура, разделенные между собой точкой. Наглядно структура JWT заголовка показана на рисунке 3.7:

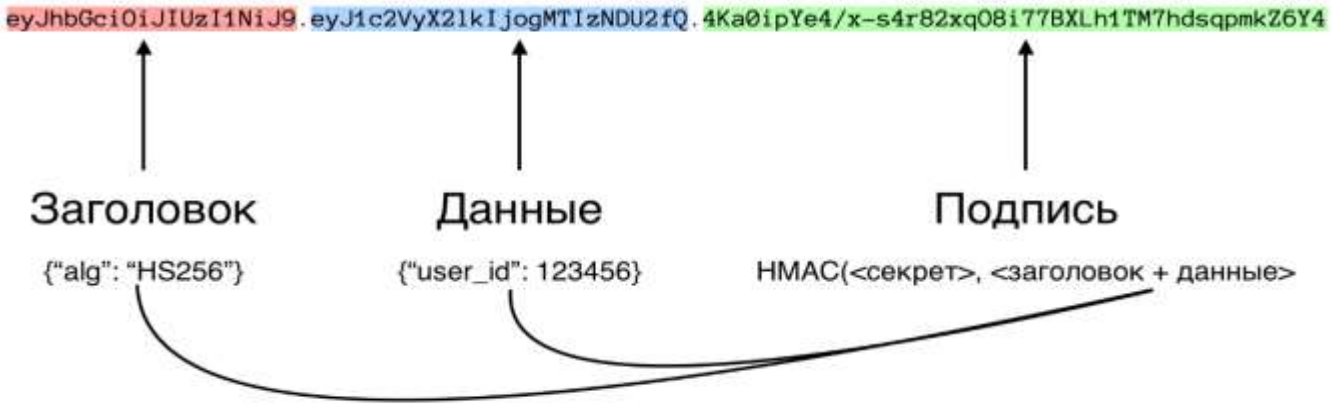

Рисунок 3.7 - Структура JWT Токена

3.3 Интерфейс пользователя приложения

Важную роль в работе программного комплекса является продуманная структура взаимосвязи программных компонентов, в которой четко прописаны потоки данных, структура ИС [13]. Так же данный раздел критически необходим при внедрении и отлавливании ошибок, ведь по структурной схеме проще всего понять ключевые моменты, основные сервисы и места где могут возникать проблемы и конфликты. На рисунке 3.8 представлена структура

одного контейнера Docker, содержащего в себе один экземпляр веб сервера NGINX, UWSGI сервера, а так же сам программный код на python, с фреймворком Flask.

Рисунок 3.8 - Структурная схема контейнера Docker

Более детально рассмотрим взаимодействие NGINX и UWSGI серверов, как один из ключевых аспектов работы микросервиса на Python. Оба сервиса друг с другом функционируют не через протокол HTTP, а через сокеты Unix и специальный протокол uwsgi. Сокеты обеспечивают двухстороннюю связь типа “точка-точка” между двумя процессами [14]. Они являются основными компонентами межсистемной и межпроцессной связи. Каждый сокет представляет собой конечную точку связи, с которой может быть совмещено некоторое имя. Он имеет определенный тип, и один процесс или несколько, связанных с ним процессов.

Сокеты находятся в областях связи (доменах). Домен сокета - это абстракция, которая определяет структуру адресации и набор протоколов. Сокеты могут соединяться только с сокетами в том же домене. Всего выделено 23 класса сокетов (см. файл <sys/socket.h>), из которых обычно используются только UNIX-сокеты и Интернет-сокеты. Сокеты могут использоваться для установки связи между процессами на отдельной системе подобно другим формам IPC.

Класс сокетов UNIX обеспечивает их адресное пространство для отдельной вычислительной системы. Сокеты области UNIX называются именами файлов UNIX. Сокеты также можно использовать, чтобы организовать связь между процессами на различных системах. Адресное пространство сокетов между связанными системами называют доменом Интернета. Коммуникации домена Интернета используют стек протоколов TCP/IP.

Протокол uwsgi - это собственный протокол, используемый сервером uWSGI. Это двоичный протокол, который может нести любой тип данных. Первые 4 байта пакета uwsgi описывают тип данных, содержащихся в пакете [15].

Каждый запрос uwsgi генерирует ответ в формате uwsgi.

Даже обработчики веб-сервера подчиняются этому правилу, поскольку HTTP-ответ является допустимым пакетом uwsgi (Необходимо посмотреть на модификатор 1 = 72).

Протокол работает в основном через TCP, но главный процесс может привязываться к одноадресной/многоадресной передаче UDP для встроенного SNMP-сервера или запросов управления кластером/обмена сообщениями.

Ведется работа по поддержке SCTP. Пример взаимодействия показан на рисунке 3.9:

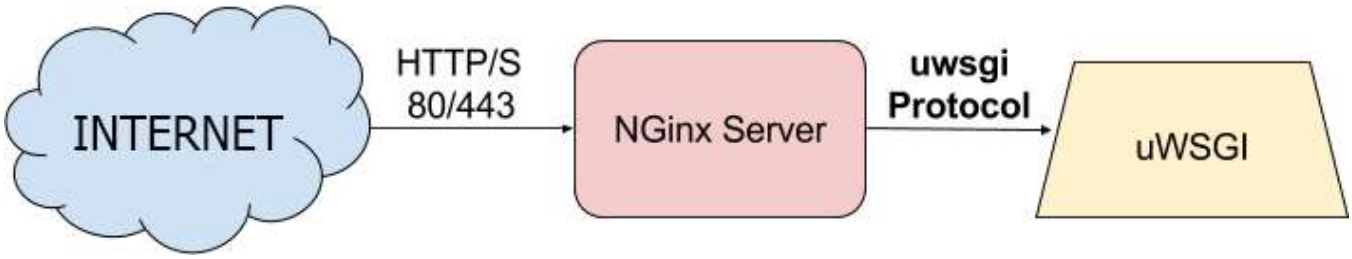

Рисунок 3.9 - Принципиальная схема взаимодействия NGINX и uWSGI

После определения термина и структуры контейнера, стоит прояснить структуру кластера контейнеров. Вся виртуализация происходит в Docker, который аппаратными средствами позволяет создавать множество операционных систем, полностью независимых друг от друга.

Так же Docker создает отдельную подсеть, в которой контейнеры могут взаимодействовать друг с другом, сеть можно сделать недоступной для основной системы или наоборот сделать контейнеры частью уже имеющейся физической сети через так называемые Bridge адаптеры, имитирующие физические устройства. Так же шлюзом сети по умолчанию будет являться сам Docker, но при необходимости можно выдать роль щлюза одному из контейнеров, либо вынести его за предел Docker. В качестве балансировщика и главного шлюза будет выполнять отдельный контейнер с Nginx, UWSGI, и Flask, которые будут работать с конечными пользователями, распределять запросы между микросервисами, ставить потоки в очередь, и обеспечивать безопасный обмен данными. Общая схема приведена на рисунке 3.10:

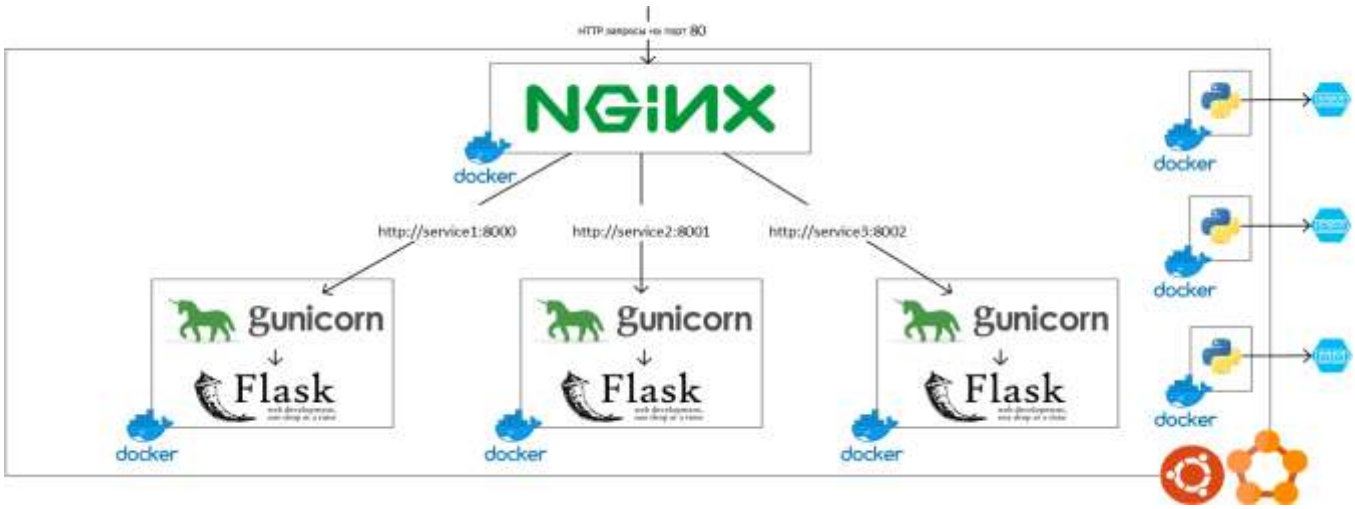

Рисунок 3.10 - Кластера контейнеров Docker Внутри системы Linux

По данным схемам можно проводить «TroubleShootig» проблем, выстраивать структуру и проводить установку программного комплекса. Либо производить обновление контейнеров.

ЗАКЛЮЧЕНИЕ

Дисертация актуальна не только для предприятия, но и для сферы информационной безопасности в целом. Поскольку угрозы никуда не уходят, а проверять себя на готовность нужно постоянно. По текущим трендам в ИБ и IT, становится видно, что машинное обучение становится все более полноправным решением на ровне, а где-то и лучше, чем традиционные методы написания программ, поскольку анализировать данные настолько быстро и точно, как ИИ, не может ни один человек, и ни одна программа, что чрезмерно важно при выявлении угроз в реальном времени.

Во время работы над дипломной работой, были выполнены следующие задачи и даны оветы:

1. Была изучена структура сканера веб уязвимостей, ее процессы и особенности. Это необходимый пункт, на котором строится все дальнейшее исследование проблем. После изучения структуры и составление схем, были выявлены закономерности, слабые места и ключевые особенности, на которых строятся требования к разработке ПК, для проверки эффетивности.

2. После аналитики проблемных мест SOC стало понятно, как можно их исправить и какими инструментами проверять, на готовность. В дальнейшем благодаря контейнеризации без проблем можно будет заменить или обновить инструментарий, на другие, более совершенные и быстрые инструменты для большей эффективности выполняемых работ.

Язык программирования Python и «Стэк» технологий полностью оправдали свой выбор в связи с легкостью разработки, сопровождения и введения в эксплуатацию. Микросервисный подход так же хорошо показал себя, как один из перспективных подходов к разработке, из-за своих плюсов в отношении надежности программного комплекса и скорости работы, а также возможности балансировки нагрузки.

СПИСОК ЛИТЕРАТУРЫ

1 Elizabeth, F. Building a Test Suite for Web Application Scanners. Elizabeth F., Romain, G., Vadim, O., Paul, B.

2 Emre, E. Web Vulnerability Scanners: A Case Study. Emre, E., Angel, R. 2017.

3 Fonseca, J., Vieira, M., & Madeira, H. (2014). Evaluation of Web Security Mechanisms using Vulnerability & Attack Injection. Dependable and Secure Computing, IEEE Transactions on, 11(5), 440–453.

4 Kinnaird, Mc. Open Source Web Vulnerability Scanners: The Cost Effective Choice? 2014 Proceedings of the Conference for Information Systems Applied Research Baltimore, Maryland USA. ISSN: 2167–1508.

5 M. Parvez, P. Zavarsky and N. Khoury»,Analysis of effectiveness of black-box web application scanners in detection of stored SQL injection and stored XSS vulnerabilities», 2015 10th International Conference for Internet Technology and Secured Transactions (ICITST), London, 2015, pp. 186–191. doi: 10.1109/ICITST.2015.7412085

6 M. Parvez, P. Zavarsky and N. Khoury, «Analysis of effectiveness of black-box web application scanners in detection of stored SQL injection and stored XSS vulnerabilities», 2015 10th International Conference for Internet Technology and Secured Transactions (ICITST), London, 2015, pp. 186–191. doi: 10.1109/ICITST.2015.7412085

7 Pallavi Deshmane. Web Vulnerability Scanner Application. Pallavi Deshmane, Shweta Singh, Nakshi Doshi, Harshit Punatar, Shashank Gangar. Imperical Journal of Interdisciplinary Research, Vol 3, Issue-2, 2017. ISSN: 2454–1362.

8 Rik A. J. Web Application Vulnerability Testing with Nessus. The OWASP Foundation. Stefan, K. SecuBat: A Web Vulnerability Scanner.

9 Stefan, K., Engin, K., Christopher, K. Nenad, J.

10 The government of the Hong Kong Special Administrative Region. An Overview of Vulnerability Scanners. 2/2018.