

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«МИРЭА – Российский технологический университет» РТУ МИРЭА**

ИКБ направление «Киберразведка и противодействие угрозам с применением технологий искусственного интеллекта» 10.04.01

Кафедра КБ-4 «Интеллектуальные системы информационной безопасности»

Отчёт по практической работе №5

по дисциплине: «Система для сбора событий и логов»

Группа: ББМО-02-22

Выполнил: Давыдов И.Д.

Проверил: Козачок А.В.

Москва, 2024

Содержание

[Введение 3](#_Toc157696782)

[1 Настроим тестовый стенд 4](#_Toc157696783)

[2 Развернутые СЗИ: 4](#_Toc157696784)

[2.1 SIEM: Установка сервера Wazuh 4](#_Toc157696785)

[2.2 IDS: Установка Snort 15](#_Toc157696786)

[2.3 ThreatHuntingTool: Установка YARA 21](#_Toc157696787)

[2.4 Scanner: Установка OpenVAS 27](#_Toc157696788)

[Заключение 33](#_Toc157696789)

# Введение

**Цель работы**: создать и настроить виртуальную среду для обнаружения уязвимостей и угроз.

**Задачи** практической работы:

1. Создать внутреннюю сеть виртуальных машин (ВМ), чтобы имитировать потенциальные уязвимости и атаки.
2. Установить и настроить систему мониторинга.
3. Развернуть уязвимые сервисы/сборки ОС в составе стенда.
4. Создать профили источников данных: Определить, какие источники данных используются для мониторинга сети.
5. Создать правила обнаружения: настроить правила обнаружения для источников данных, чтобы автоматически обнаруживать потенциальные угрозы безопасности.
6. Настроить инструменты для мониторинга и обнаружения угроз безопасности, а также анализа данных.
7. Запустить сканеры уязвимостей.
8. Создать «Локальные» угрозы безопасности.
9. Проанализировать потенциальные угрозы: Использовать логи и инструменты мониторинга для выявления потенциальных угроз безопасности.

# Настроим тестовый стенд

Тестовый стенд из виртуальных машин: виртуальная машина на безе Ubuntu, является узлом Wazuh. Виртуальная машина на базе Kali, является узлом OpenVAS.

# Развернутые СЗИ:

# SIEM: Установка сервера Wazuh

Будем использовать Wazuh.

ВМ на рисунке 2.1.1.

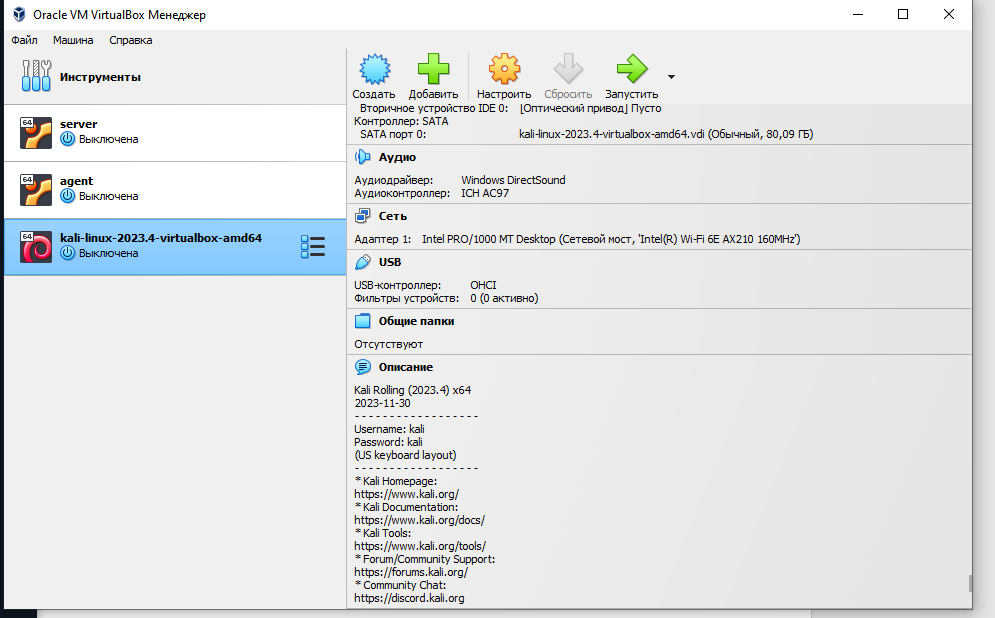


Рисунок 2.1.1 – Созданные виртуальные машины Убедимся, что ВМ пингуются между собой (рис. 2.1.2):

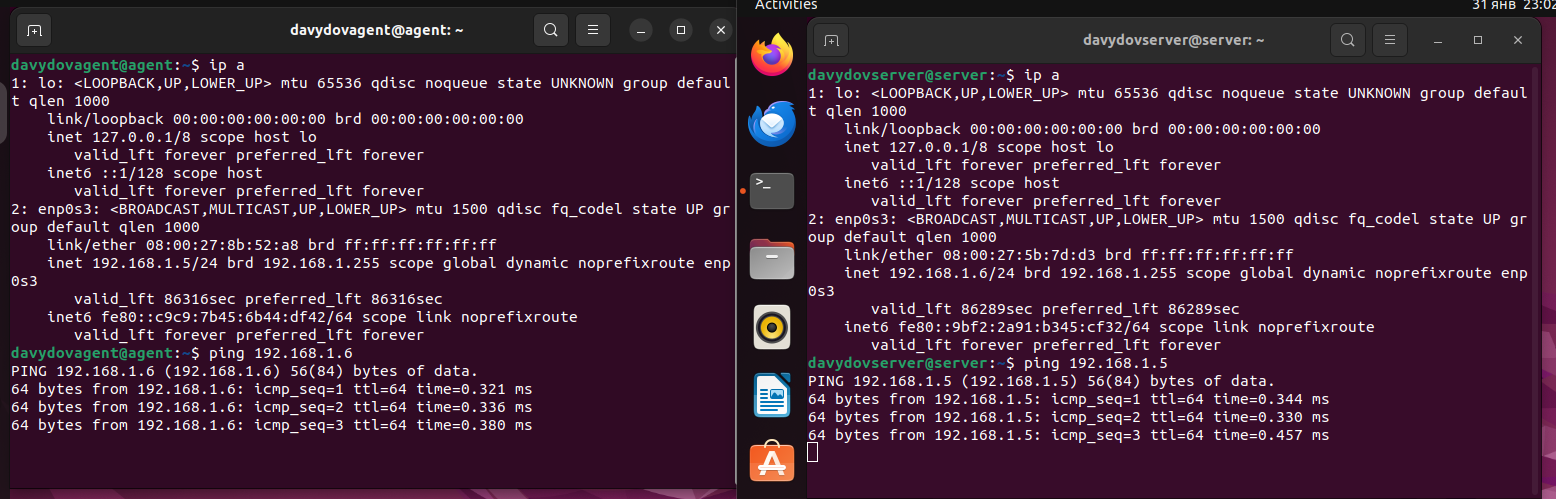


Рисунок 2.1.2 – Проверка сетевого обмена

Напомним, как всё это устанавливалось.

**Установка сервера Wazuh (рис. 2.1.3).**

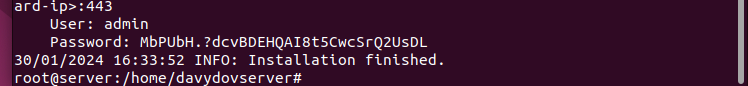
=

Рисунок 3.1 – Установка сервера Wazuh, видно Логин(admin) и Пароль(MbPUbH.?dcvBDEHQAI8t5CwcSrQ2UsDL)

**Установка агента Wazuh (рис. 2.1.4)**

Рисунок 2.1.4 – Установка агента Wazuh

**Настроим проверку целостности файлов.**

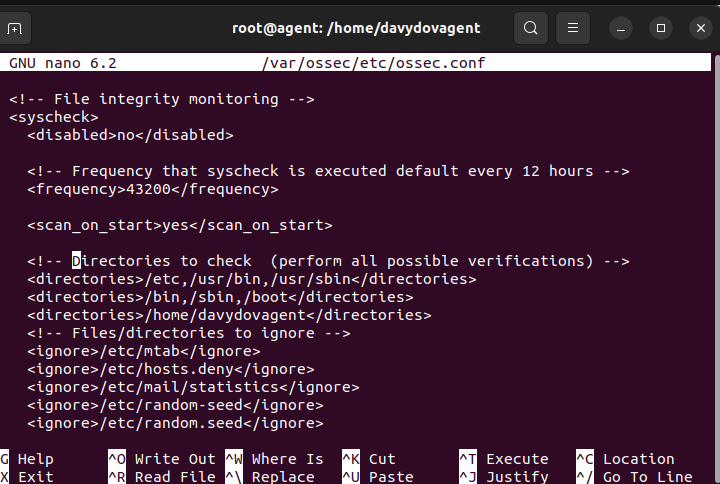
Отредактируем конфигурацию агента /var/ossec/etc/ossec.conf добавив в список проверяемых папок /home/davydovagent (рис. 2.1.5-2.1.6).

Рисунок 2.1.5 – Добавление директорий и файлов к мониторингу файловой системы

Видим добавленные файлы из папки /home (рис. 2.1.7).

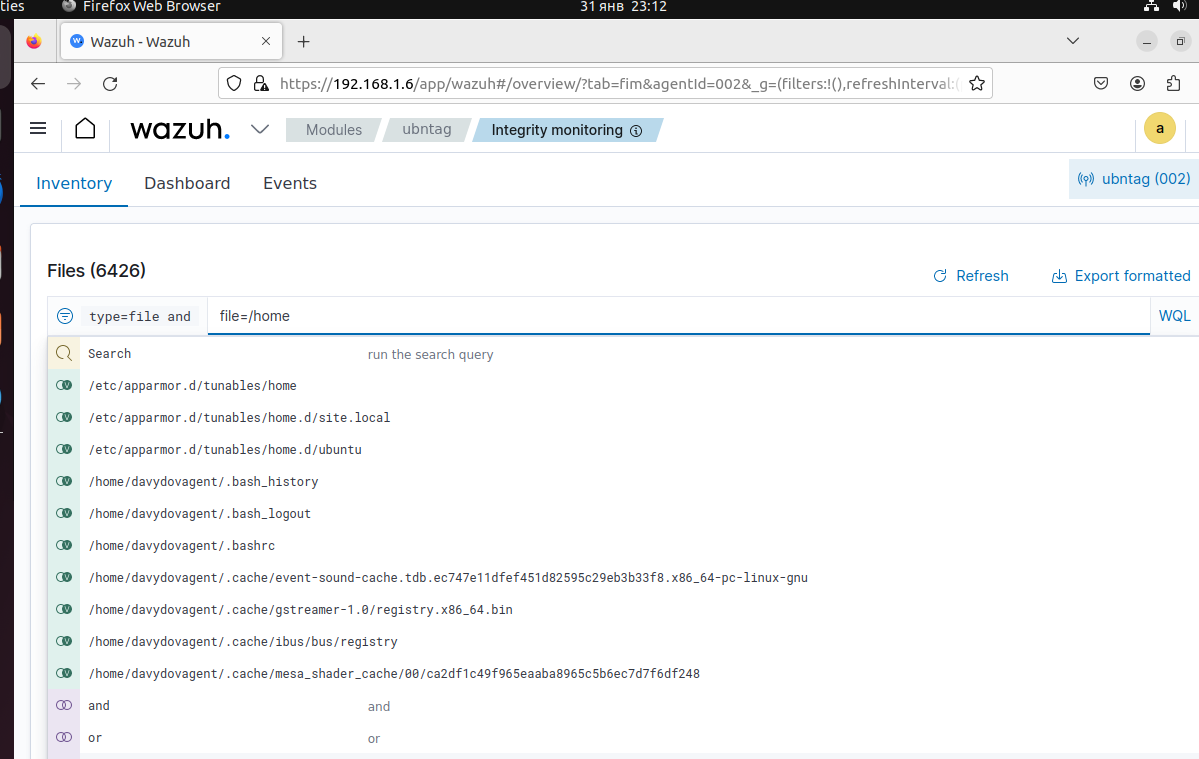


Рисунок 2.1.7 – Добавленная директория

**Настройка выявления уязвимостей.**

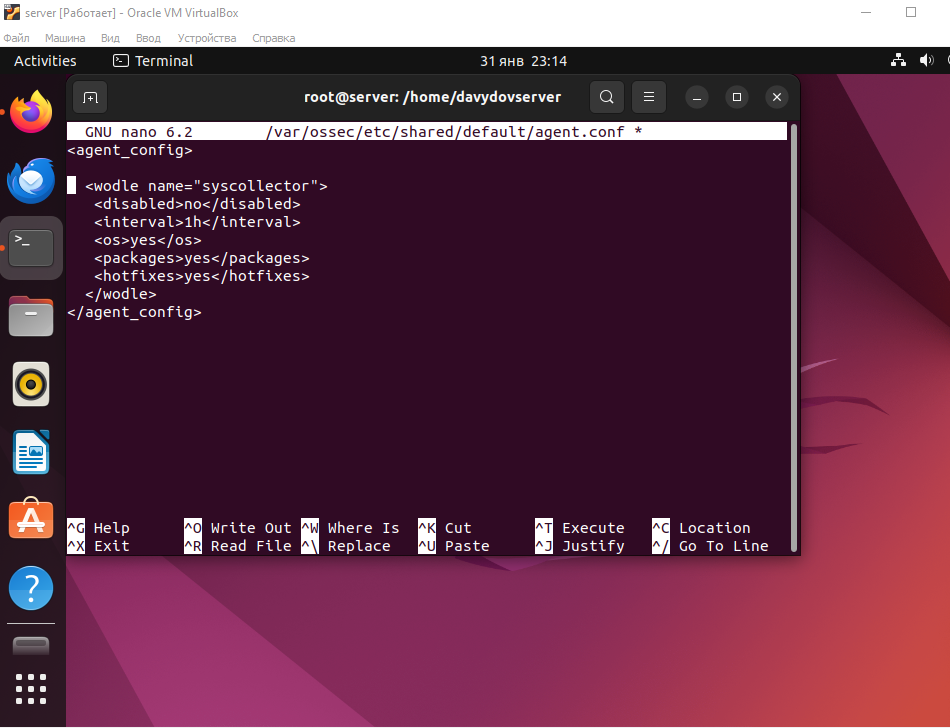
Сконфигурируем общего агента на сервере. Добавление раздела, отвечающего за syscollector в конфигурационный файл агента или сервера представлен на рисунке 2.1.8

Рисунок 2.1.8 – Подключаем syscollector

Активация модуля детектора уязвимостей и указание операционных систем агентов в конфигурационном файле сервера. Включим обнаружение уязвимостей для Ubuntu (рис. 2.1.9).

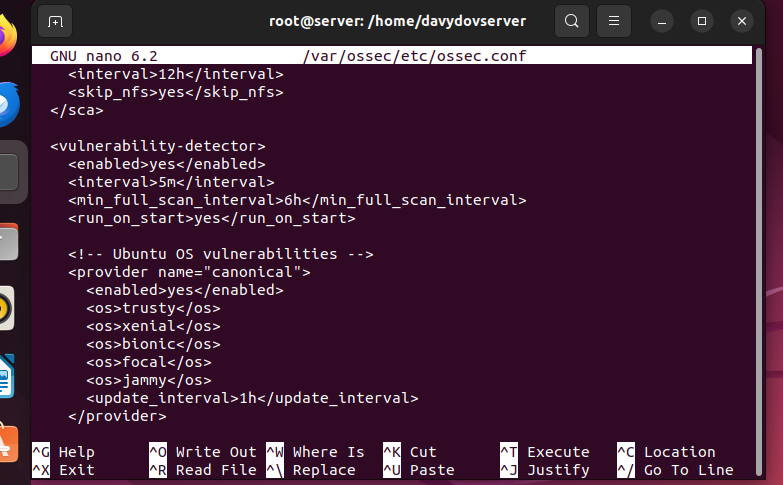


Рисунок 2.1.9 – Активация детектора и подключение ОС агентов Перезапуск сервиса агента и/или сервера для применения изменений (рис.

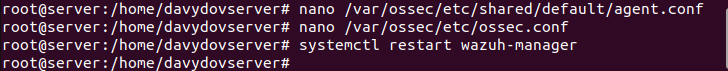
2.1.10).

Рисунок 2.1.10 – Перезапуск сервиса сервера

Видим список уязвимостей (рис. 2.1.11).

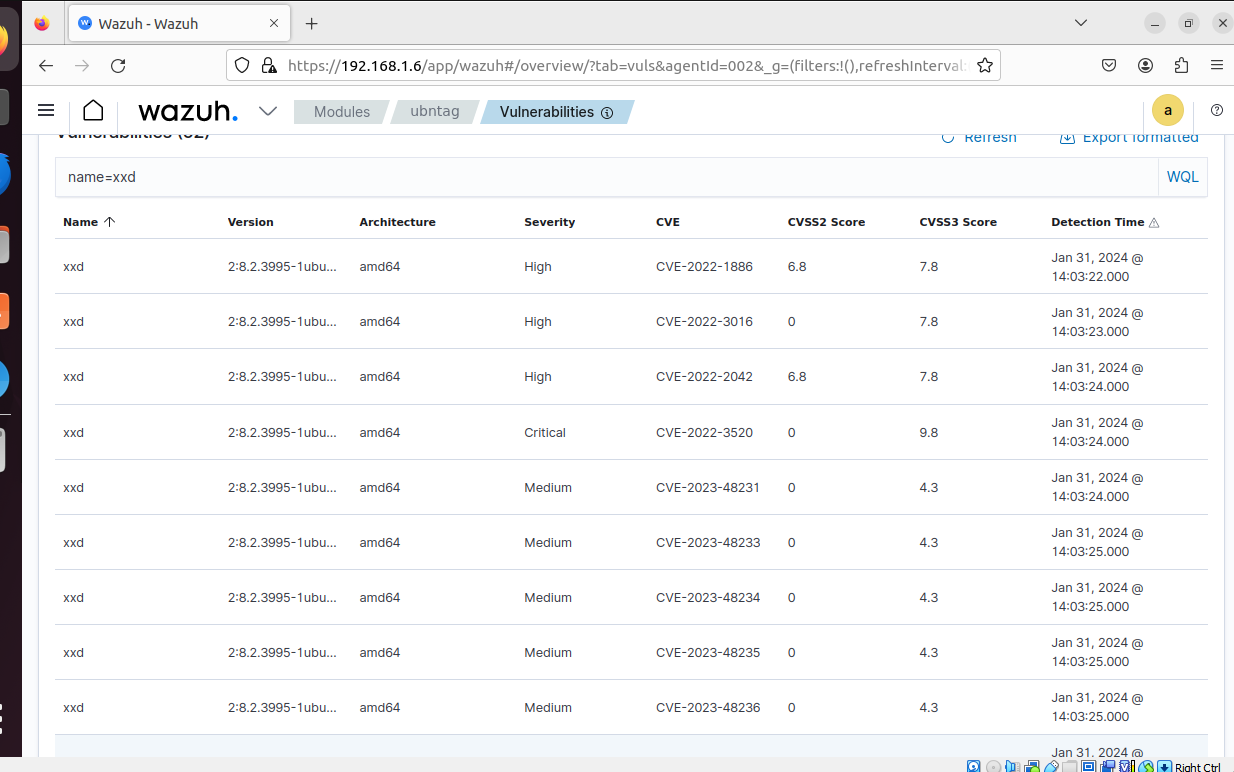


Рисунок 2.1.11 – События детектора уязвимостей

**Настройка выявления скрытых процессов.**

Установим rootkit.

Изменим частоту проверок агента до 60 секунд с целью уменьшения таймаута срабатывания рутчекера (рис. 2.1.12).

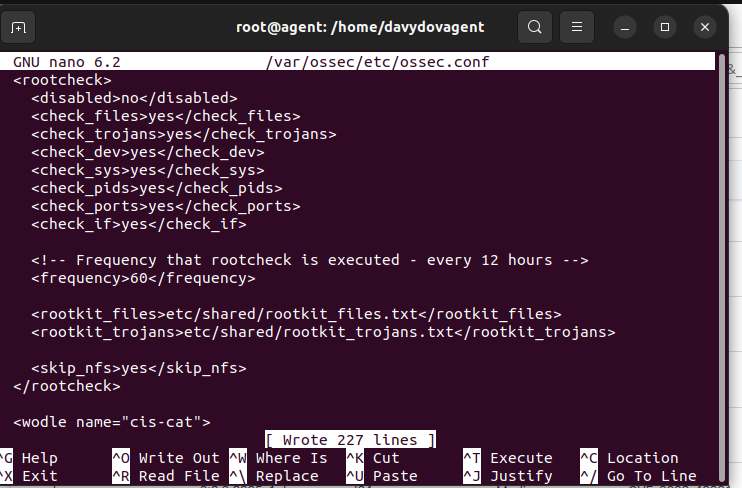


Рисунок 2.1.12 – Настройка частоты срабатывая рутчекера Перезапуск сервиса агента для применения изменений (рисунок 2.1.13).

Рисунок 2.1.13 – Перезапуск агента Эмуляция атаки представляет из себя следующий алгоритм:

* Клонирование репозитория с руткитом (рис. 2.1.14);
* Компиляция руткита;
* Загрузка руткита в ядро;
* Отправка сигнала KILL -63 со случайным PID для обнаружения руткита. Имитация атаки (рис. 2.1.15): 

Рисунок 2.1.14 – Клонирование исходного кода руткита

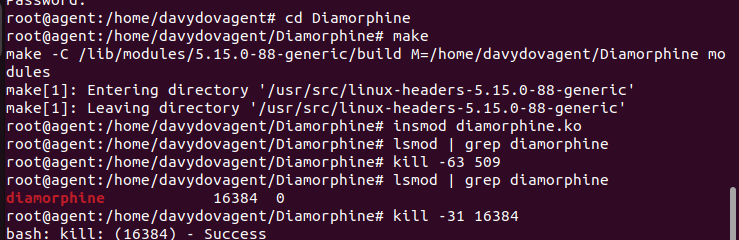




Рисунок 2.1.15 – Компиляция, загрузка руткита, который создает скрытый процесс, скрытие процесса

Скроем процесс rsyslogd с помощью руткита. Для этого нужно отправить сигнал KILL -31 с PID rsyslogd (рис. 2.1.16 (команды видно в 2.1.5)).

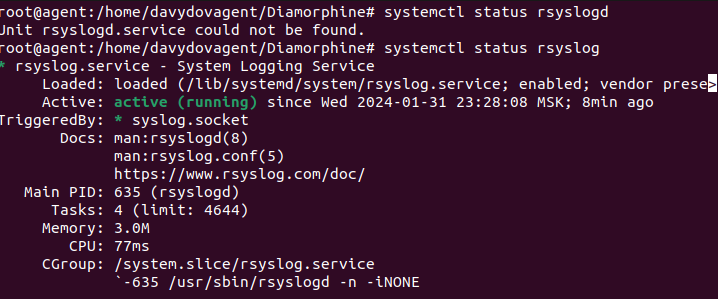


Рисунок 2.1.16 – Сокрытие процесса rsyslogd Посмотрим результат (рис. 2.1.17).

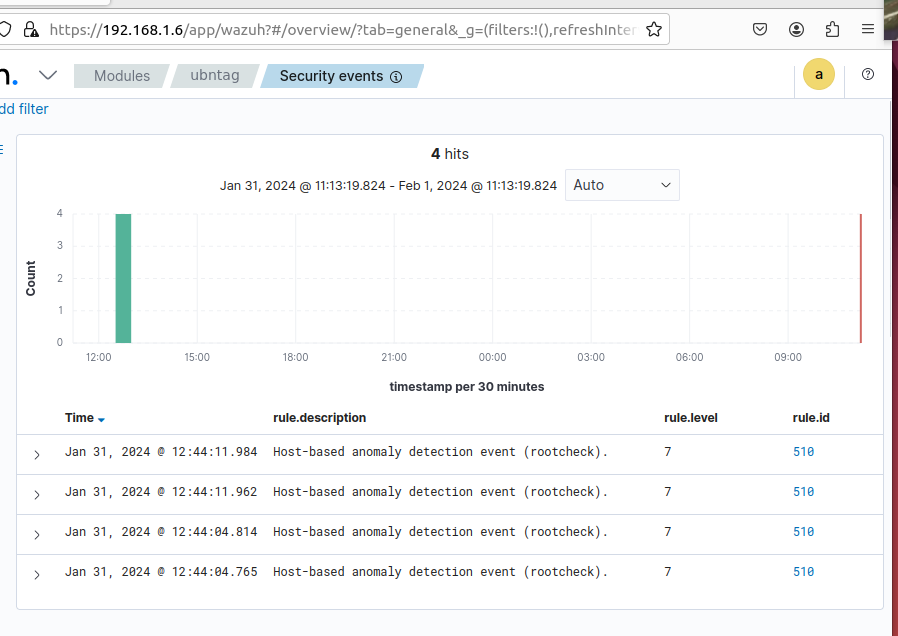


Рисунок 2.1.17 – Результат скрытого процесса

**Настройка выявления SQL-инъекций.**

Настройка выявления SQL-инъекций: Для этого установим Apache и настроим агент. Установка веб-сервера (рис. 2.1.18).

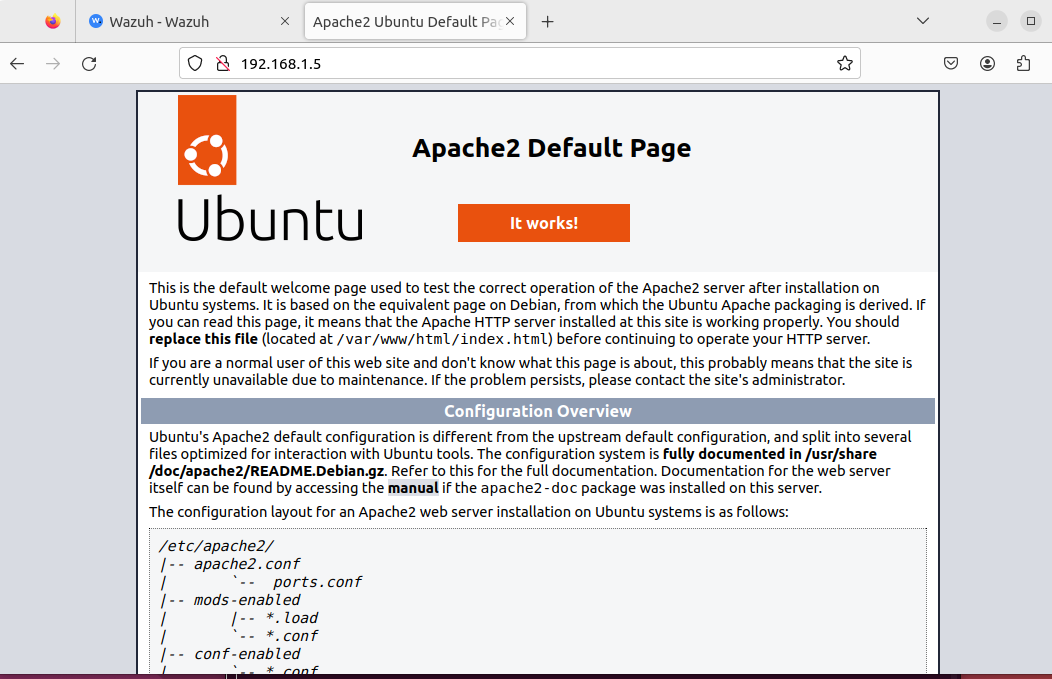


Рисунок 2.1.18 – Приветственное окно веб-сервера

Настройка конфигурационного файла агента для подключения веб- сервера к мониторингу. Добавим отслеживание apache (рис. 2.1.19).

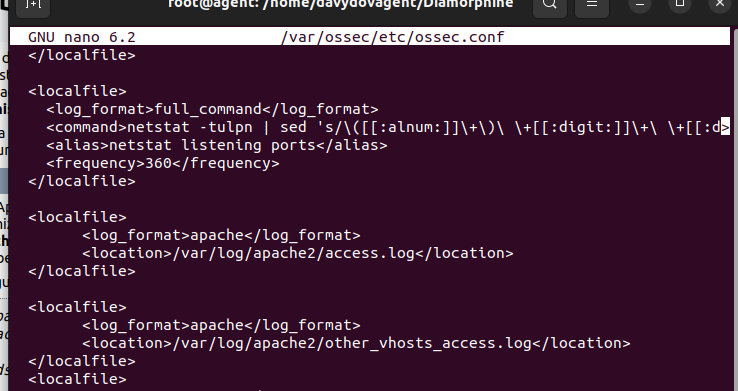


Рисунок 2.1.19 – Настройка конфига агента

Список команд: sudo apt install apache2 sudo ufw app list

sudo ufw allow 'Apache' sudo systemctl status apache2 curl http://<UBUNTU\_IP>

Сэмулируем атаку. Для этого выполним команду:

curl -XGET "http://192.168.1.5/users/?id=SELECT+\*+FROM+users"; Посмотрим на sql иньекцию (рис. 2.1.20).

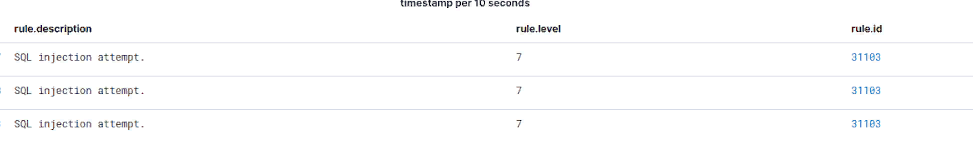


Рисунок 2.1.20 – Результат

# IDS: Установка Snort

Snort — популярный выбор для запуска системы обнаружения сетевых вторжений или сокращенно NIDS. Он отслеживает данные пакета, отправленные и полученные через определенный сетевой интерфейс. Используем команд: apt-get install snort –y.

**Версия snort (рис. 2.2.1):**

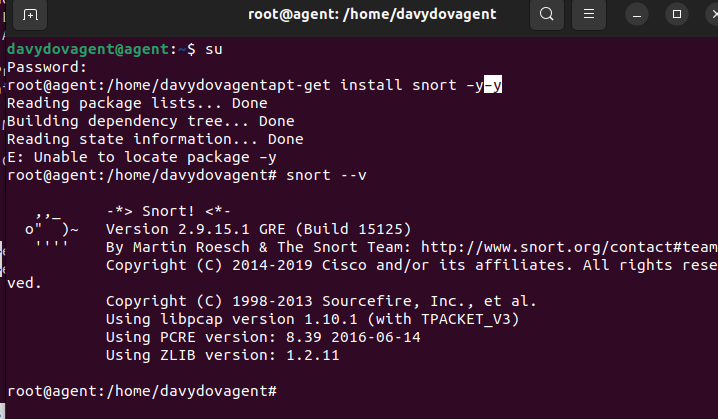


Рисунок 2.2.1 – Версия Snort

В процессе установки необходимо будет указать адрес сканируемой сети. Ее можно всегда поменять в конфигурационном файле /etc/snort/snort.conf (рисунок 3.2). Командой nano /etc/snort/snort.conf (рис. 2.2.3).

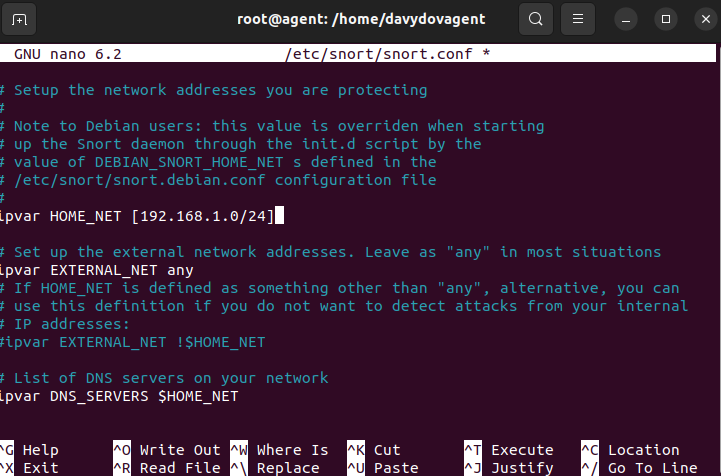


Рисунок 2.2.3 – Указываем адрес сети

**Загрузим базу правил и скопируем правила для Snort (рисунок 2.2.4).**



Рисунок 2.2.4 – Загрузка правил Snort

Для тестирования работы Snort создадим правило и запустим Snort.

Протестируем работоспособность Snort. Для этого создадим правило «alert icmp any any -> any any (msg:"ICMP connection attempt:"; sid:1000010; rev:1;)» в файле-/etc/snort/rules/local.rules (рис.2.2.5).

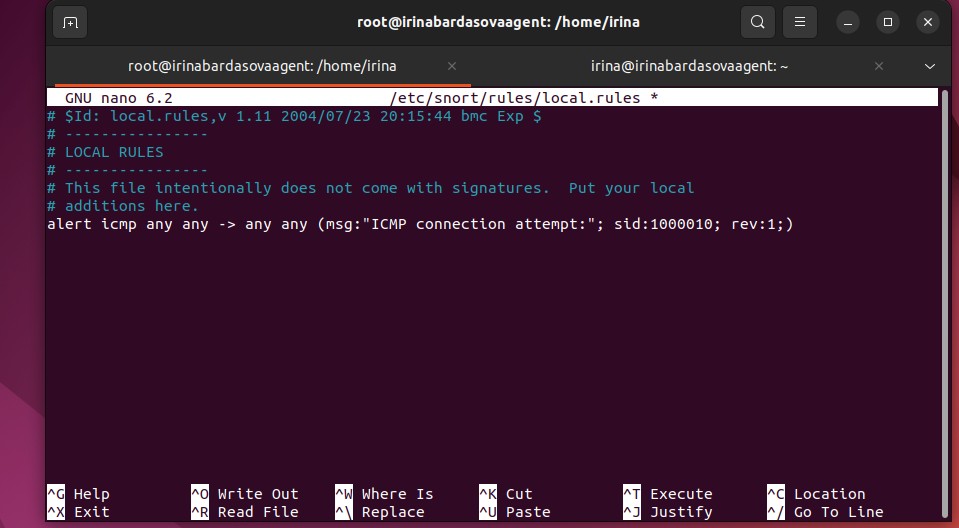


Рисунок 2.2.5 – Правило для Snort

Запустим Snort командой «sudo snort -q -A console -c

/etc/snort/rules/local.rules» (рис. 2.2.6).

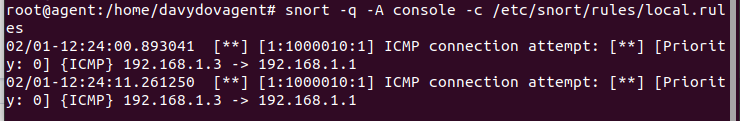


Рисунок 2.2.6 – Тестирование работоспособности Snort

**Подключим Snort к Wazuh.**

Подключим лог-файл с алертами Snort к Wazuh и перезагрузим сервер Wazuh (рис. 2.2.7).

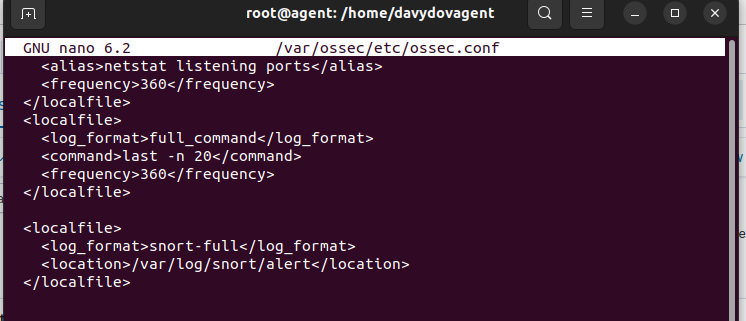
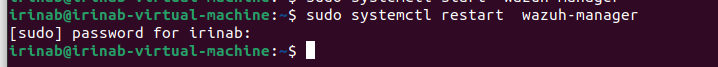


Рисунок 2.2.7 – Подключение Snort к Wazuh

Перезапустим сервисы (рис. 2.2.8).

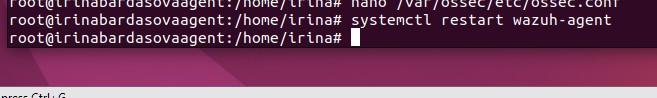


Рисунок 2.2.8 – Перезапуск сервиса агента и сервера

Укажем в конфигурационном файле snort, что подключать правила будем только из файла snort2.conf. Остальные файлы закомментируем (рис. 2.2.9).

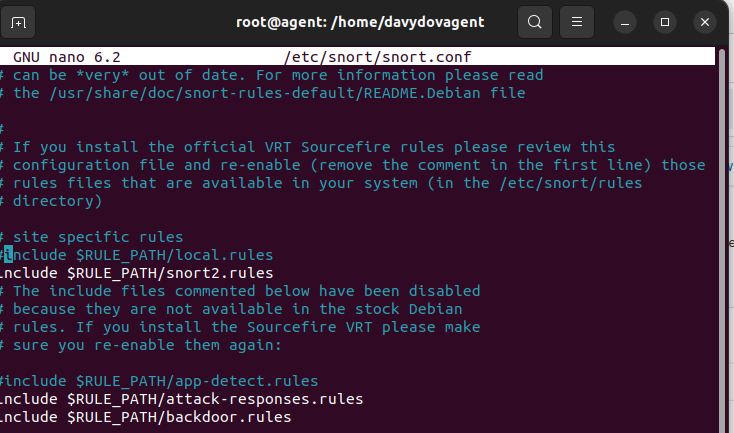


Рисунок 2.2.9 – Конфигурационный файл Проверка наличия данных в Wazuh по Snort (рис. 2.2.10).

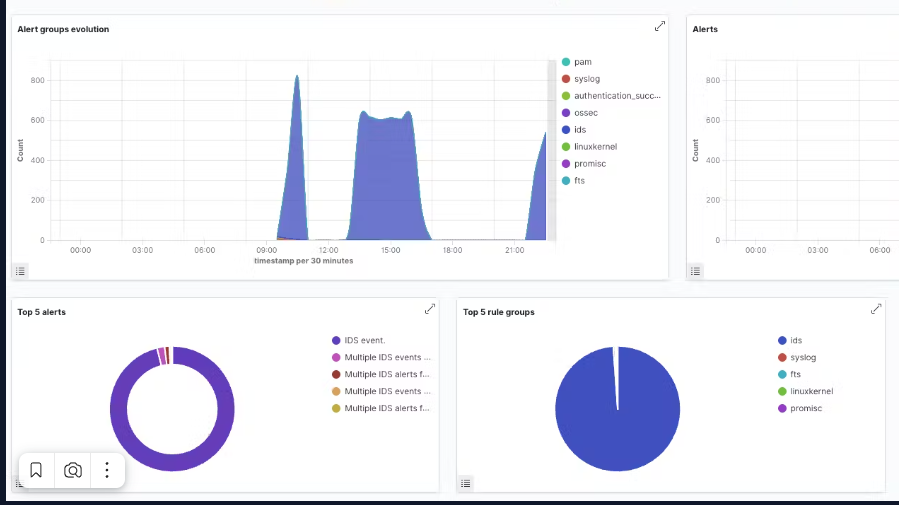


Рисунок 2.2.10 – Проверка наличия Snort в Wazuh

# ThreatHuntingTool: Установка YARA

Установим инструмент YARA. Используем команду apt install yara –y (рис.2.3.1). 

Рисунок 2.3.1 – Установка YARA

**Подключим YARA к Wazuh.**

Рассмотрим схему подключения YARA к Wazuh (рис. 2.3.2).



Рисунок 2.3.2 – Схема подключения YARA к Wazuh

Во-первых, нам нужно сообщить менеджеру, какое действие мы хотим выполнить и при каких обстоятельствах оно должно быть запущено. Для этого отредактируем файл конфигурации /var/ossec/etc/ossec.conf, добавив в конец файла следующий набор параметров:

Отредактируем файл конфигурации (рис. 2.3.3).

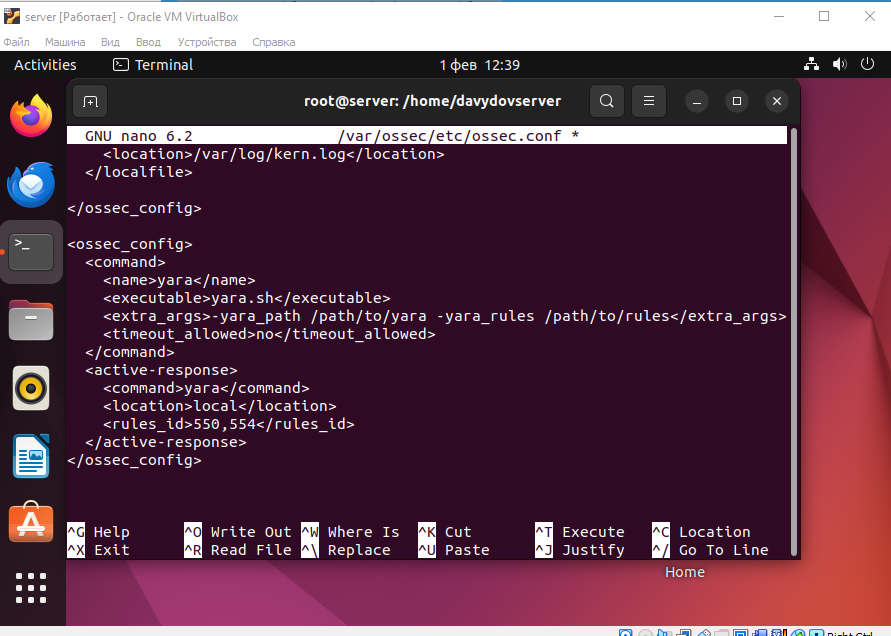


Рисунок 2.3.3 – Конфигурация файла ossec.conf сервера

Теперь нам нужно определить набор правил и декодеров для запуска оповещений о событиях, генерируемых активным ответом YARA.

Создадим файл декодера /var/ossec/etc/decoders/yara\_decoders.xml (рисунок 2.3.4). Аналогичным образом создадим файл правил

/var/ossec/etc/rules/yara\_rules.xml (рисунок 2.3.5).

Определим набор правили декодеров для запуска оповещений о событиях, генерируемых активным ответом YARA.

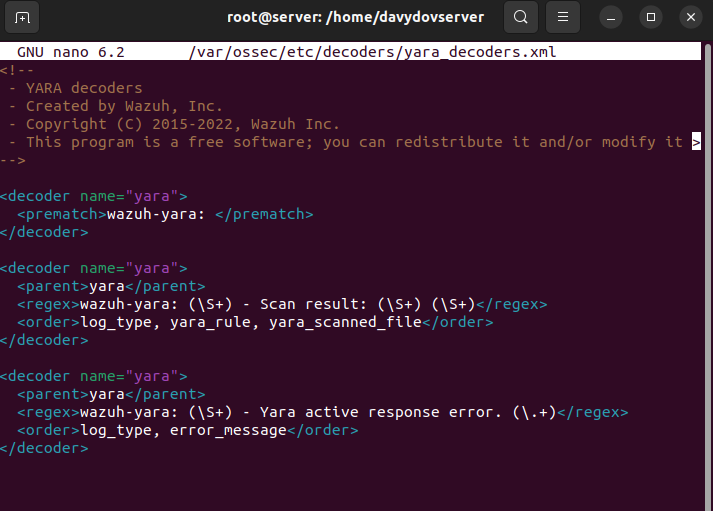


Рисунок 2.3.4 – Изменение файла yara\_decoders.xml

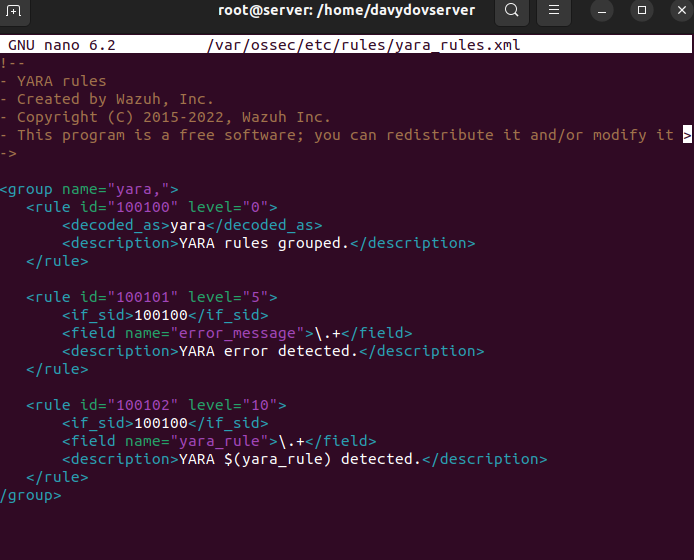


Рисунок 2.3.5 – Изменение файла yara\_rules.xml

Создадим директорию для yara и загрузим в нее правила (рис. 2.3.6).



Рисунок 2.3.6 – Подготовка директории yara

Создадим скрипт, настроенный для запуска как часть настроек активного ответа, определенных в менеджере Wazuh (рис. 2.3.7).

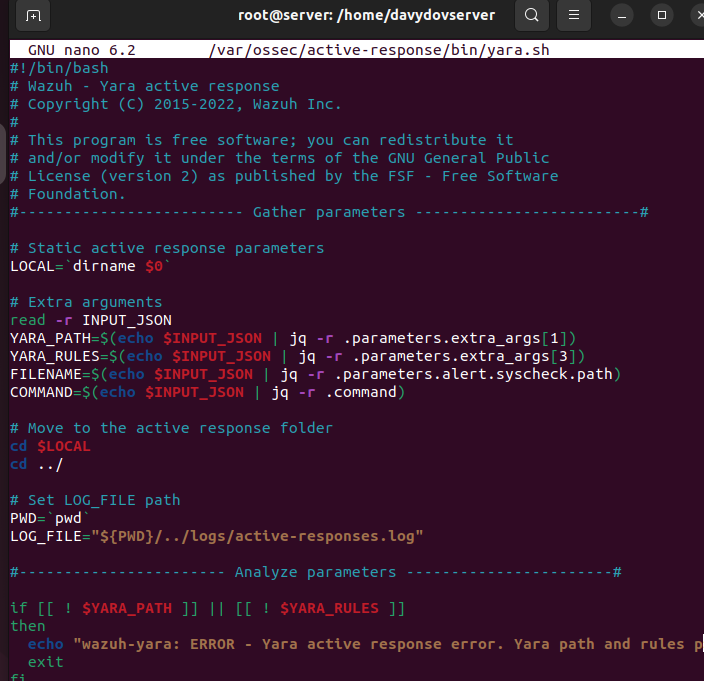


Рисунок 2.3.7 – Созданный скрипт Отредактируем права доступа к скрипту (рис. 2.3.8).

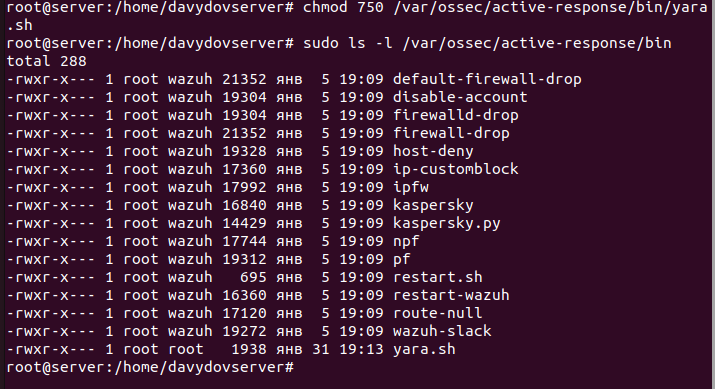
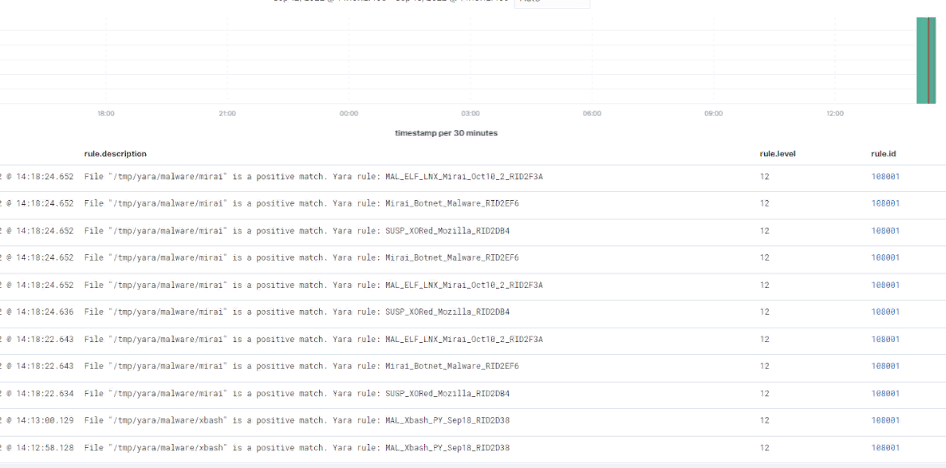


Рисунок 2.3.8 – Права доступа yara.sh Проверим наличие данных в Wazuh по YARA (рис. 2.3.9).

Рисунок 2.3.9 – Данные YARA в Wazuh

# Scanner: Установка OpenVAS

Установим сервер OpenVAS с помощью команды «sudo apt update && sudo apt upgrade -y && sudo apt install -y openvas» (рис. 2.4.1).

Рисунок 2.4.1 – Обновления и установка

OpenVAS поставляется с собственной службой Redis для Kali GNU/Linux. Убедимся, что сервис Redis настроен для корректной работы с OpenVAS командой

«systemctl status [redisserver@openvas.service»](mailto:redisserver@openvas.service) (рис. 2.4.2).

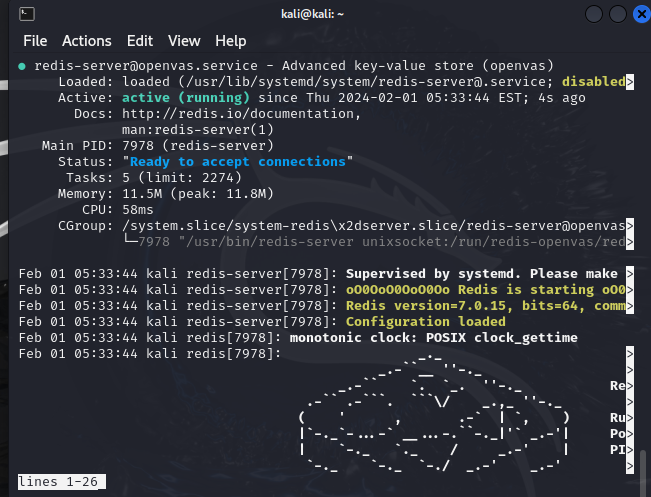


Рисунок 2.4.2 – Проверка службы redis

Настройка сервиса.

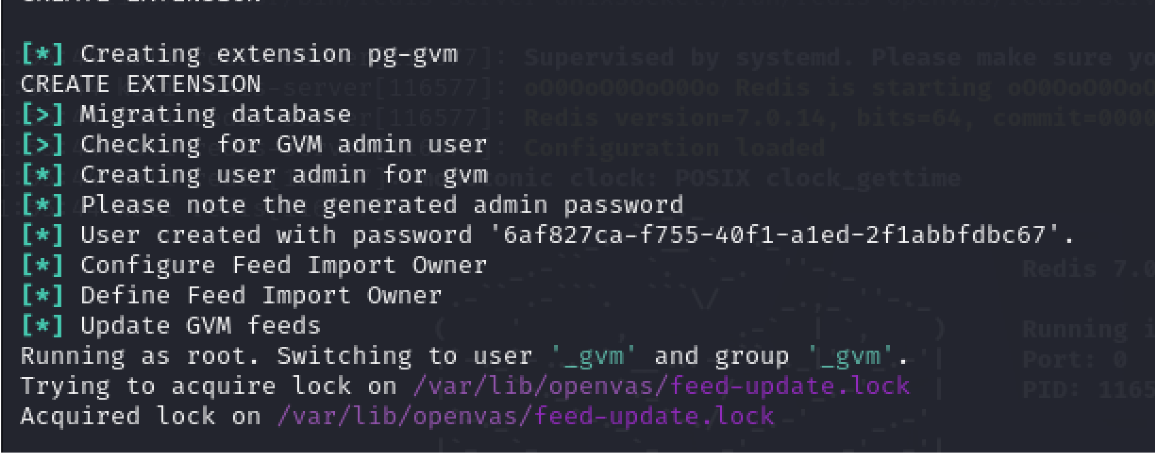


Рисунок 2.4.3 – Пароль

Запуск greenbone(рисунок 2.4.4).

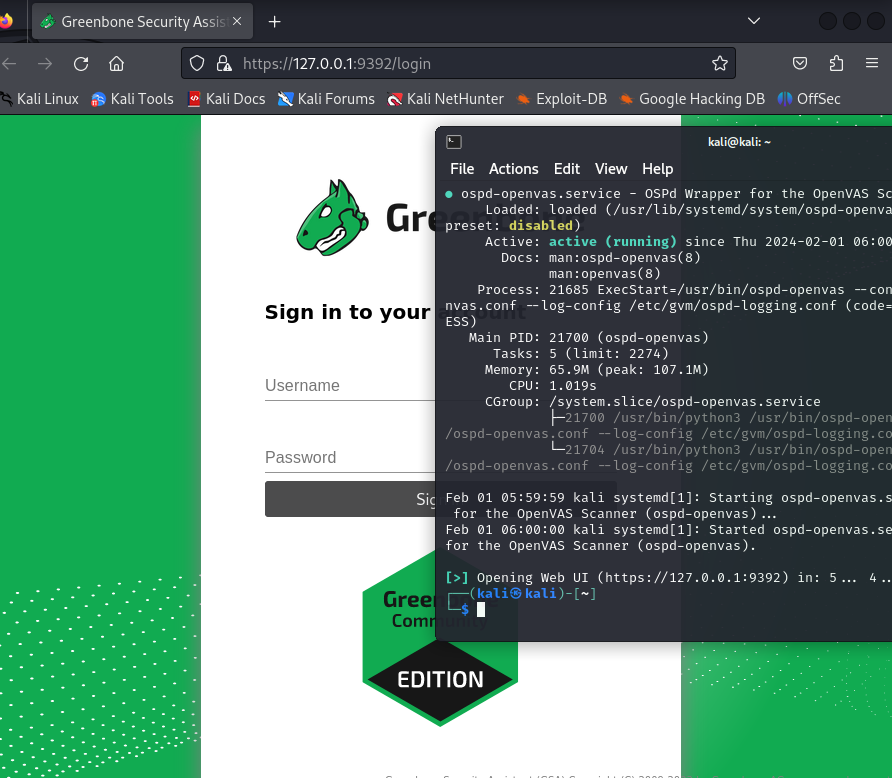


Рисунок 2.4.4 – запуск

Переходим в веб-интерфейс по адресу https://127.0.0.1:9392/ Вводим учетные данные пользователя admin и сразу попадаем на главную страницу (рис.2.4.5-2.4.6)

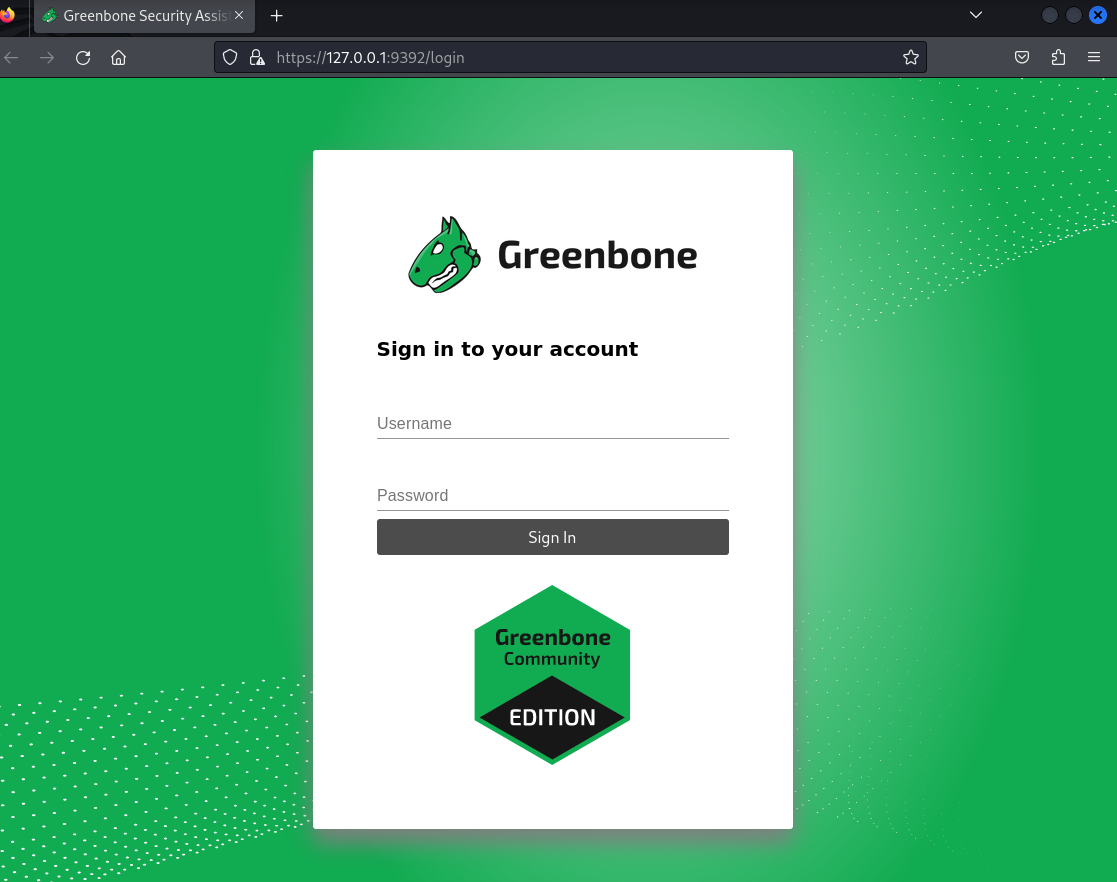


Рисунок 2.4.5 – Ввод учётной записи OpenVAS

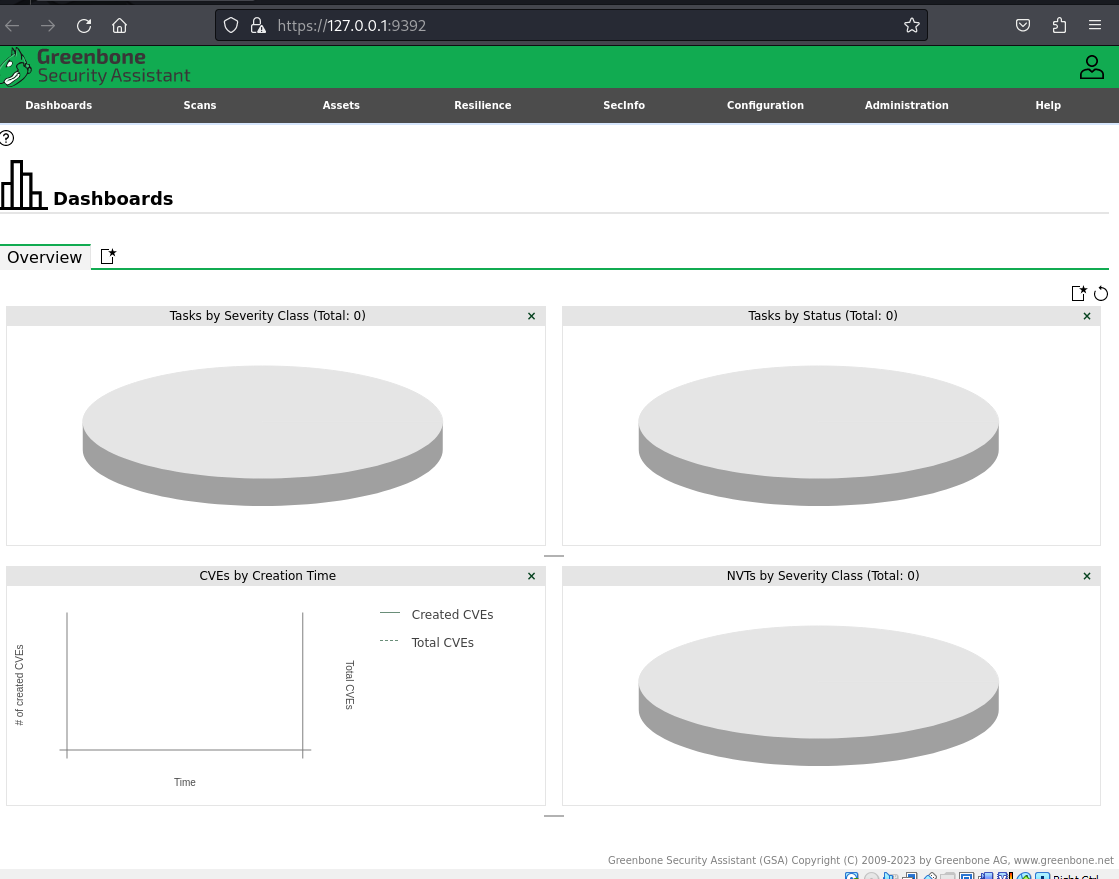


Рисунок 2.4.6 – Главная страница веб-интерфейса OpenVAS

**Сканирование узла.**

Во вкладке Scans-Tasks выберем Task Wizard. Укажем IP-адрес хоста, который хотим проверить. Запустим сканирование (рис. 2.4.7-2.4.8).

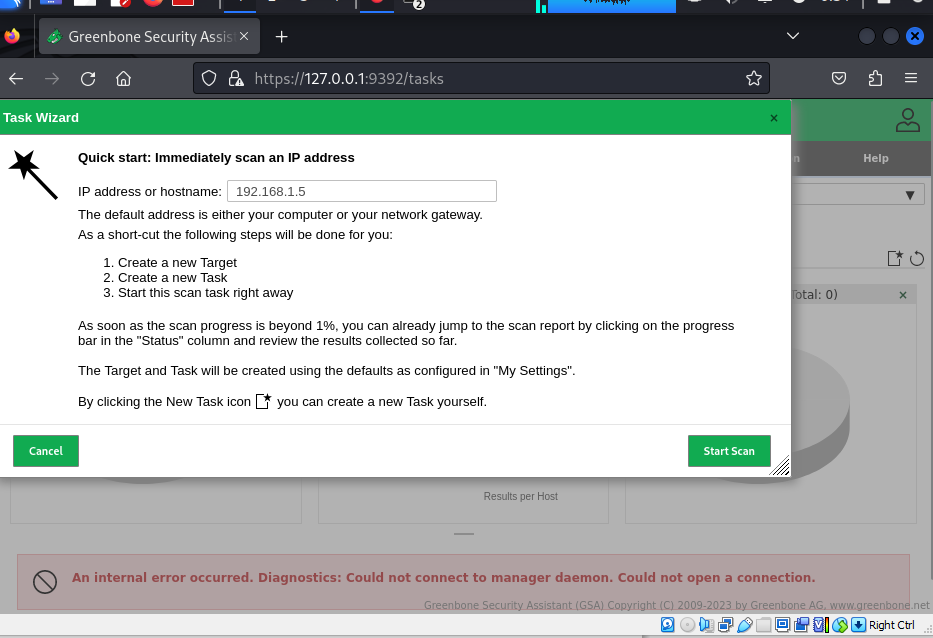
Рисунок 2.4.7 – Создание сканирования



Рисунок 2.4.8 – Результат сканирования

# Заключение

Были получены полезные знания и навыки мониторинга. Цель практической работы была достигнута.

В процессе выполнения данной практической работы были успешно реализованы следующие задачи:

1. Создали внутреннюю сеть виртуальных машин (VM), чтобы имитировать потенциальные уязвимости и атаки.
2. Установили и настроили систему мониторинга.
3. Развернули уязвимые сервисы/сборки ОС в составе стенда.
4. Создали профили источников данных: Определили, какие источники данных используются для мониторинга сети.
5. Создали правила обнаружения: настроили правила обнаружения для источников данных, чтобы автоматически обнаруживать потенциальные угрозы безопасности.
6. Настроили инструменты для мониторинга и обнаружения угроз безопасности, а также анализа данных.
7. Запустили сканеры уязвимостей.
8. Создали «Локальные» угрозы безопасности.
9. Проанализировали потенциальные угрозы.