Decluttering the Toolset – Labs and Exercises

In this document, we will delve into a range of open-source and cost-effective technologies and resources to help you curate your cybersecurity architecture toolkit. By carefully evaluating and filtering through available solutions, you can identify the most suitable tools for your specific needs. To enhance your understanding of these tools and their applications, we will provide step-by-step labs and recommended exercises that focus on assessing vulnerabilities and risk profiles. By gaining a deep understanding of potential threats, you can then select and implement appropriate defenses accordingly.

With a well-equipped toolkit at your disposal, you will be able to swiftly and effectively respond to any adversary that emerges, proactively seizing opportunities to fortify your defenses before problems arise. By meticulously preparing the ideal set of tools in advance, you can ensure a strong, resilient cybersecurity posture and confidently tackle any challenges that come your way. This document aims to guide you through the process of assembling a comprehensive and adaptable cybersecurity toolkit, empowering you to safeguard your systems and data with unwavering vigilance.

The document covers the following topics:

- Lab 1: Microsoft Threat Modeling Tool
- Lab 2: OWASP Threat Dragon
- Lab 3: Intrusion detection/prevention systems using Snort
- Lab 4: Firewall configuration using OPNsense
- Lab 5: SIEM solution using Graylog
- Lab 6: Antivirus software implementation using ClamAV
- Lab 7: Endpoint detection and response using Wazuh
- Exercise 1: Setting up and configuring Keycloak for IAM
- Lab 8: Data encryption with VeraCrypt

- Lab 9: Vulnerability scanning with OpenVAS
- Lab 10: Security configuration management using Ansible
- Lab 11: Patch management with WSUS
- Lab 12: Digital forensics with The Sleuth Kit and Autopsy
- Lab 13: Incident response with Security Onion
- Exercise 2: Static application security testing with SonarQube
- Lab 14: Dynamic application security testing with OWASP ZAP
- Lab 15: Setting up and securing an AWS environment
- Lab 16: Implementing and configuring a GRC tool
- Lab 17: Penetration testing with Kali Linux and Metasploit
- Lab 18: Security automation with StackStorm

What is in the toolbox?

Selecting the right tools is fundamental to building an effective cybersecurity architecture. With the overwhelming array of solutions on the market, architects must thoughtfully curate a toolkit tailored to their organization's specific risks, constraints, and use cases.

As noted in the book, this document is to provide insight to the various types of technology you may need to utilize or experience within an enterprise environment. With this in mind, the technologies discussed within this document are open source or free tools that will help you learn the concepts that can be applied to technologies that you may find within the enterprise. As discussed within the book, understanding the concepts allows you to be a cybersecurity architect that is able to pivot quickly and hit the ground running in any situation.

In addition, even though the book and these labs are meant to provide access to those new to cybersecurity and technology, it does make assumptions that you know how to install Linux and Windows operating systems. It is for this reason that these basic concepts are not covered within the labs, because that would make these labs even

longer. With that in mind, I will be placing basic installation guides on my website, www.secdoc.tech, so that if you do not know how to install an operating system, you will not be excluded from continuing on with these labs.

Threat modeling and risk assessment tools

Let us look at the labs.

Lab 1: Microsoft Threat Modeling Tool (TMT)

Before diving into the hands-on lab exercise, it is important to understand the value of threat modeling and tools like Microsoft TMT that support the process. As discussed, threat modeling provides a structured system for architects to methodically assess risks and weaknesses. Hands-on practice is key to skill building for threat modeling and attack simulations. This lab will guide you through installing Microsoft TMT, creating a sample model of a 3-tier web application, executing automated threat analysis, assessing and planning mitigations for identified risks, generating documentation, and reviewing how to iterate the model as systems evolve. With firsthand experience of the threat modeling lifecycle, you will be equipped to apply these risk analysis techniques to real-world environments. Now, let’s launch into the lab!

The subsequent sections will walk through the step-by-step lab, from downloading and configuring Microsoft TMT to constructing data flow diagrams, analyzing output, documenting findings, and understanding how to continuously update models to meet the dynamic threats and adapting architectures that cybersecurity architects face in their roles.

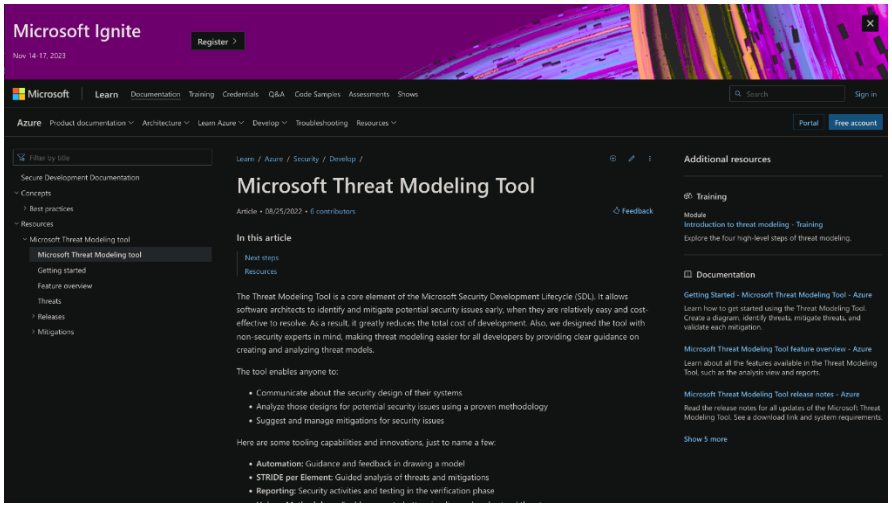


Figure 1 – Microsoft Ignite Threat Modeling Tool

These are the prerequisites:

- A Windows-based computer system
- Administrative privileges for software installation

With an understanding of the value of threat modeling, we now launch hands-on by installing Microsoft's Threat Modeling Tool. This section will guide you through visiting Microsoft's site to download the latest Threat Modeling Tool installer package, running the package with administrative privileges to install the software, completing the provided on-screen prompts, launching the tool from your start menu or desktop shortcut once setup finishes, and familiarizing yourself with the user interface.

With Microsoft TMT configured on your Windows machine, you have a powerful tool to construct robust threat models that serve as the scaffolding for risk identification and mitigation in your environments. The subsequent sections will build upon TMT's installation by utilizing its automated analysis and templated diagrams to methodically assess sample architecture, surface risks, define mitigating controls, and create shareable documentation - all critical aspects of the threat modeling lifecycle.

So let's get started by downloading the Microsoft Threat Modeling Tool and getting it running on our test system! The installation walkthrough is provided in detail next:

1. First let us look at the installation and initial configuration. Download Microsoft Threat Modeling Tool. Visit the official Microsoft website (https://learn.microsoft.com/en-us/azure/security/develop/threat-modeling-tool) or repository hosting the TMT:

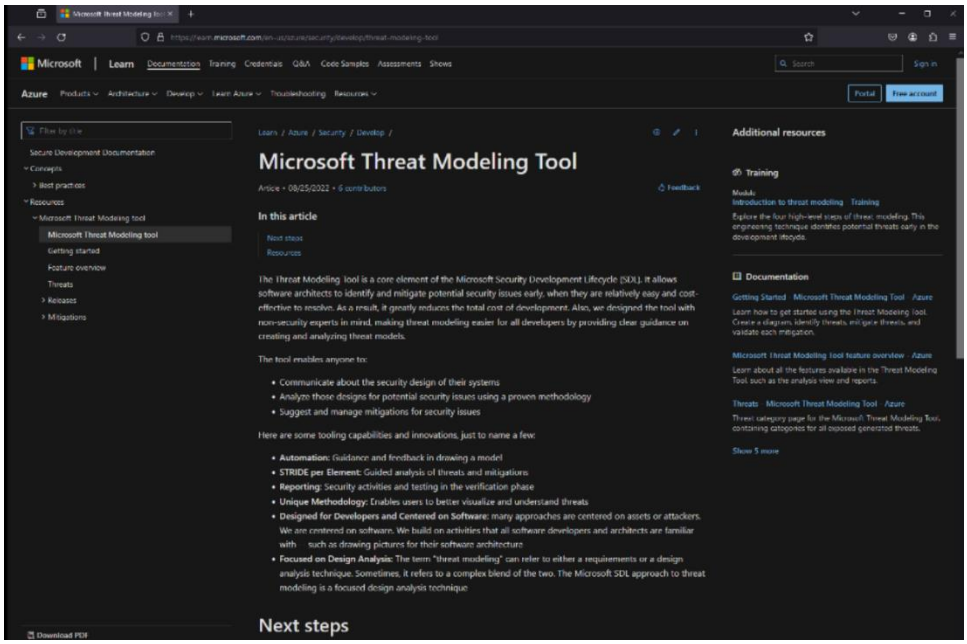


Figure 2 – Microsoft Threat Modeling Tool

2. Download the latest version of the TMT installer package:

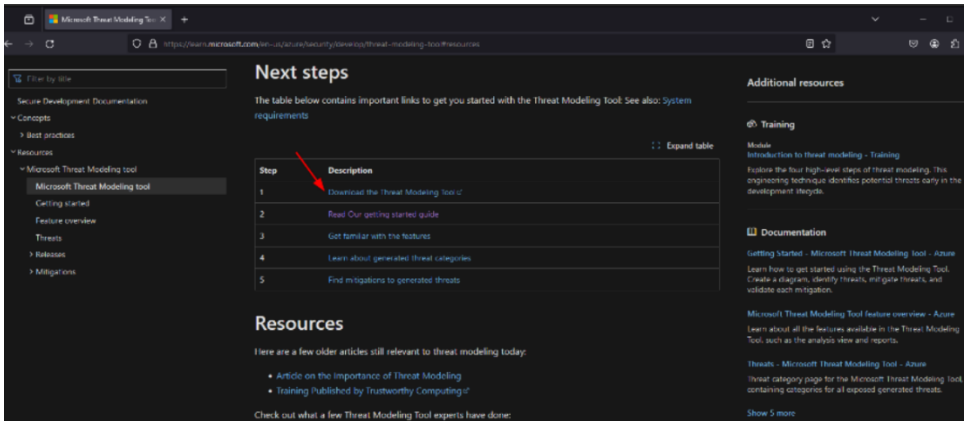


Figure 3 – Download Threat Modeling Tool

3. To install the tool, run the installer package with administrative privileges.

4. Follow the on-screen instructions to complete the installation process:

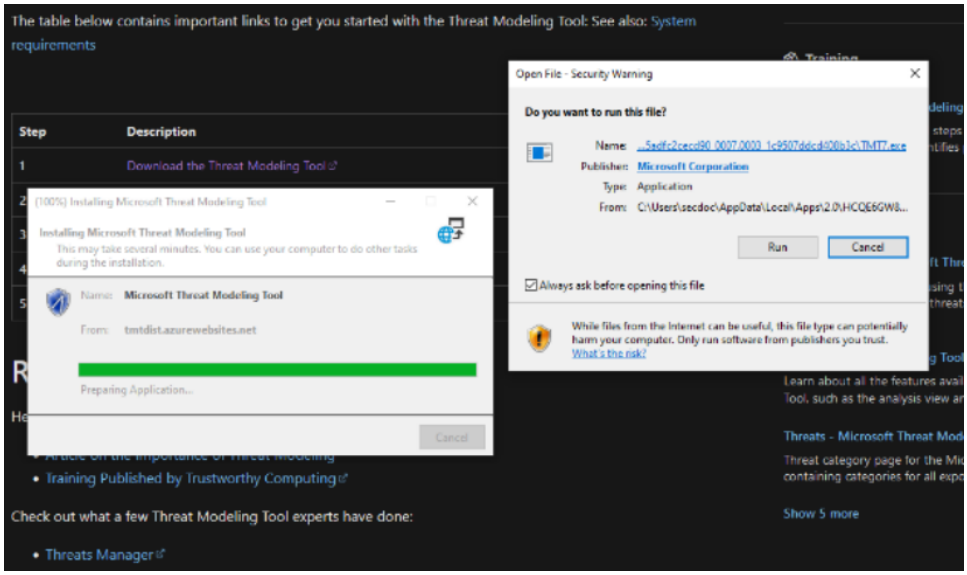


Figure 4 – Installation of Threat Modeling Tool

5. Open TMT and configure the environment. Launch the TMT from the Start Menu or desktop shortcut. Familiarize yourself with the user interface upon opening.

6. Now we move on to building the Threat Model. Create a new model. Click on create **New Model** from the dashboard menu:

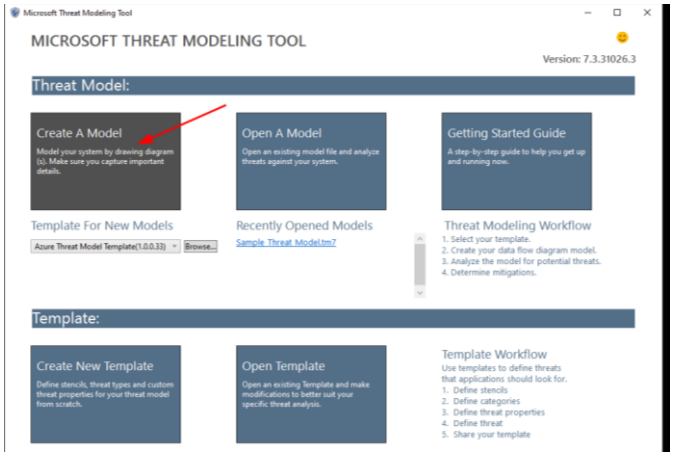


Figure 5 – Microsoft Threat Modeling Tool Dashboard

7. The **Add Diagram** window will appear to create the base diagram:

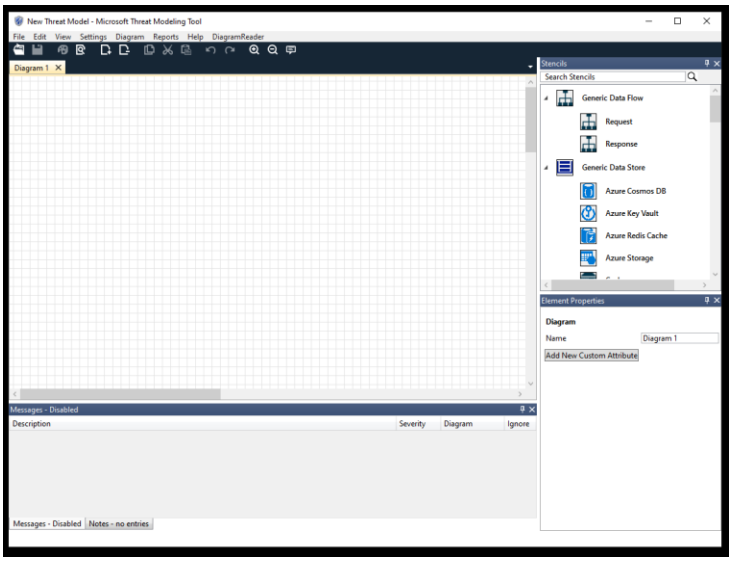


Figure 6 – Adding a Diagram

8. Edit the diagram. Drag and drop elements such as external entities (**User Accounts**), processes (**Web Server**, **Application Server**, **Database Server**), data stores, and data flows between them:

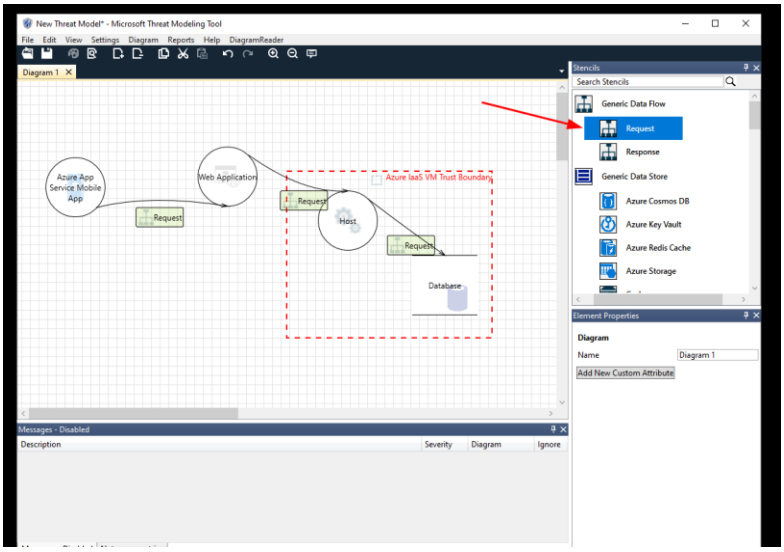


Figure 7 – Adding Elements to Diagram for Data Flows

9. Define trust boundaries by identifying and outlining the perimeter where trust levels change, e.g., the internet boundary, internal network, etc.:

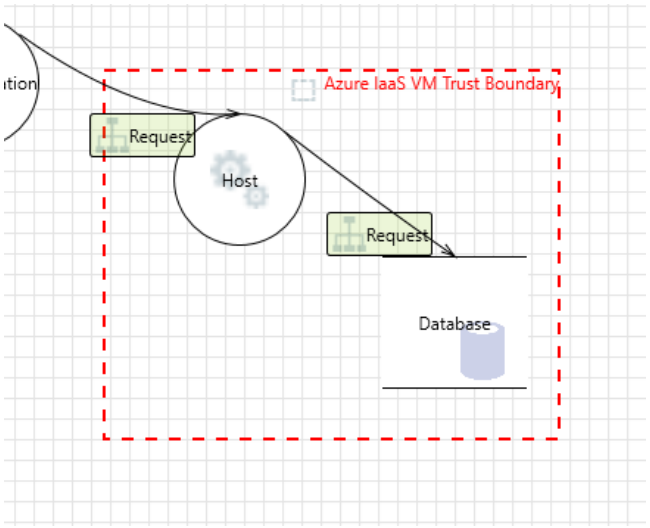


Figure 8 – Trust Boundary Definition

10. Click **View** and select **Analysis**. Select the ID within the **Threat** list to identify details about the threat properties:

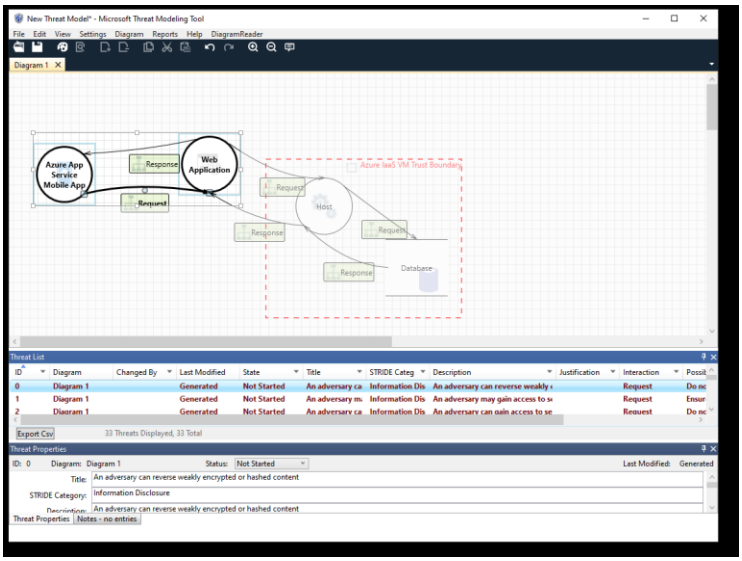


Figure 9 – Threat Analysis

11. Review the threats identified by the tool, such as SQL injection, **Cross-Site Scripting** (**XSS**), or account hijacking:

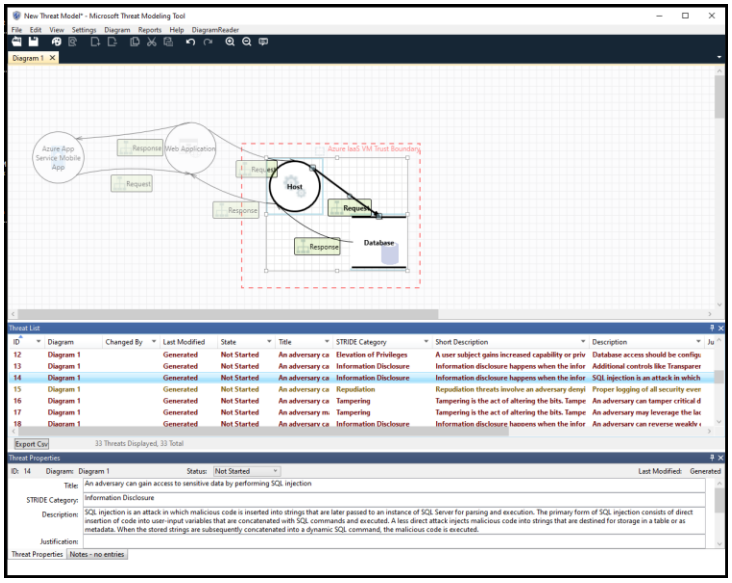


Figure 10 – Threat Details

12. Now to assess and plan mitigation, for each identified threat, use the tool to assess the potential impact.

13. Document remediation options such as input validation, employing parameterized queries, and implementing robust access controls:

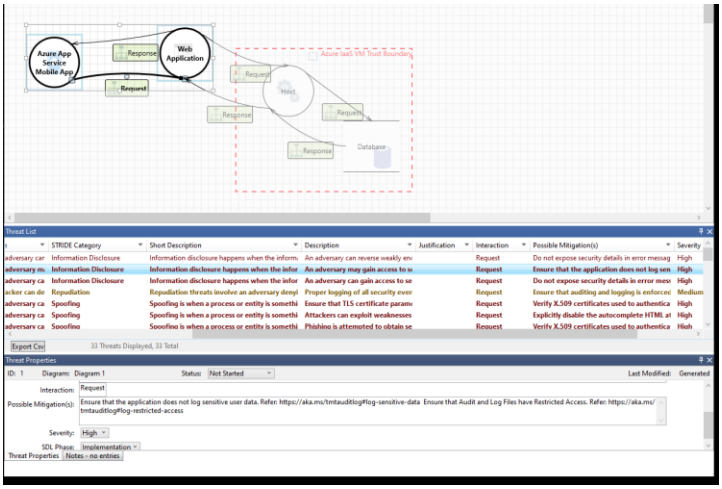


Figure 11 – Possible Mitigation Documentation

14. Now, we generate documentation. Create threat modeling reports via the **Report** feature:

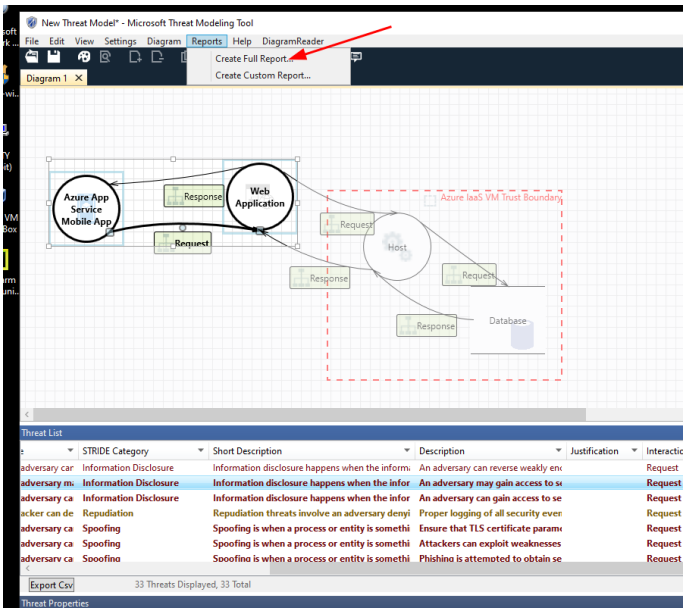


Figure 12 – Report Creation

15. Capture the identified risks, their priority, and the status of the mitigations:

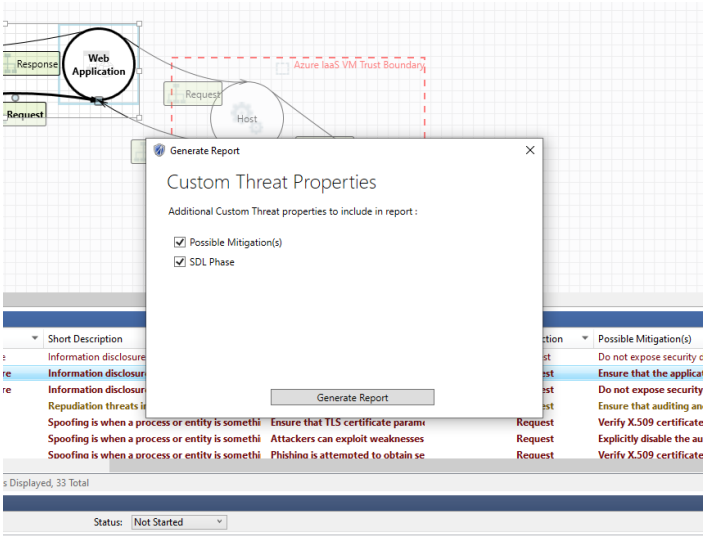


Figure 13 – Report Export

16. Save and distribute the threat model documentation among the development and security teams:

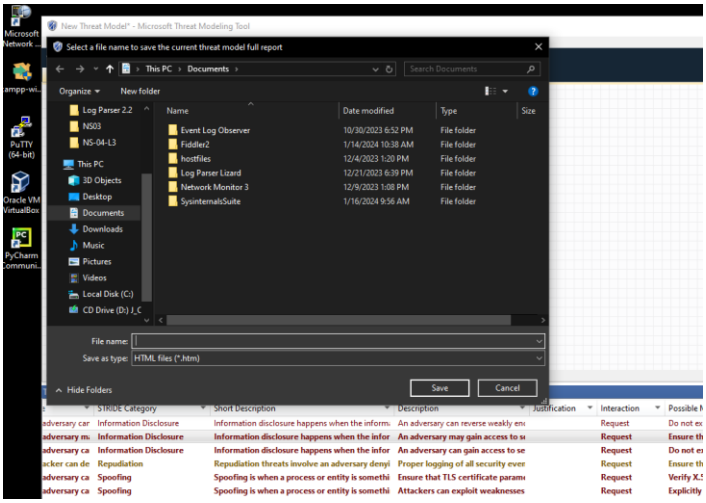


Figure 14 – Saving Report

17. Iterate as design evolves. Regularly revisit the threat model as system design changes or as new threats emerge:

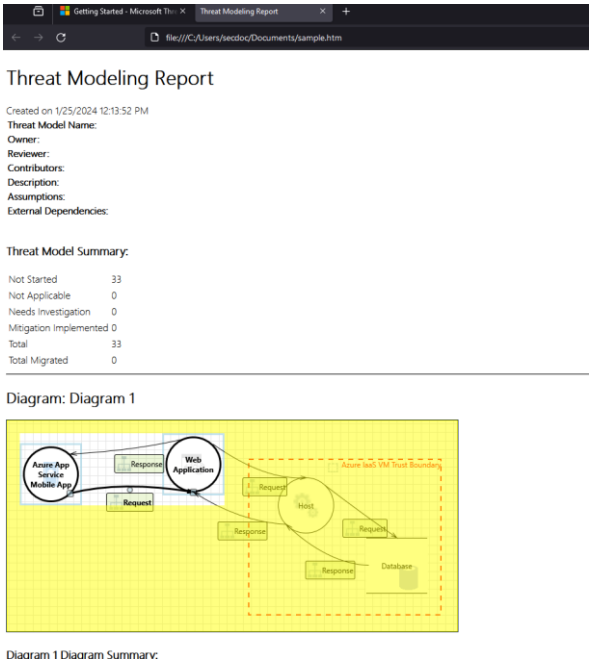


Figure 15 – Threat Model Report

18. Update the model and re-run the threat analysis as needed.

Lab 2: OWASP Threat Dragon

OWASP Threat Dragon is an open-source threat modeling tool that helps cybersecurity professionals and software developers identify, understand, and mitigate potential security threats in their applications and systems. Developed by the **Open Web Application Security Project** (**OWASP**), Threat Dragon provides a user-friendly interface for creating and sharing threat models, promoting collaboration among team members. By visually representing the architecture of a system and its associated threats, Threat Dragon enables users to proactively address security concerns early in the development process. This tool supports various threat modeling methodologies, such as STRIDE and LINDDUN, and allows users to customize threat libraries to suit their specific needs. With its intuitive design and extensive documentation, OWASP Threat Dragon is an essential tool for organizations looking to enhance their application security and build more resilient systems.

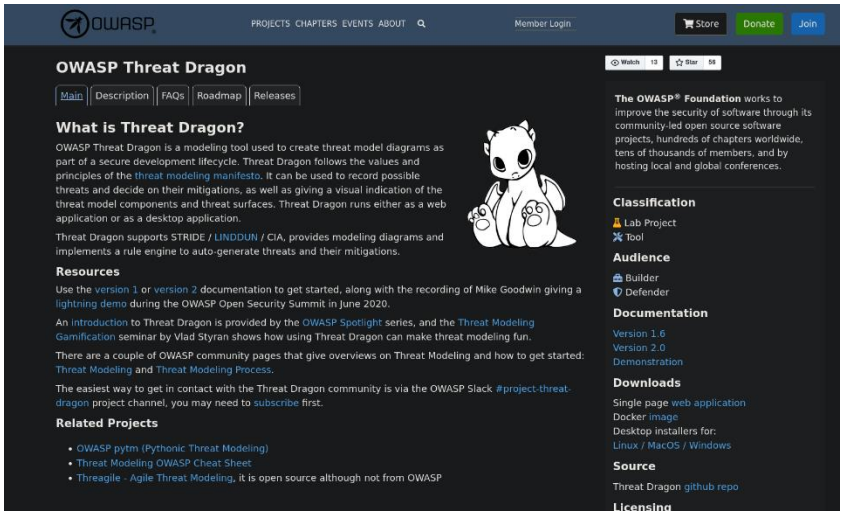


Figure 16 – OWASP Threat Dragon

The prerequisites are:

- A computer system with a modern web browser or desktop application support
- Network access for downloading software

Let us look at the lab:

1. Download and install OWASP Threat Dragon. Navigate to the OWASP Threat Dragon GitHub page or official website (https://owasp.org/www-project-threat-dragon/).
2. Download the desktop version appropriate for your operating system or access the web application:

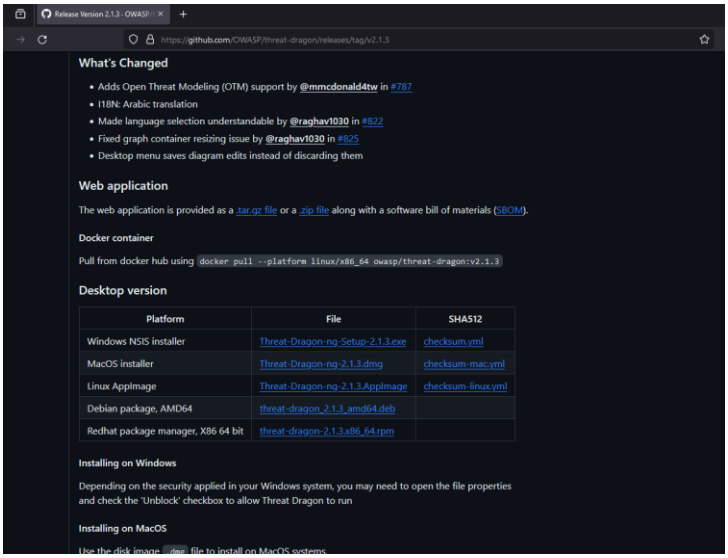


Figure 17 – Thread Dragon Download

3. Install the desktop application if applicable, following the instructions provided.

> **Note**
>
> Version 2 threat models are not backward compatible with Version 1 models, so if you plan to start with version 1 and then move to version 2, realize you will be required to recreate the models.

4. Open Threat Dragon and familiarize yourself with its interface:

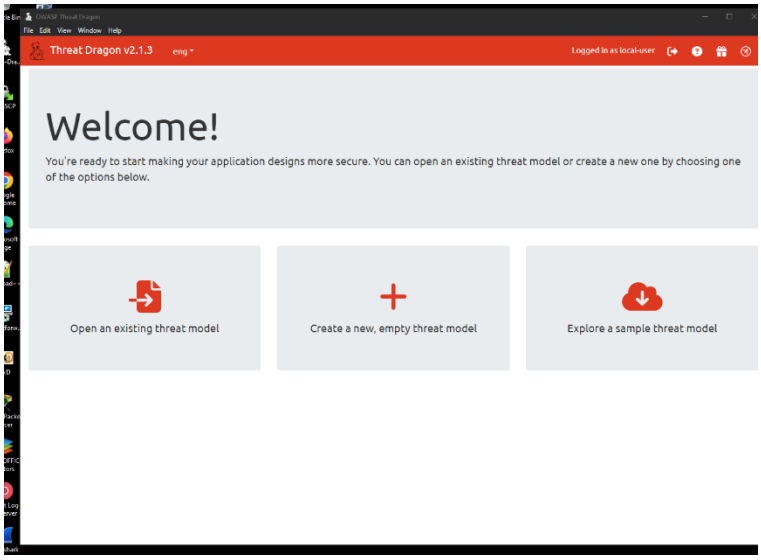


Figure 18 – Threat Dragon Dashboard

5. Now let's look at modeling and analysis. First, we look at creating a new project. Start a new project by specifying a name and description for your system under analysis:

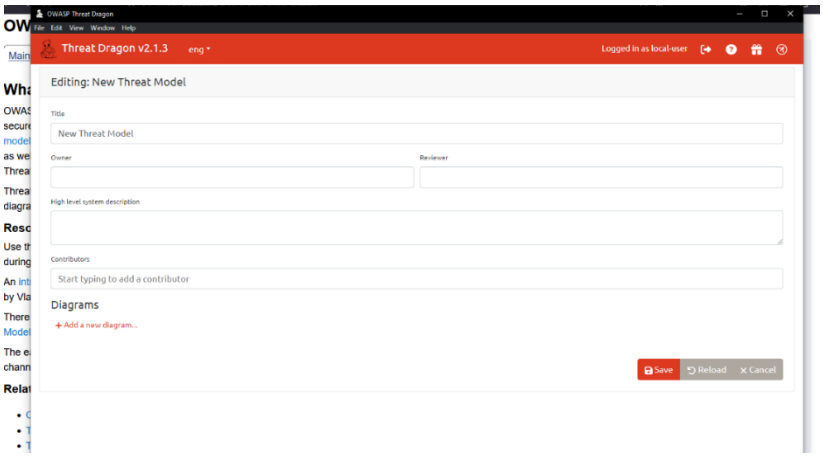


Figure 19 – Threat Model Creation

6. Develop a **Data Flow Diagram** (**DFD**). Utilize the diagramming features to create a DFD, placing processes, data stores, data flows, actors, and trust boundaries:

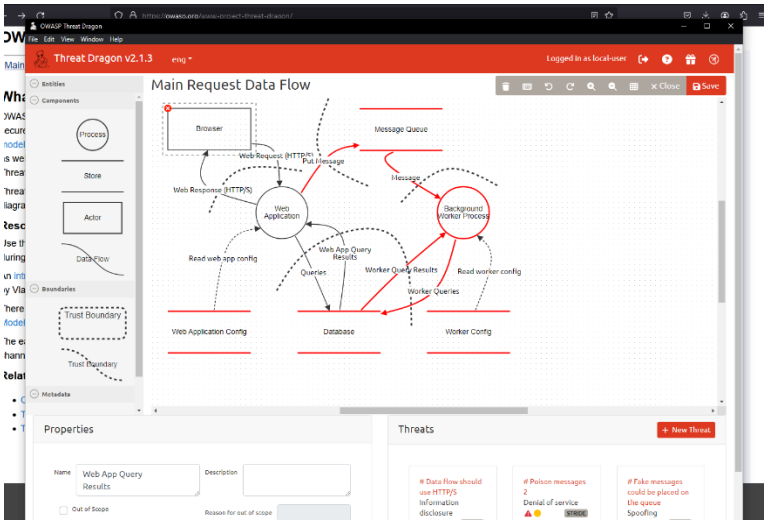


Figure 20 – Data Flow Diagram

7. Apply threat analysis. Engage threat libraries such as STRIDE or choose from pre-configured Attack Libraries to apply threat rules to your model.

8. Execute the threat analysis to identify potential security issues.

9. Review and detail threats. Examine each identified threat and assign a risk rating and priority level:

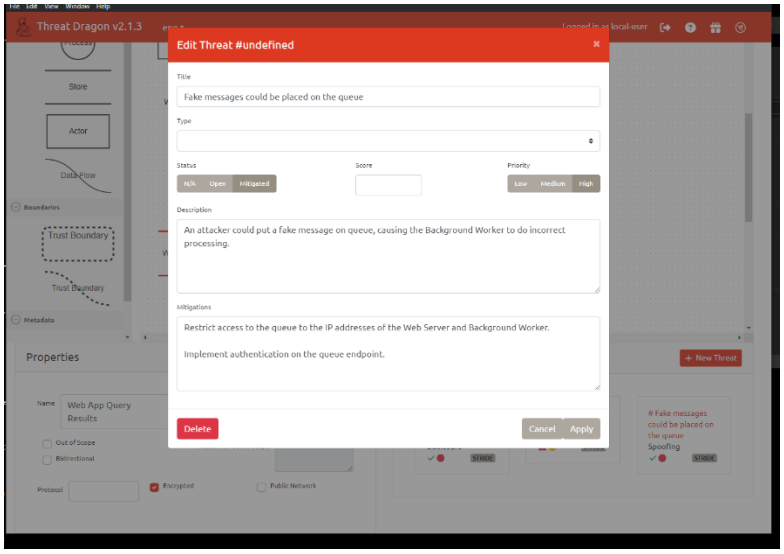


Figure 21 – Editing Threat Information

10. Document the mitigation strategies and countermeasures for each threat.

11. Generate reports that summarize the threat analysis findings:

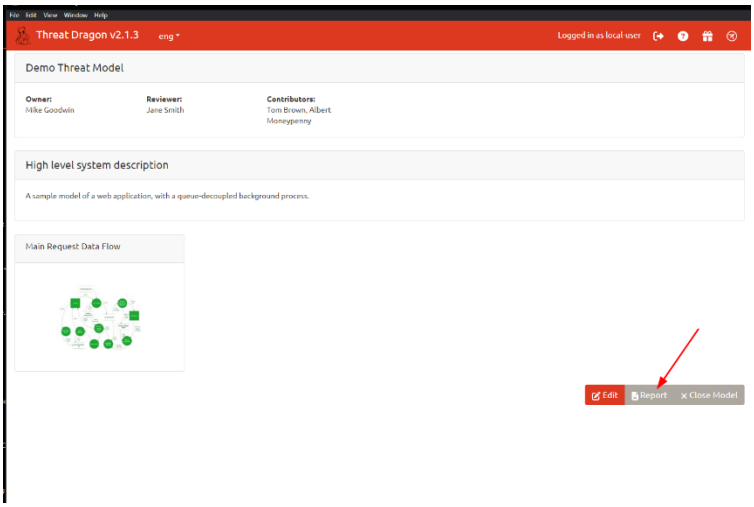


Figure 22 – Report Generation

12. Use these reports to communicate with stakeholders, guiding subsequent security hardening efforts:

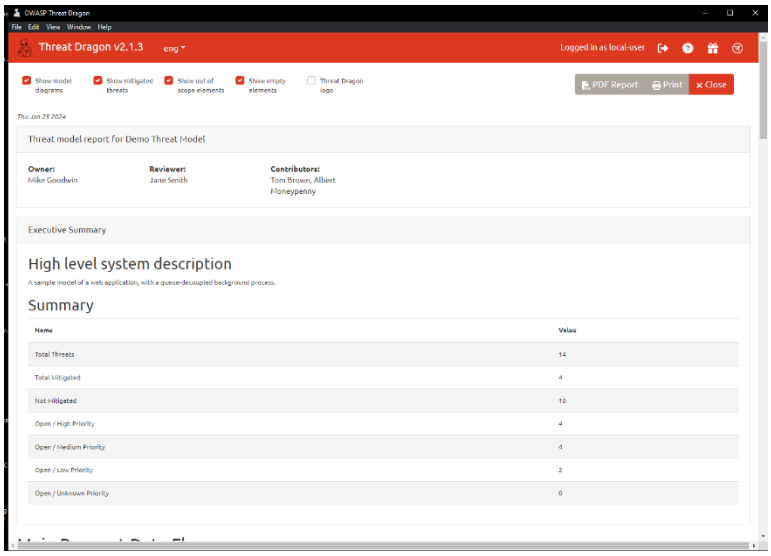


Figure 23 – Threat Model Report

13. Next, for continuous updating—update the threat model regularly, particularly with changes in the system architecture or after the discovery of new vulnerabilities.

14. Rerun the analysis and revise the mitigation strategies as required.

In both labs, it is essential to document the learning outcomes and ensure that the threat model remains a living document, iteratively improved upon as the system it represents evolves. Additionally, these tools should be integrated within the SDLC to enhance the security posture from the earliest stages of development.

Network defense and monitoring tools

Let us look at the labs.

Lab 1: Intrusion Detection/Prevention System (IDS/IPS) Using Snort in a Virtual Environment

Snort is a widely-used, open-source **intrusion detection system** (**IDS**) that helps network administrators and security professionals monitor and protect their networks from various security threats. Developed by Martin Roesch in 1998 and now maintained by Cisco Systems, Snort is a powerful tool that performs real-time traffic analysis and packet logging on IP networks. It uses a rule-based language to detect and alert users about potential security breaches, such as malware infections, port scans, and attempts to exploit known vulnerabilities. Snort can be configured to work as a passive IDS, simply monitoring network traffic and raising alerts, or as an **intrusion prevention system** (**IPS**), actively blocking suspicious traffic. With its extensive community support, regular rule updates, and compatibility with various platforms, Snort has become an essential component of many organizations' cybersecurity strategies, helping them detect and respond to threats more effectively.

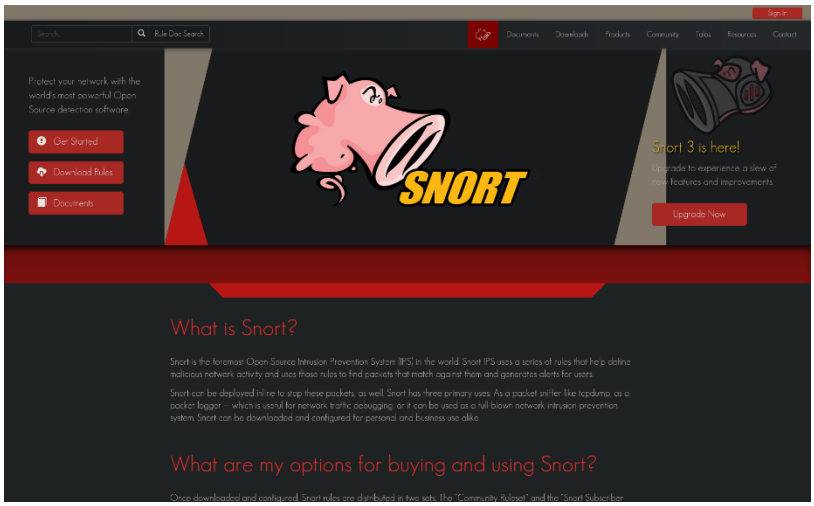


Figure 24 – Snort Website

The link to access it is at https://www.snort.org/.

The prerequisites are:

- Virtualization software (e.g., VirtualBox, VMware)
- A **virtual machine** (**VM**) with a Linux-based system installed
- Network access within the virtual environment
- Administrative privileges for software installation within the VM

Let us look at the steps:

1. Create a new VM within your virtualization software. Allocate sufficient resources (CPU, Memory, and Network Adapters).
2. Install a Linux distribution (such as Ubuntu) on the VM.
3. Configure one network adapter in bridged mode and another in host-only or internal network mode for testing.
4. Update your system's package database with (for Debian-based systems):

```
sudo apt-get update
```

5. Install Snort using the package manager with `sudo apt-get install snort`:

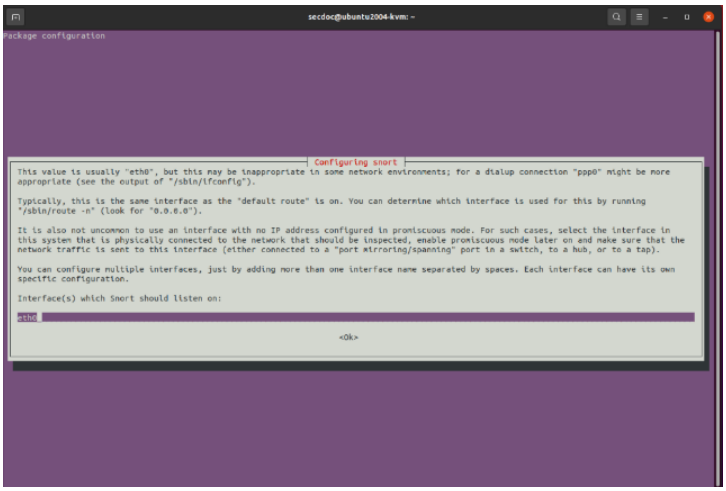


Figure 25 – Assigning Listening Interface for Snort

6. During installation, configure the network interface you want Snort to monitor.
7. Set up the internal network variable in the Snort configuration file:

```
/etc/snort/snort.conf
```

8. Now, let's look at rule management. Navigate to `/etc/snort/rules`.

9. Create or update Snort rules to define what traffic should be inspected.

10. With the configuration and rule files in place, edit the `snort.conf` to modify a few parameters. Open the configuration file in your favorite text editor, for example using nano with the following command:

```
sudo nano /etc/snort/snort.conf
```

Find these following sections in the configuration file and change the parameters to reflect the examples here:

```
# Setup the network addresses you are protecting
ipvar HOME_NET server_public_ip/32

# Set up the external network addresses. Leave as
"any" in most situations
ipvar EXTERNAL_NET !$HOME_NET

# Path to your rules files (this can be a relative
path)
var RULE_PATH /etc/snort/rules
var SO_RULE_PATH /etc/snort/so_rules
var PREPROC_RULE_PATH /etc/snort/preproc_rules

# Set the absolute path appropriately
var WHITE_LIST_PATH /etc/snort/rules
var BLACK_LIST_PATH /etc/snort/rules
```

In the same `snort.conf` file, scroll down to the `section 6` and set the output for `unified2` to log under filename of `snort.log`, as follows:

```


# unified2
# Recommended for most installs
output unified2: filename snort.log, limit 128
```

Lastly, scroll down towards the bottom of the file to find the list of included rule sets. You will need to uncomment the `local.rules` to allow Snort to load any custom rules:

```
include $RULE_PATH/local.rules
```

11. Once you are done with the configuration file, save the changes and exit the editor.

12. Now, next to validating settings. Your Snort should now be ready to run. Test the configuration using the parameter `-T` to enable test mode:

```
sudo snort -T -c /etc/snort/snort.conf
```

After running the Snort configuration test, you should get a message like the following example:

```
--== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.16 GRE (Build 118)
   ''''    By Martin Roesch & The Snort Team:
http://www.snort.org/contact#team
           Copyright (C) 2014-2020 Cisco and/or its
affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc.,
et al.
           Using libpcap version 1.8.1
           Using PCRE version: 8.39 2016-06-14
           Using ZLIB version: 1.2.11

           Rules Engine: SF_SNORT_DETECTION_ENGINE
Version 3.1
           Preprocessor Object: SF_DCERPC2  Version
1.0
           Preprocessor Object: SF_SSH  Version 1.1
           Preprocessor Object: SF_FTPTELNET  Version
1.2
           Preprocessor Object: SF_SDF  Version 1.1
           Preprocessor Object: SF_DNP3  Version 1.1
           Preprocessor Object: SF_REPUTATION  Version
1.1
           Preprocessor Object: SF_IMAP  Version 1.0
           Preprocessor Object: SF_SMTP  Version 1.1
           Preprocessor Object: SF_GTP  Version 1.1
           Preprocessor Object: appid  Version 1.1
```

```
            Preprocessor Object: SF_MODBUS  Version 1.1
            Preprocessor Object: SF_POP  Version 1.0
            Preprocessor Object: SF_DNS  Version 1.1
            Preprocessor Object: SF_SSLPP  Version 1.1
            Preprocessor Object: SF_SIP  Version 1.1

Snort successfully validated the configuration!
Snort exiting
```

This is how it'll look:

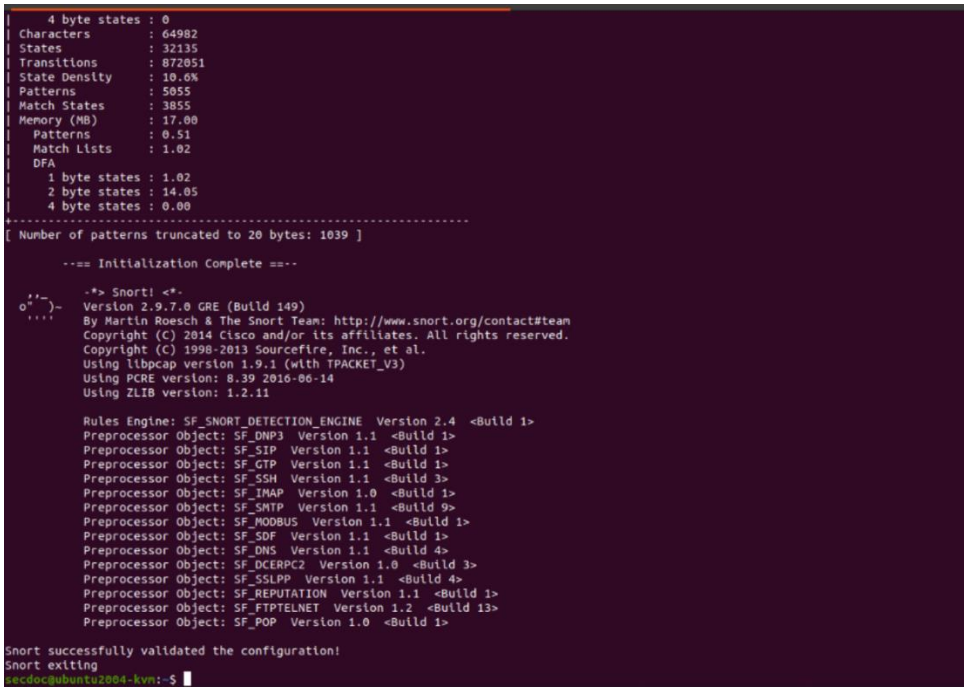


Figure 26 – Snort Initialization

In case you get an error, the response in the terminal should tell you what the problem was and where to fix it. Most likely problems are missing files or folders, which you can usually resolve by either adding any you might have missed in the setup above, or by commenting out unnecessary inclusion lines in the `snort.conf` file. Check the configuration part and try again.

13. To test if Snort is logging alerts as intended, add a custom detection rule alert on incoming ICMP connections to the `local.rules` file. Open your local rules in a text editor. Then add the following example line to the file:
```
alert tcp any any -> 192.168.1.0/24 80 (msg:"Possible Web Traffic"; sid:1000001;)
```

14. Run Snort in IDS mode. Execute Snort in console mode with the following:

```
sudo snort -q -A console -i [interface] -c /etc/snort/snort.conf -l /var/log/snort
```

15. Run Snort in IPS mode. Implement inline mode by adding the `-Q` switch and using appropriate inline-specific rules.

16. For testing and validation, use another VM or the host machine to generate network traffic that Snort should detect. Ensure this testing machine is connected to the same virtual network.

17. Review the alerts in `/var/log/snort`.

Lab 2: Firewall Configuration Using OPNsense in a Virtual Environment

OPNsense is a powerful, open-source firewall and routing platform based on FreeBSD. It provides a feature-rich, user-friendly web interface for managing and securing networks of all sizes. As a fork of pfSense, OPNsense has quickly gained popularity among network administrators and cybersecurity professionals for its stability, performance, and extensive set of built-in security features. These features include a stateful firewall, IDS/IPS, **virtual private network** (**VPN**) support, traffic shaping, and more. OPNsense also offers a plugin system that allows users to extend its functionality with additional security tools and services, such as Suricata, Snort, and OpenVPN. With its active community, regular updates, and commitment to open-source principles, OPNsense has established itself as a reliable and flexible solution for organizations looking to protect their networks from various security threats while maintaining a high level of control and customization.

Figure 27 – OPNsense Website

The prerequisites are a VM to install OPNsense with at least two network interfaces configured. Let us look at the steps:

1. Prepare a new VM with at least two network adapters.
2. One network adapter should be configured in bridged mode (to simulate the WAN), and the other in internal network mode (to simulate the LAN).
3. Download the OPNsense ISO from the official website, https://opnsense.org/.
4. Burn the ISO to a USB drive or attach it to a VM.
5. Boot from the drive or ISO and follow the installation prompts:

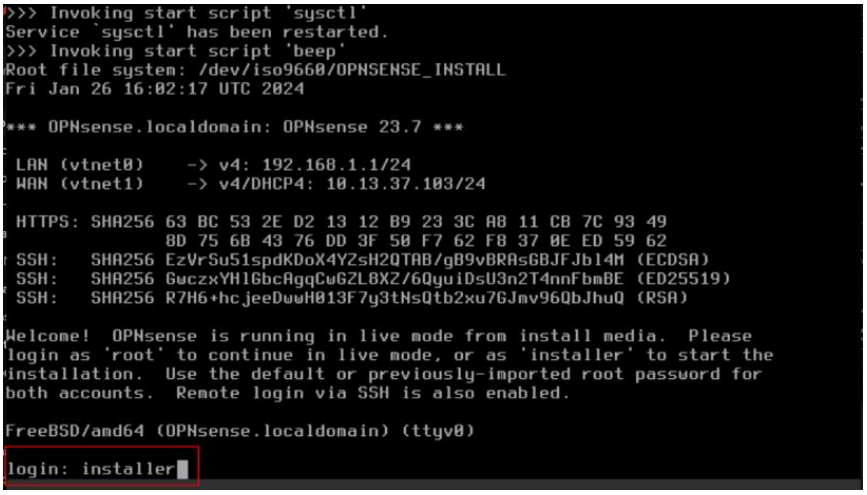


Figure 28 – Installation of OPNsense

The default password for OPNsense is `opnsense` and is the password to use for the default root and installer accounts.

Configure a keyboard:

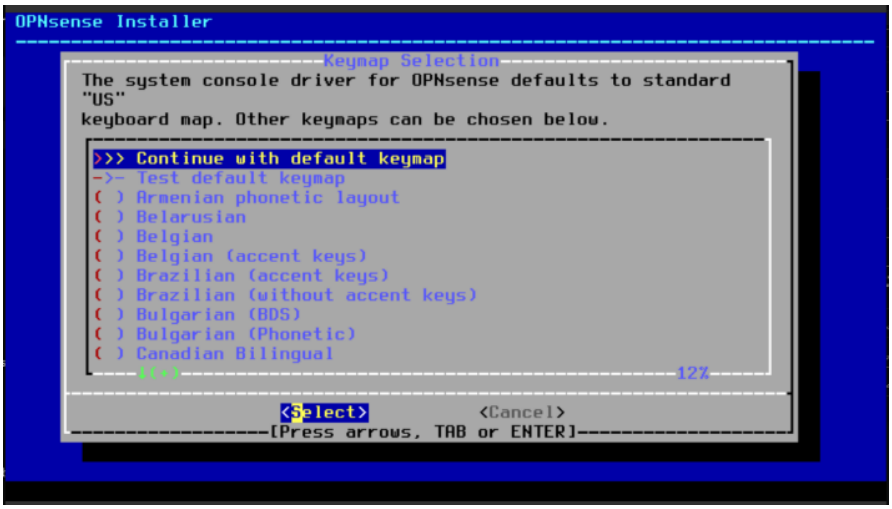


Figure 29 – Keyboard Mapping Configuration

The default is **Unix File System** (**UFS**), but I prefer the **Zettabyte File System** (**ZFS**), but either are acceptable selections:

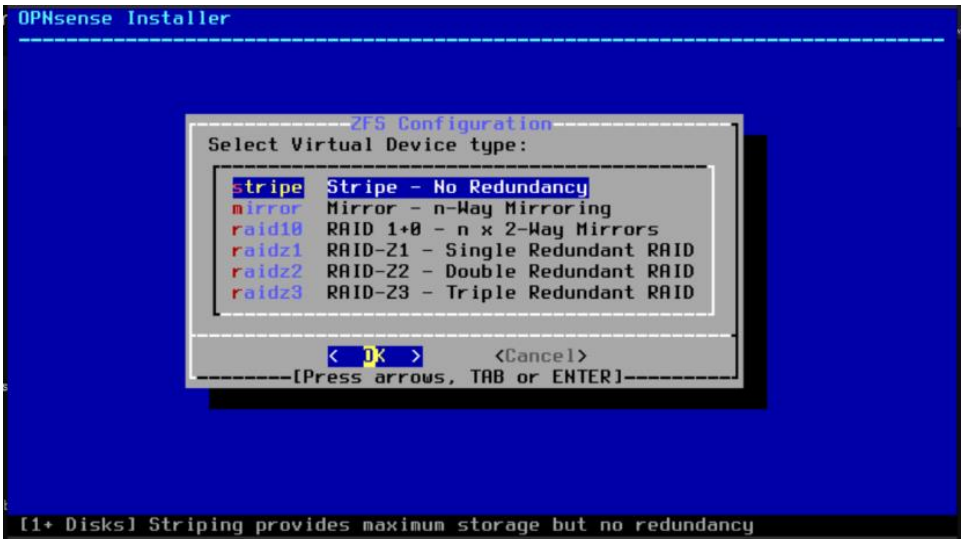


Figure 30 – ZFS Configuration

If this is a VM using a single disk, the **Stripe – No Redundancy** should the left as the default selection.

6. Press the spacebar to select the drive.

   Once you have selected the ZFS configuration, the next screen will allow you select the hard drive that OPNsense will be installed upon. Select the drive and click **OK**:

Figure 31 – Harddrive Selection

At this point, the screen now provides warning that you are about to destroy any data that exists on the disk selected:

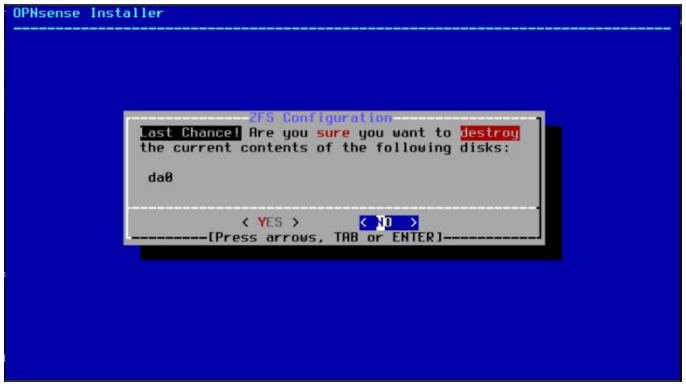


Figure 32 – Final Installation Warning

7. Click the *TAB* key on the keyboard to select **YES** to continue:

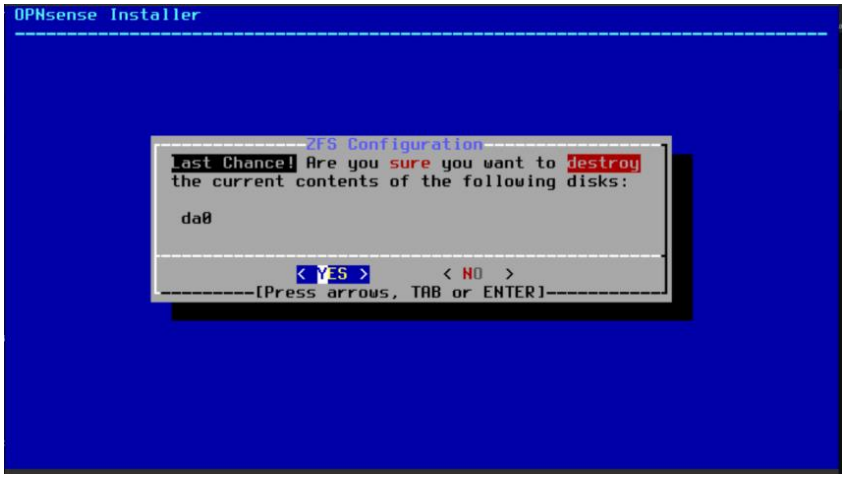


Figure 33 – Select YES to Install

8. At this point you can change the default root account password:

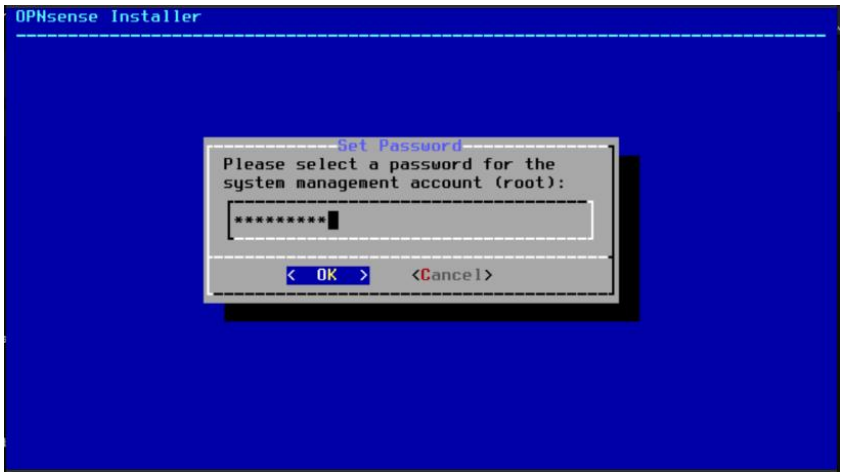


Figure 34 – Define Root Password

9. Upon changing the password, the following prompt will show up:

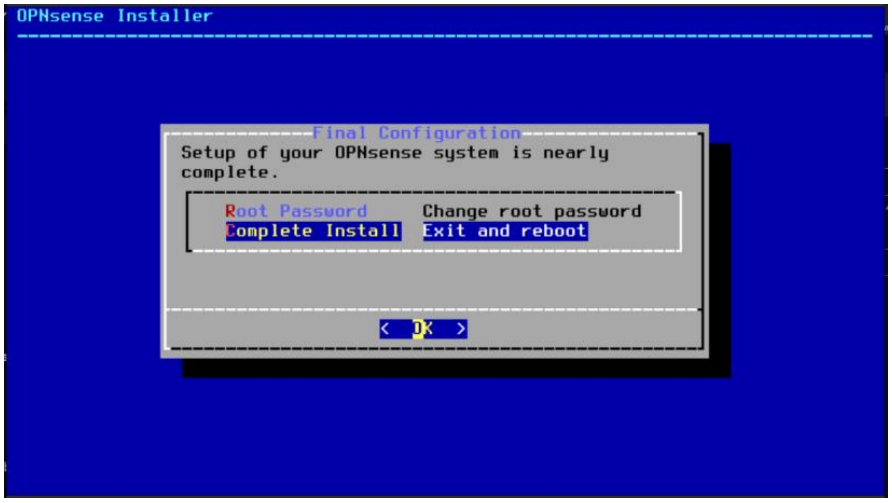


Figure 35 – Complete Installation

10. At this point you can change the LAN IP Addressing scheme and recommend that you do in the event there is a conflict with an existing LAN/Network IP range:

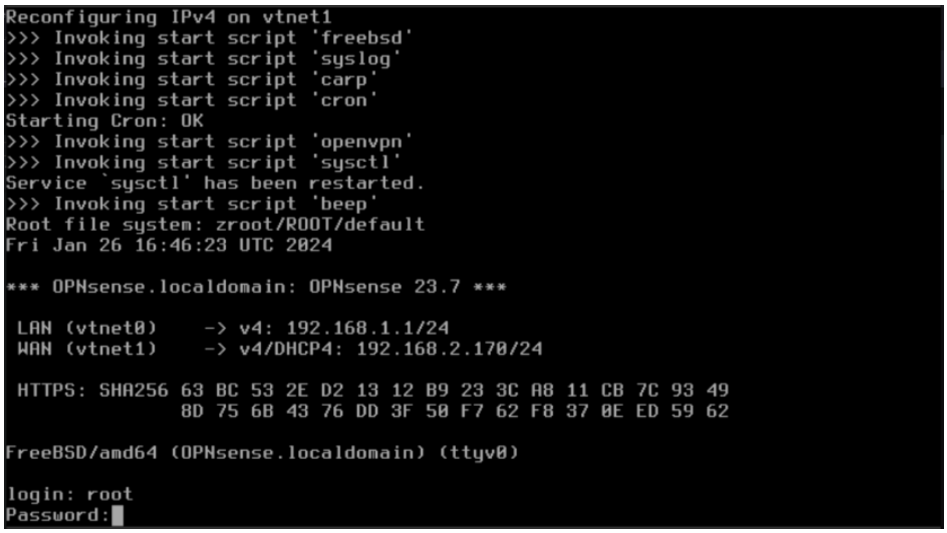


Figure 36 – Terminal Login for Initial Configuration

11. Select option 2 to assign LAN Interface IP Address:

Figure 37 – Interface Configuration

12. Select option `1` to configure LAN Interface and choose `N` to not assign IP address via DHCP:

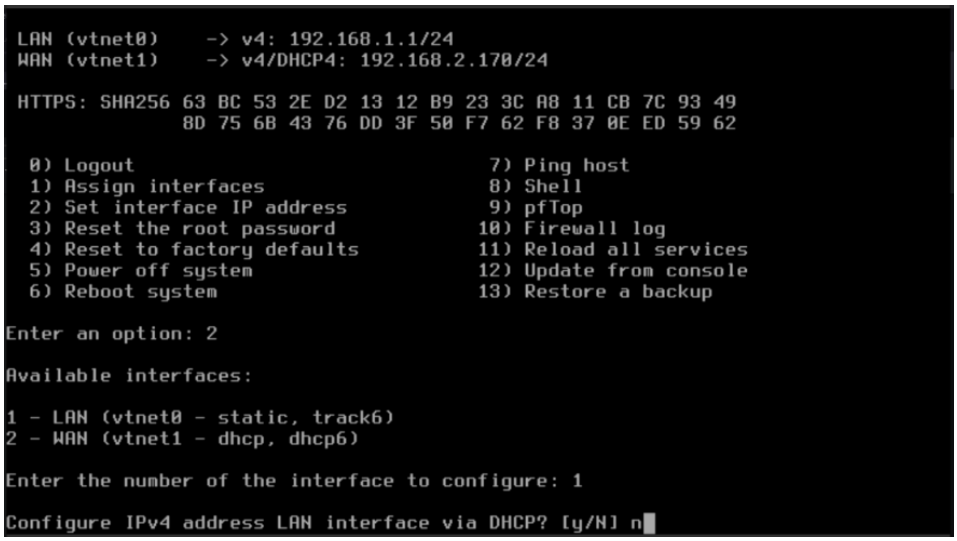


Figure 38 – Define LAN/WAN Interfaces

13. Enter the desired IP Address, such as the example `10.13.37.1`, and select the desired CIDR/subnet range such as the `/24` or `255.255.255.0` example:

```
  5) Power off system                   12) Update from console
  6) Reboot system                      13) Restore a backup

Enter an option: 2

Available interfaces:

1 - LAN (vtnet0 - static, track6)
2 - WAN (vtnet1 - dhcp, dhcp6)

Enter the number of the interface to configure: 1

Configure IPv4 address LAN interface via DHCP? [y/N] n

Enter the new LAN IPv4 address. Press <ENTER> for none:
> 10.13.37.1


Subnet masks are entered as bit counts (like CIDR notation).
e.g. 255.255.255.0 = 24
     255.255.0.0   = 16
     255.0.0.0     = 8

Enter the new LAN IPv4 subnet bit count (1 to 32):
> 24
```

Figure 39 – Define LAN Subnet Scheme

14. Press enter since there is no upstream gateway for the LAN:

```
Available interfaces:

1 - LAN (vtnet0 - static, track6)
2 - WAN (vtnet1 - dhcp, dhcp6)

Enter the number of the interface to configure: 1

Configure IPv4 address LAN interface via DHCP? [y/N] n

Enter the new LAN IPv4 address. Press <ENTER> for none:
> 10.13.37.1


Subnet masks are entered as bit counts (like CIDR notation).
e.g. 255.255.255.0 = 24
     255.255.0.0   = 16
     255.0.0.0     = 8

Enter the new LAN IPv4 subnet bit count (1 to 32):
> 24

For a WAN, enter the new LAN IPv4 upstream gateway address.
For a LAN, press <ENTER> for none:
>
```

Figure 40 – LAN Interface Upstream Gateway Configuration

15. While you can configure IPv6 addressing to do further testing, this has been disabled by selecting **No** and pressing *Enter*:

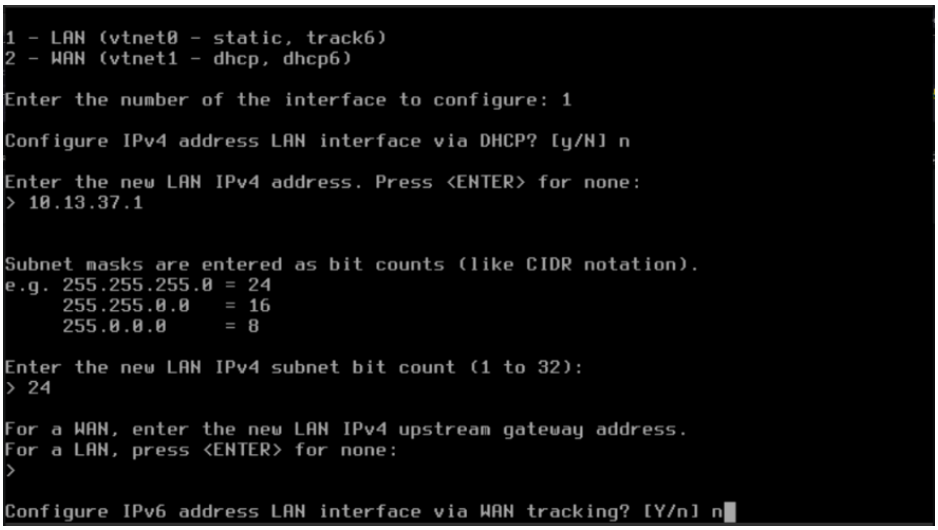


Figure 41 – LAN Interface IPv6 Configuration

16. To enable DCHP server on the LAN for attached devices, select `Yes`:

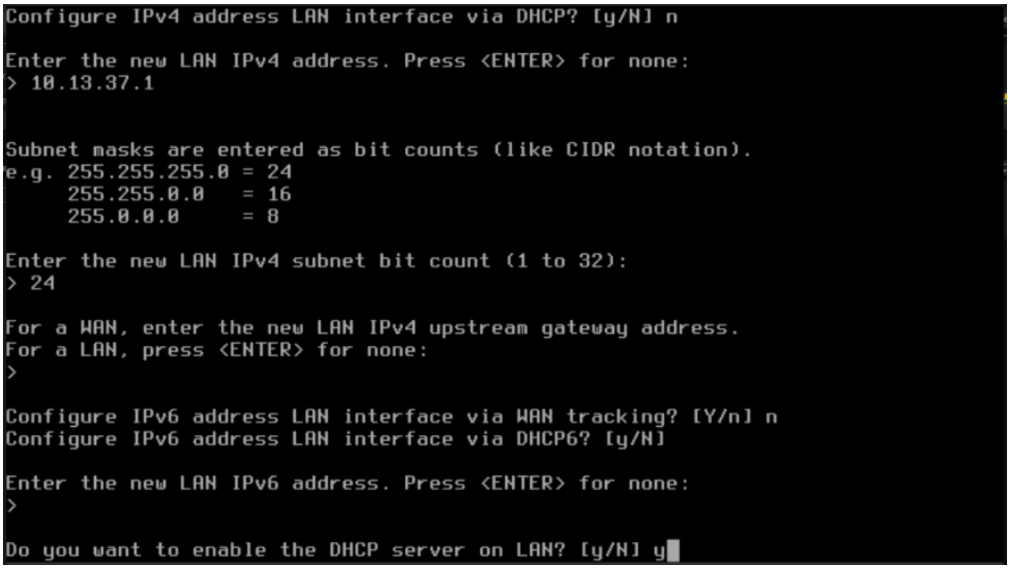


Figure 42 – Enabling a DHCP Server on the LAN Interface

17. Enter the starting and ending client IP address range as in the example `10.13.37.100`-`10.13.37.250`:

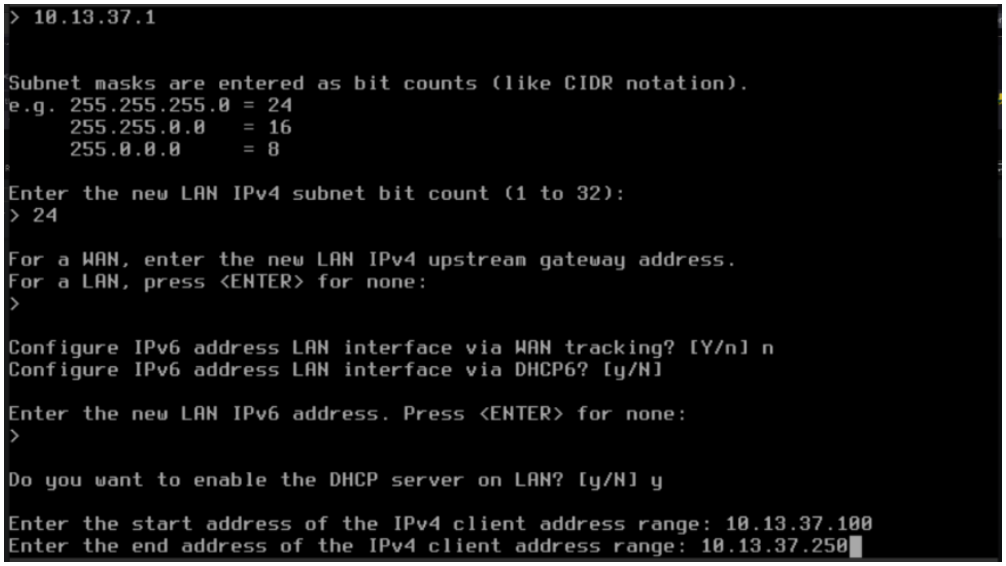


Figure 43 – Defining DHCP Scope

18. Select `No`, when prompted to change web GUI protocol:

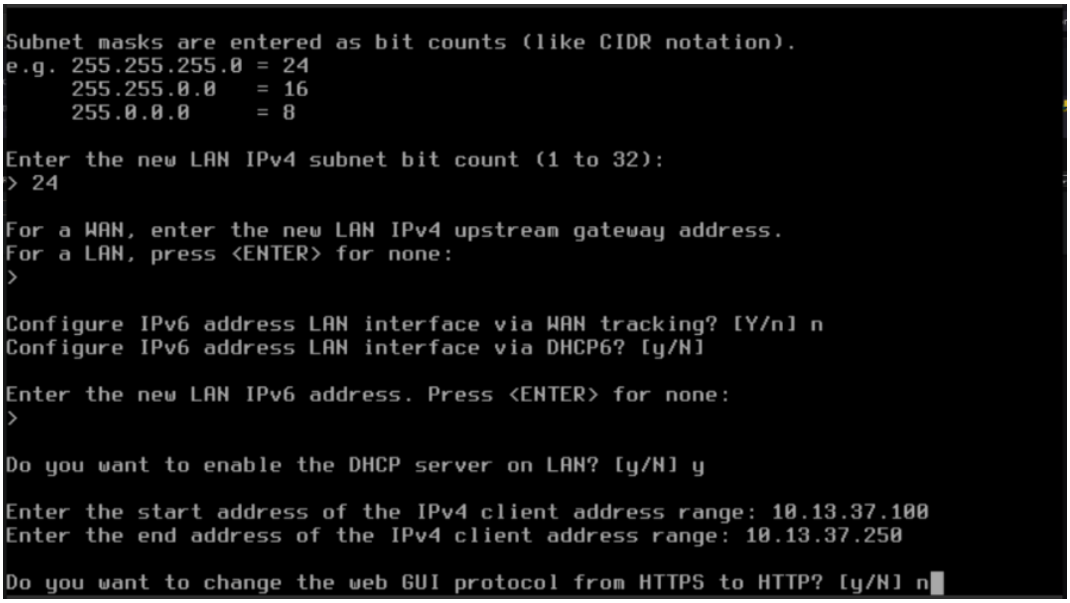


Figure 44 – HTTP/HTTP Web GUI Protocol

19. Select `Yes` to generate self-signed certificate:

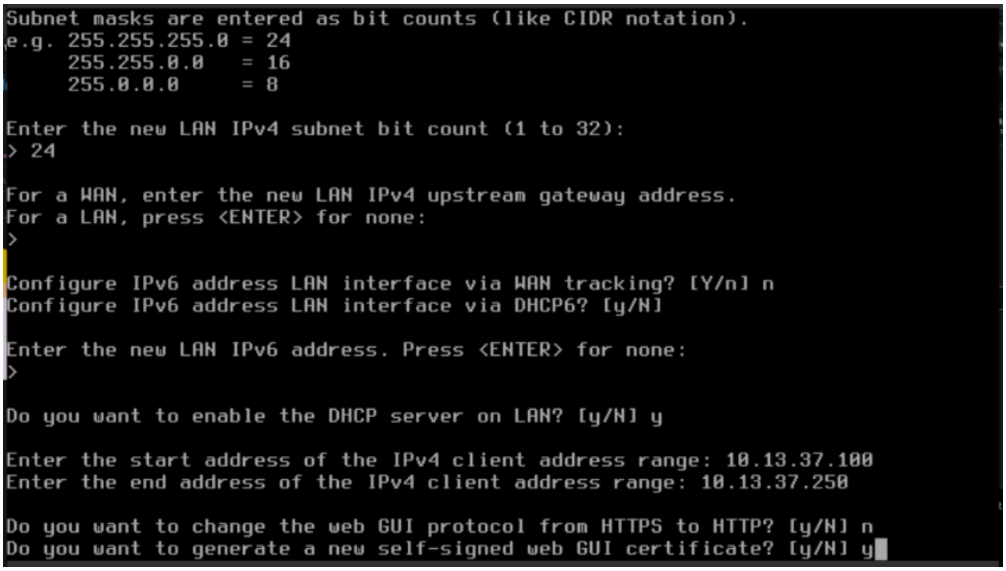


Figure 45 – Allow Self-signed Certificate

20. Select `No` to restore web GUI defaults:

```
e.g. 255.255.255.0 = 24
     255.255.0.0   = 16
     255.0.0.0     = 8

Enter the new LAN IPv4 subnet bit count (1 to 32):
> 24

For a WAN, enter the new LAN IPv4 upstream gateway address.
For a LAN, press <ENTER> for none:
>

Configure IPv6 address LAN interface via WAN tracking? [Y/n] n
Configure IPv6 address LAN interface via DHCP6? [y/N]

Enter the new LAN IPv6 address. Press <ENTER> for none:
>

Do you want to enable the DHCP server on LAN? [y/N] y

Enter the start address of the IPv4 client address range: 10.13.37.100
Enter the end address of the IPv4 client address range: 10.13.37.250

Do you want to change the web GUI protocol from HTTPS to HTTP? [y/N] n
Do you want to generate a new self-signed web GUI certificate? [y/N] y
Restore web GUI access defaults? [y/N]
```

Figure 46 – Web GUI Access Defaults

21. After installation, access the OPNsense web interface from another machine in the same network using the default IP provided at the end of the installation:

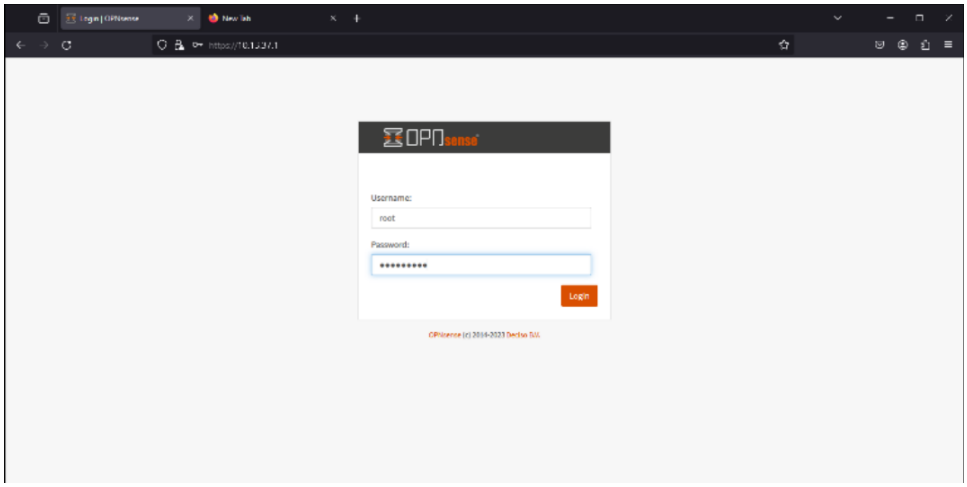


Figure 47 – OPNsense Web Management Interface

22. Let us now look at the initial setup wizard:

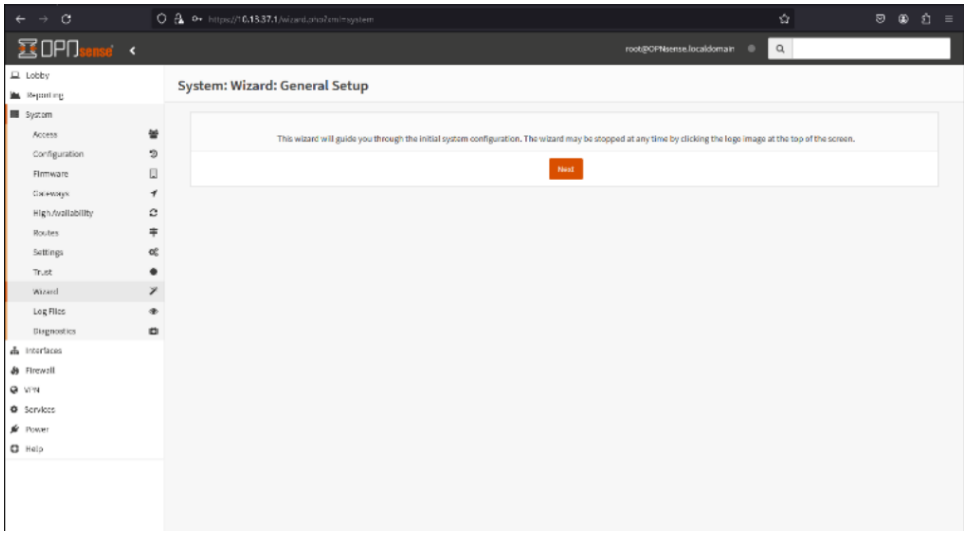


Figure 48 – OPNsense Setup Wizard

23. Use the setup wizard to configure the basic settings such as **Hostname**, **Domain**, **DNS servers**, and **Time zone**:

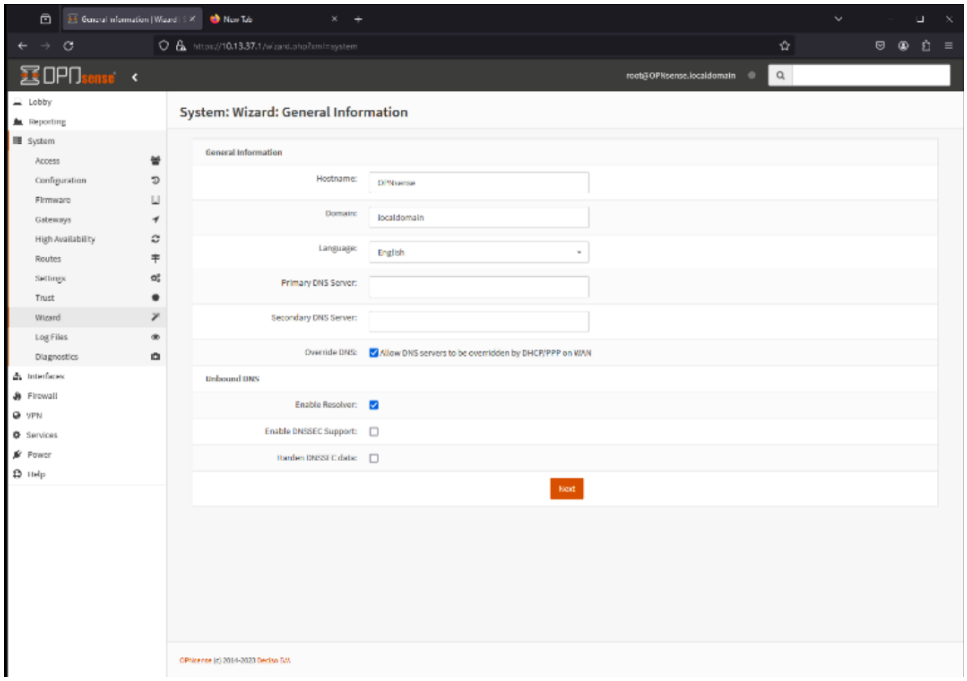


Figure 49 – Wizard General Configuration

24. Assign and configure WAN and LAN interfaces, ensuring proper IP address configuration. This should have been configured earlier in the steps, but if you need to adjust you can do so through the initial wizard setup:

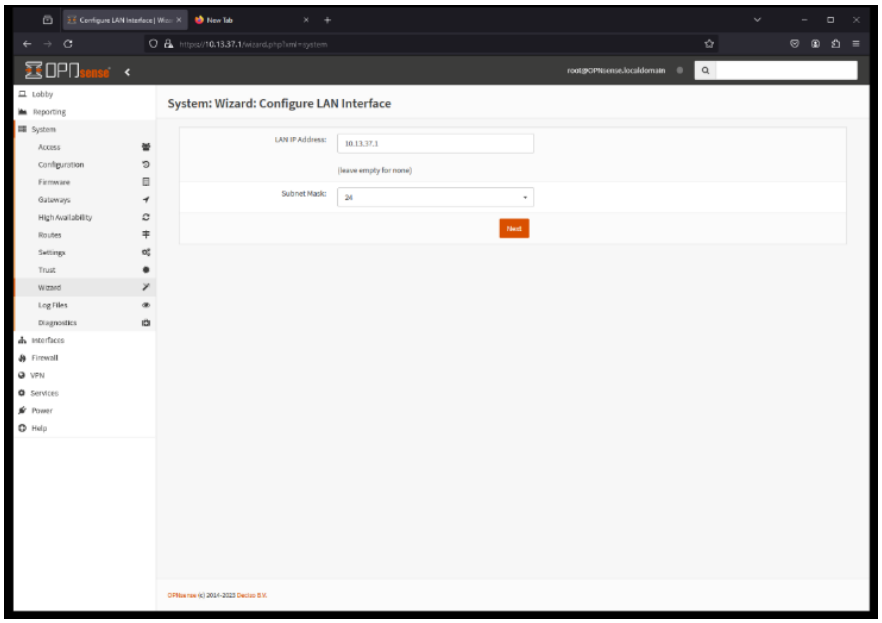


Figure 50 – Wizard LAN Configuration

25. The OPNsense dashboard is then presented:

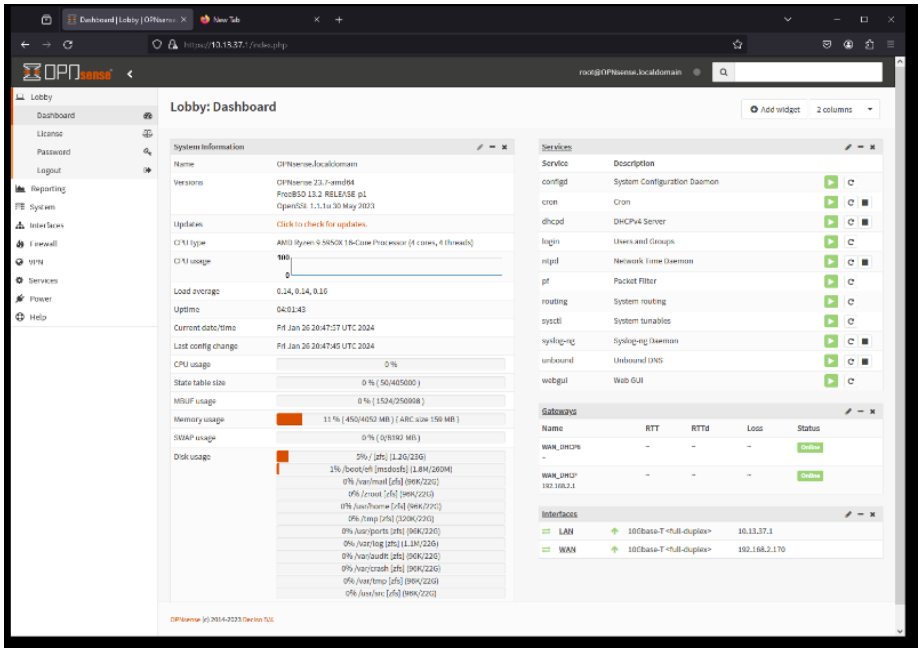


Figure 51 – OPNsense Dashboard

26. Now, let us look at the firewall rule creation. Navigate to **Firewall** | **Rules**:

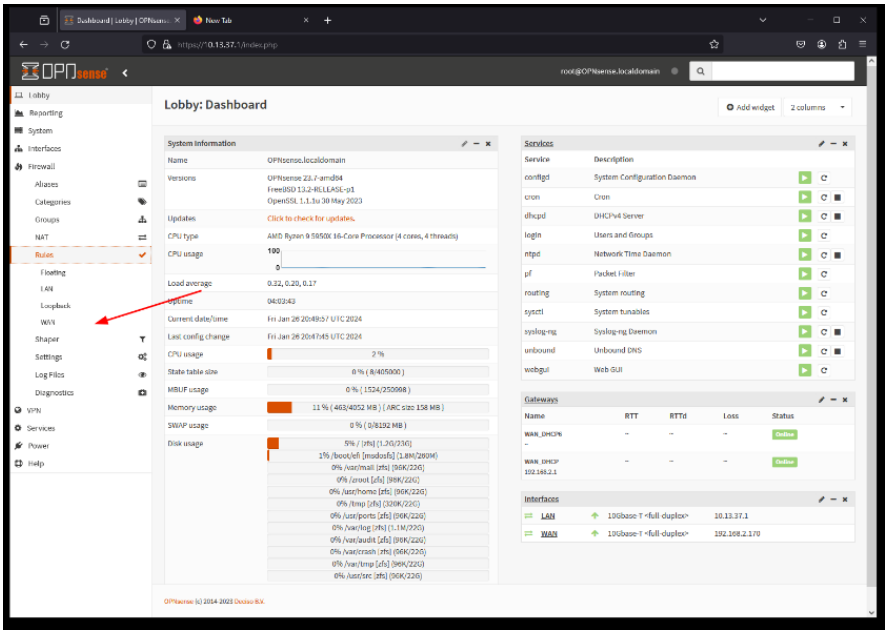


Figure 52 – Dashboard Firewall Rule

27. Add new firewall rules to control inbound and outbound traffic:

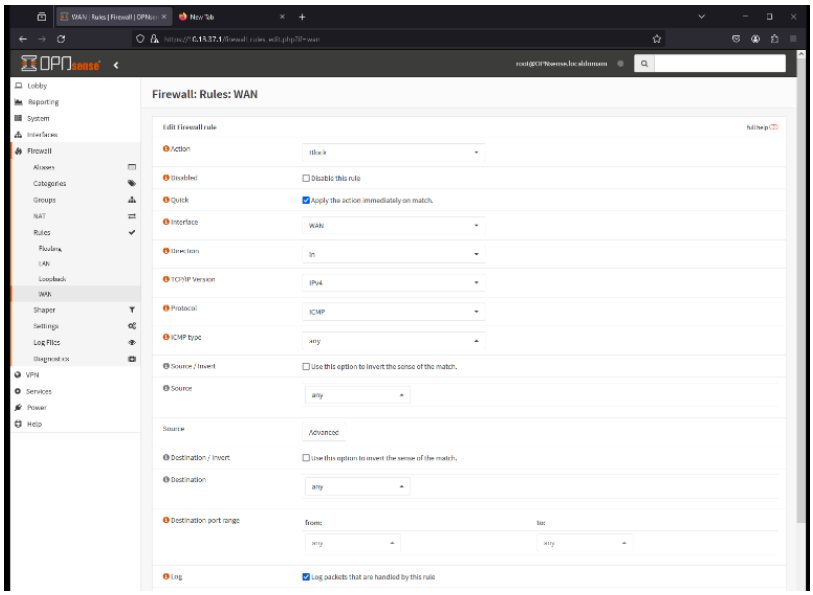


Figure 53 – WAN Firewall Rule

28. In this example, create a WAN firewall rule to BLOCK inbound ICMP. Click the **Log Packets** that are handled by this rule option and provide **Categories** and a description. Then click **Save**:

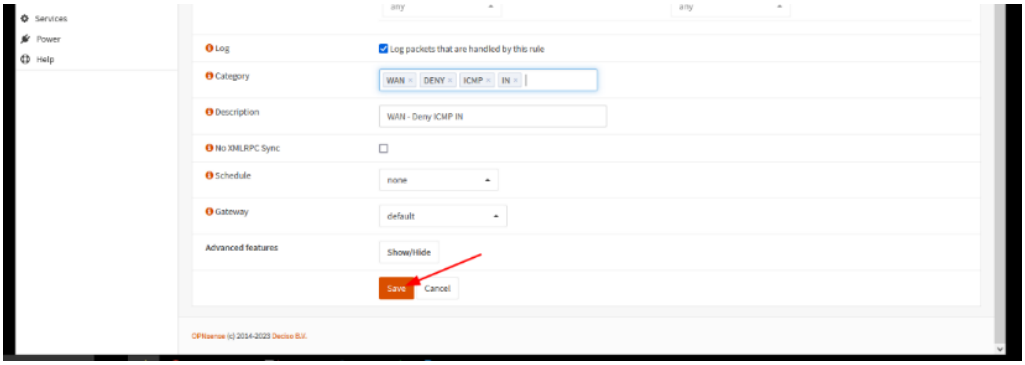


Figure 54 – Saving Firewall Rule

29. Once you save the rule click **Apply Changes**:

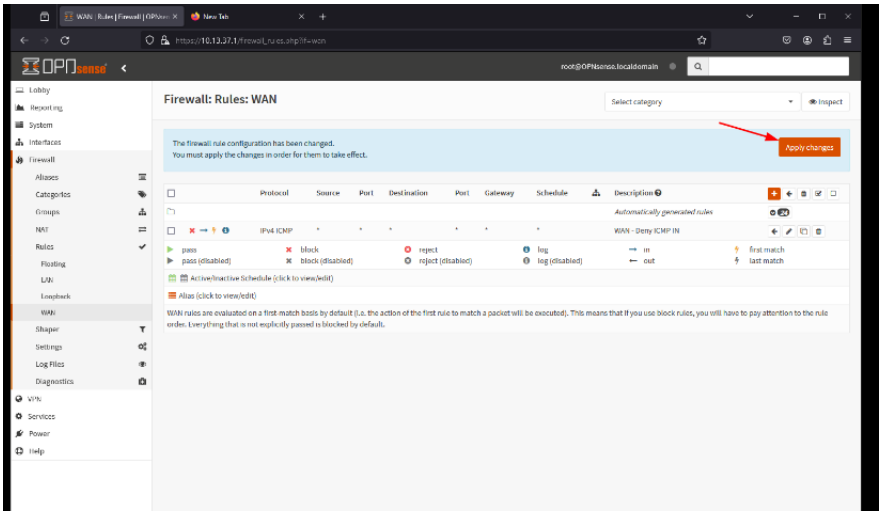
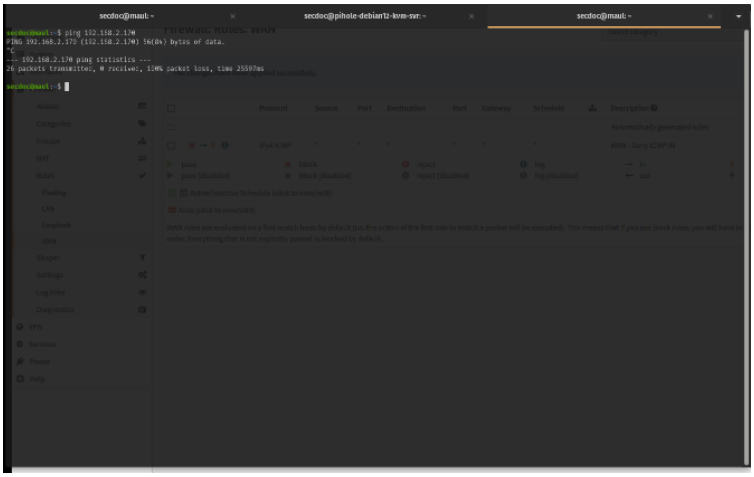
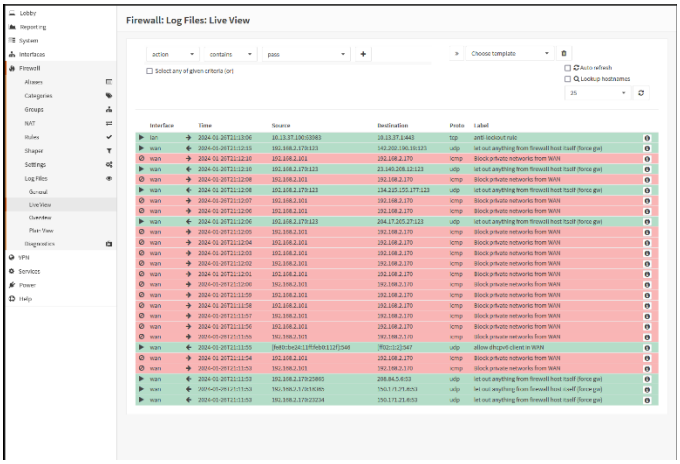


Figure 55 – Applying Changes

30. Next, test the firewall rules by attempting to access resources from various network locations:



(a) Testing Rule



(b) Live View

Figure 56 – Live Firewall Logs

31. Monitor traffic and manage firewall rules regularly.

    One particular note, like pfSense, OPNsense supports various plug-ins or modules such as Snort and Surricata. OPNsense, supports other plug-ins not available within pfSense such as Zenarmor (https://www.zenarmor.com/) which you can install and extend the capabilities of the firewall to include Layer-7 inspection and proxy capabilities.

    Following is a sample screenshot of the dashboard and visibility associated with Zenarmor:

Figure 57 – OPNsense Zenarmor Plugin Dashboard

Let’s move on to the next lab.

Lab 3: SIEM Solution Using Graylog Open in a Virtual Environment

Graylog is an open-source, centralized log management and analysis platform that helps organizations collect, store, and analyze log data from various sources, such as servers, applications, and network devices. Built on top of Elasticsearch, MongoDB, and other open-source technologies, Graylog provides a scalable and flexible solution for managing large volumes of log data in real-time. Its user-friendly web interface allows users to easily search, filter, and visualize log data, enabling them to quickly identify and investigate potential security incidents, performance issues, and other important events. Graylog also offers a powerful alerting system that can notify administrators when specific conditions are met, such as when a critical error occurs or when suspicious activity is detected. With its extensive plugin ecosystem and APIs, Graylog can integrate with a wide range of tools and services, making it a versatile and valuable addition to any organization's monitoring and logging infrastructure. Whether used for security, compliance, or troubleshooting purposes, Graylog empowers organizations to gain valuable insights from their log data and make informed decisions to improve their overall IT operations.

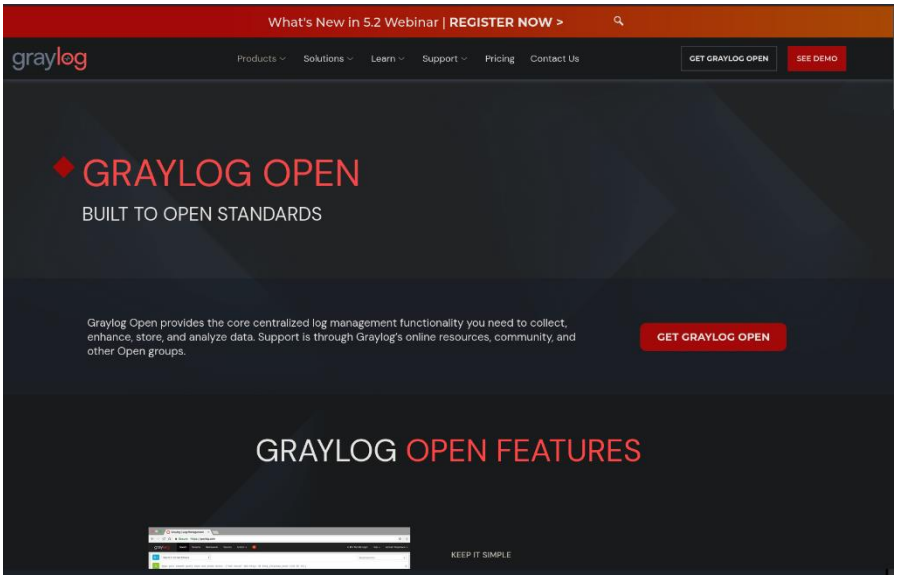


Figure 58 – Graylog Website

The prerequisites are:

- A VM capable of running Graylog (in this lab/example the VM is Ubuntu 22.04 Server with 4-cores and 16GB of RAM and 100GB Harddrive).
- Administrative privileges for software installation
- Java 8 or higher

Let us look at the steps:

1. Create a new VM to host the Graylog server.
2. Allocate sufficient resources based on the expected volume of log data.
3. Set up network adapters to communicate with other VMs (for log sources) and potentially the host machine.
4. For the Graylog installation, install MongoDB and Elasticsearch as Graylog dependencies.

5. Download and install Graylog repository configuration with the package manager or go to https://go2docs.graylog.org/5-2/downloading_and_installing_graylog/ubuntu_installation.html. The lab shows the step-by-step instructions associated with installing Graylog Server 5.2 running Opensearch as referenced on the Graylog Website.

6. Now, let's move to the installation of Graylog. To configure the Graylog server to use a particular time zone, run the following command:

```
sudo timedatectl set-timezone UTC
```

Note

The `mongodb` package included with Ubuntu distributions is not maintained by MongoDB Inc and conflicts with the official `mongodb-org` package. If the `mongodb` package is already installed, you must uninstall it first before proceeding.

7. To install MongoDB Community Edition, import the public key for the package management system, if `gnupg` and `curl` are not already installed, run:

```
sudo apt-get install gnupg curl
```

8. Import the MongoDB public GPG key from https://pgp.mongodb.com/server-7.0.asc using this command:

```
curl -fsSL https://pgp.mongodb.com/server-7.0.asc | \
   sudo gpg -o /usr/share/keyrings/mongodb-server-7.0.gpg \
   --dearmor
```

9. For Ubuntu 22.04 (Jammy), create the repository file at `/etc/apt/sources.list.d/mongodb-org-7.0.list`:

```
echo "deb [ arch=amd64,arm64 signed-
by=/usr/share/keyrings/mongodb-server-7.0.gpg ]
https://repo.mongodb.org/apt/ubuntu jammy/mongodb-
org/7.0 multiverse" | sudo tee
/etc/apt/sources.list.d/mongodb-org-7.0.list
```

10. Reload the local package database - run this command to refresh the local package index:

```
sudo apt-get update
```

11. To install the latest stable version, issue the following:

```
sudo apt-get install -y mongodb-org
```

12. If you need to incorporate proxies or other restricted environments, you can implement a keyserver solution using a widget.

```
wget -qO-
'http://keyserver.ubuntu.com/pks/lookup?op=get&search=0xf56
79a222c647c87527c2f8cb00a0bd1e2c63c11' | sudo apt-key add -
```

13. Configure MongoDB to start automatically when the operating system boots up, and check that the MongoDB service is active and running:

```
sudo systemctl daemon-reload

sudo systemctl enable mongod.service

sudo systemctl restart mongod.service

sudo systemctl --type=service --state=active | grep mongod
```

14. The OpenSearch documentation recommends following their user guides for installation at the following URL: https://opensearch.org/docs/latest/install-and-configure/install-opensearch/debian/.

15. To integrate OpenSearch with Graylog, you can take these steps. This example demonstrates installing OpenSearch using the DEB package. Import the public GPG key to verify that the APT repository is signed:

```
curl -o-
https://artifacts.opensearch.org/publickeys/opensearch.pgp
| sudo gpg --dearmor --batch --yes -o
/usr/share/keyrings/opensearch-keyring
```

16. Create an APT repository for OpenSearch:

```
echo "deb [signed-by=/usr/share/keyrings/opensearch-
keyring]
https://artifacts.opensearch.org/releases/bundle/opensearch
/2.x/apt stable main" | sudo tee
/etc/apt/sources.list.d/opensearch-2.x.list
```

17. Check that the repository was added correctly:

```
sudo apt-get update
```

18. After adding the repository information, display all OpenSearch versions available for installation:

```
sudo apt list -a opensearch
```

19. Select the OpenSearch version to install (the latest version will be installed if a specific version is not specified):

```
sudo apt-get install opensearch
```

20. To configure Graylog for OpenSearch, start by opening the Graylog configuration file in YAML format:

```
sudo nano /etc/opensearch/opensearch.yml
```

21. At minimum, update these configuration fields (for an unsecured single-node setup) to integrate OpenSearch with Graylog:

```
cluster.name: graylog
node.name: ${HOSTNAME}
path.data: /var/lib/opensearch
path.logs: /var/log/opensearch
```

You can see these in the following image:

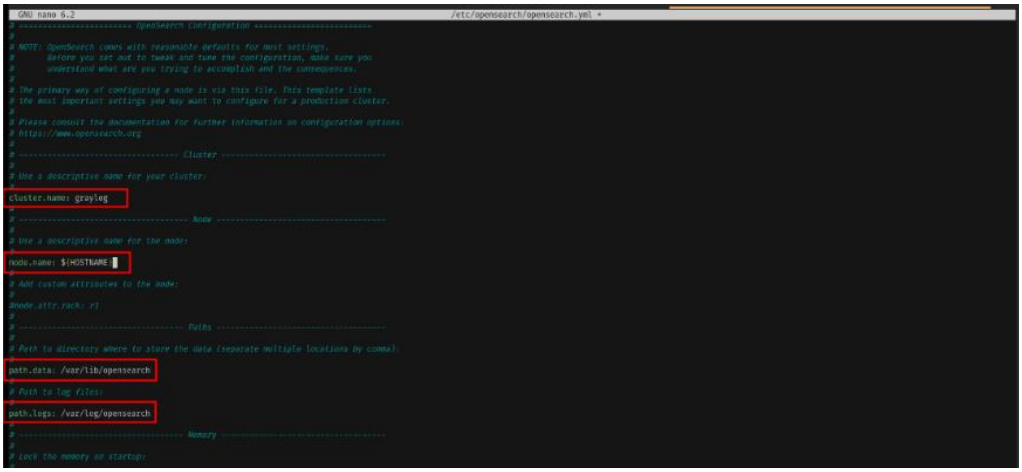


Figure 59 – Opensearch YAML Configuration – Parameters Part 1

Updating the OpenSearch YAML file is crucial when setting up Graylog to ensure seamless integration and optimal performance. The YAML file contains essential configuration settings that define how OpenSearch, a powerful open-source search and analytics engine, interacts with Graylog. By modifying this file, you can customize various aspects of OpenSearch's behavior, such as cluster settings, node roles, and memory allocation. Proper configuration of the OpenSearch YAML file is necessary to ensure that Graylog can efficiently index, store, and retrieve log data, enabling fast and accurate search results. Additionally, updating the YAML file allows you to enable important features like authentication, encryption, and data replication, which are critical for securing your log data and ensuring high availability. Failing to update the OpenSearch YAML file correctly can lead to suboptimal performance, compatibility issues, and even data loss. Therefore, taking the time to carefully review and modify the YAML file is an essential step in setting up a robust and reliable Graylog implementation:

```
discovery.type: single-node

network.host: 0.0.0.0

Http.port: 9200
```

These are visible in the following image:

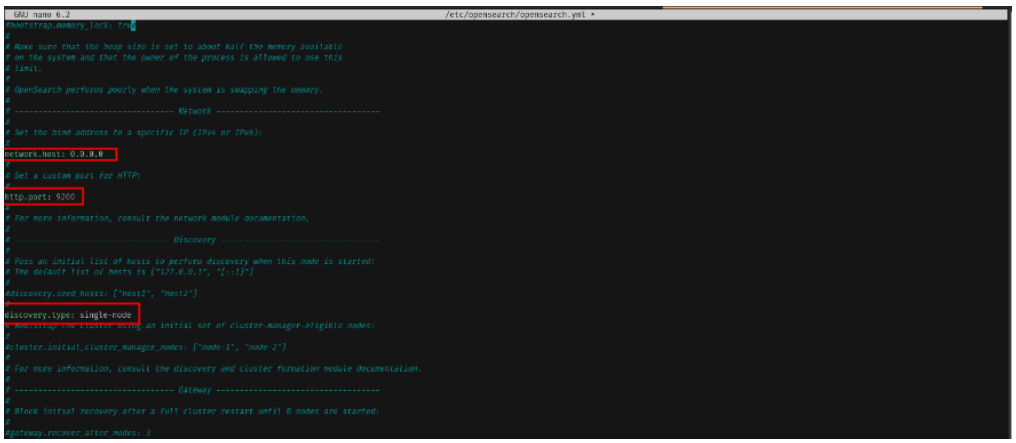


Figure 60 – Opensearch YAML Configuration – Parameters Part 2

By setting `action.auto_create_index: false` in the OpenSearch YAML configuration, you enhance security, optimize performance, ensure compatibility with Graylog's index management strategies, and promote better governance over your log data indices:

```
action.auto_create_index: false

plugins.security.disabled: true
```

It's important to note that disabling the OpenSearch security plugin does not mean that your log data is unprotected. Graylog's built-in security features, combined with proper network isolation and access controls, can still provide a robust security framework for your log management system.

However, if your organization has specific security requirements or compliance regulations that mandate the use of the OpenSearch security plugin, you may need to keep it enabled and configure it accordingly. In such cases, you should refer to the Graylog documentation and OpenSearch security plugin documentation to ensure proper integration and configuration:

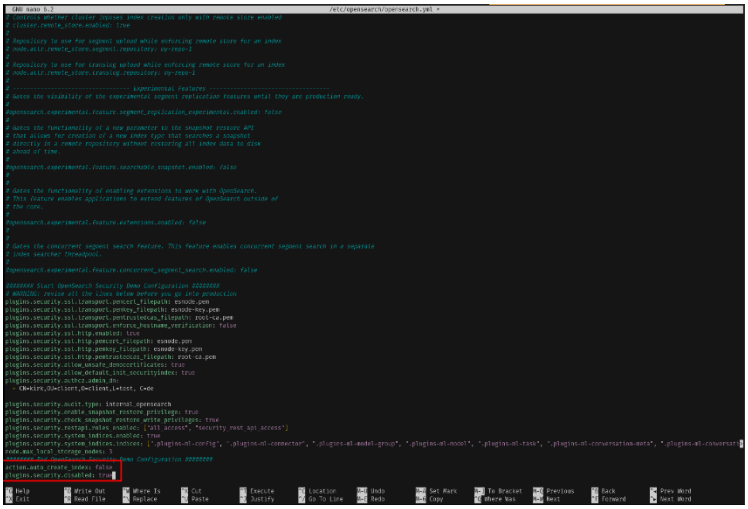


Figure 61 – Opensearch YAML Configuration – Parameters Part 3

22. Once all updates are made to the `opensearch.yaml` file, press *CTRL+O* and press *Enter* and then exit by pressing *CTRL+X*:

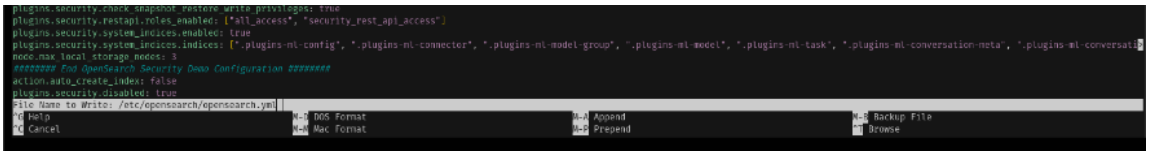


Figure 62 – Saving Configuration

23. Use these commands to add the Graylog package repository and install Graylog:

```
wget https://packages.graylog2.org/repo/packages/graylog-5.2-repository_latest.deb

sudo dpkg -i graylog-5.2-repository_latest.deb

sudo apt-get update && sudo apt-get install graylog-server
```

Adding the Graylog package repository and installing Graylog from it is crucial to ensure a smooth and secure setup of your log management system. By using the official Graylog package repository, you gain access to the latest stable version of Graylog, which is thoroughly tested and optimized for production

environments. The package repository also simplifies the installation process by handling dependencies and providing automatic updates, ensuring that your Graylog instance remains up-to-date with the latest features, bug fixes, and security patches. This streamlined approach reduces the risk of compatibility issues and vulnerabilities that may arise from manual installations or using outdated packages. Moreover, the Graylog package repository provides a consistent and reliable way to deploy Graylog across multiple systems or environments, making it easier to maintain and scale your log management infrastructure. By leveraging the package repository, you can focus on configuring and using Graylog to gain valuable insights from your log data, rather than worrying about the intricacies of the installation process. Overall, adding the Graylog package repository and installing Graylog from it is a best practice that ensures a robust, secure, and maintainable log management solution:

Figure 63 – Installing Graylog

24. Review the instructions in the Graylog server configuration file at `/etc/graylog/server/server.conf` and modify as necessary. You must also add values for `password_secret` and `root_password_sha2` in this file, as Graylog will fail to start if these parameters are missing.

    Generate a password_secret value by running this command:

```
< /dev/urandom tr -dc A-Z-a-z-0-9 | head -c${1:-96};echo;
```

    Reviewing and modifying the Graylog server configuration file (`/etc/graylog/server/server.conf`) is essential to ensure that your Graylog instance is set up correctly and securely. The configuration file contains various settings that control the behavior and performance of your Graylog server, such as network interfaces, authentication mechanisms, and data retention policies. By carefully reviewing and adjusting these settings, you can tailor Graylog to your specific requirements and optimize its performance for your environment.

    One of the most critical aspects of configuring Graylog is setting the `password_secret` and `root_password_sha2` parameters. These parameters are essential for securing your Graylog installation and protecting sensitive data:

    A. `password_secret`: This is a randomly generated secret string used for securely encrypting and decrypting sensitive data, such as user passwords and access tokens. Without a valid `password_secret`, Graylog will not be able to start, as it cannot ensure the security of stored data. Generating a strong, random `password_secret` is crucial to prevent unauthorized access and protect your log data from potential breaches.

    B. `root_password_sha2`: This parameter sets the SHA-2 hash of the root user's password. The root user is a built-in administrative account with full access to the Graylog system. By setting a strong, hashed password for the root user, you prevent unauthorized access to your Graylog instance and ensure that only authorized personnel can manage the system.

To generate a secure `password_secret` value, you can use the command:

```
< /dev/urandom tr -dc A-Z-a-z-0-9 | head -c${1:-96};echo;
```

This command reads random data from the `/dev/urandom` device, filters out only alphanumeric characters, and generates a 96-character random string. Using a randomly generated secret helps ensure the strength and uniqueness of the `password_secret`, making it much harder for attackers to guess or crack.

In summary, reviewing and modifying the Graylog server configuration file, particularly setting the `password_secret` and `root_password_sha2` parameters, is critical for securing your Graylog installation. By generating a strong, random `password_secret` and setting a hashed password for the root user, you protect your log data, prevent unauthorized access, and ensure the overall integrity of your log management system. Neglecting to set these parameters or using weak values can leave your Graylog instance vulnerable to security risks and compromise the confidentiality and reliability of your log data:

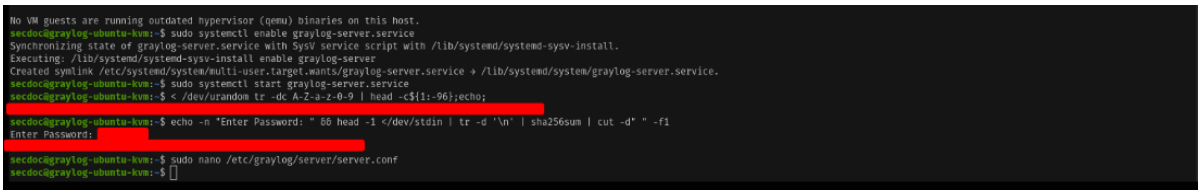


Figure 64 – Graylog Password Secret

25. Then generate the `root_password_sha2` hash using this command:

```
echo -n "Enter Password: " && head -1 </dev/stdin | tr -d
'\n' | sha256sum | cut -d" " -f1
```

26. Once both hash values are generated, save these to a text file so they can be placed within the `/etc/graylog/server/server.conf` file.

    Update and uncomment the file for the following configuration items:

```
password_secret
```

```
root_password_sha2
```

Saving the generated hash values for `password_secret` and `root_password_sha2` to a text file before placing them in the `/etc/graylog/server/server.conf` file is important for several reasons:

A. **Secure Storage**: Storing the hash values in a separate text file allows you to keep them in a secure location, such as an encrypted disk or a password manager. This is particularly important for the `password_secret`, as it is used to encrypt sensitive data in your Graylog instance. By storing the hash values separately, you reduce the risk of accidentally exposing them to unauthorized users or committing them to version control systems.

B. **Backup and Recovery**: Having a separate text file with the hash values serves as a backup in case the `/etc/graylog/server/server.conf` file becomes corrupted, accidentally modified, or lost. If you need to restore your Graylog configuration or migrate to a new server, having the hash values readily available in a text file will make the process faster and easier.

C. **Auditing and Documentation**: Storing the hash values in a text file creates a record of the values used for your Graylog installation. This can be useful for auditing purposes, as you can verify that the correct hash values were used and that they haven't been changed unexpectedly. It also serves as documentation for future reference, making it easier for you or other administrators to understand and maintain the Graylog setup.

After saving the hash values to a text file, you need to update and uncomment the corresponding configuration items in the `/etc/graylog/server/server.conf` file:

A. `password_secret`: Locate the `password_secret` setting in the configuration file and uncomment it by removing the `#` symbol at the beginning of the line. Paste the generated `password_secret` hash value

from your text file after the = sign. This setting is critical for Graylog to securely encrypt and decrypt sensitive data.

B. `root_password_sha2`: Find the `root_password_sha2` setting in the configuration file and uncomment it by removing the # symbol at the beginning of the line. Paste the generated `root_password_sha2` hash value from your text file after the = sign. This setting ensures that the root user's password is securely stored and validated.

By updating and uncommenting these configuration items with the hash values from your text file, you are enabling the security features essential for protecting your Graylog installation. Failure to set these values correctly may result in Graylog failing to start or leaving your system vulnerable to unauthorized access.

In summary, saving the generated hash values to a text file before placing them in the Graylog server configuration file is a best practice that promotes secure storage, backup and recovery, and auditing. Updating and uncommenting the `password_secret` and `root_password_sha2` settings with the correct hash values is crucial for ensuring the security and proper functioning of your Graylog instance:

Figure 65 – Update Graylog Server Configuration with Secret

The `http_bind_address` setting in the Graylog server configuration file (`/etc/graylog/server/server.conf`) is used to specify the IP address and port on which the Graylog web interface will listen for incoming HTTP connections. This setting determines how users and other systems can access the Graylog web interface.

It's important to carefully consider the `http_bind_address` setting and configure it according to your specific requirements. Setting it incorrectly or exposing the Graylog web interface to untrusted networks can pose security risks and allow unauthorized access to your log management system:

```
http_bind_address
```

The `http_bind_address` setting in the Graylog server configuration determines the IP address and port on which the Graylog web interface listens for incoming HTTP connections. It plays a critical role in controlling access to the web interface and should be configured in alignment with your network setup and security requirements. By properly setting the `http_bind_address`,

you can ensure that the Graylog web interface is accessible to authorized users and systems while minimizing the risk of unauthorized access:

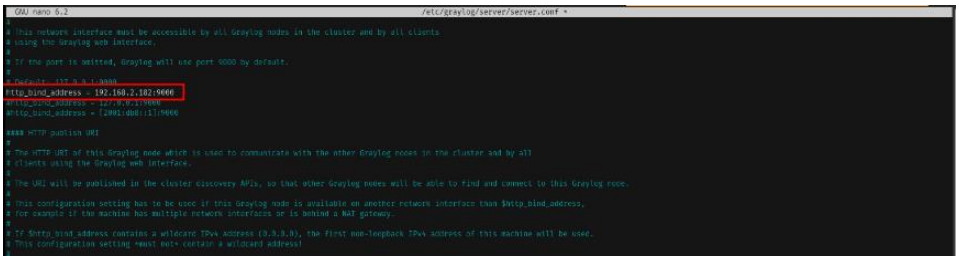


Figure 66 – Update Graylog Server Configuration – Binding IP Address

The configuration file for Graylog still references Elasticsearch settings like `elasticsearch_hosts`, `elasticsearch_index_prefix`, `elasticsearch_analyzer`, `elasticsearch_shards`, and `elasticsearch_replicas` because Graylog was originally designed to work with Elasticsearch as its backend storage and search engine. However, OpenSearch has emerged as a popular alternative to Elasticsearch, and Graylog now supports using OpenSearch as well:

```
rotation_strategy

elasticsearch_max_docs_per_index
```

Together, the `rotation_strategy` and `elasticsearch_max_docs_per_index` settings help you manage the lifecycle of Elasticsearch indices in Graylog. They allow you to control how data is organized, rotated, and retained over time, ensuring optimal performance and storage efficiency:

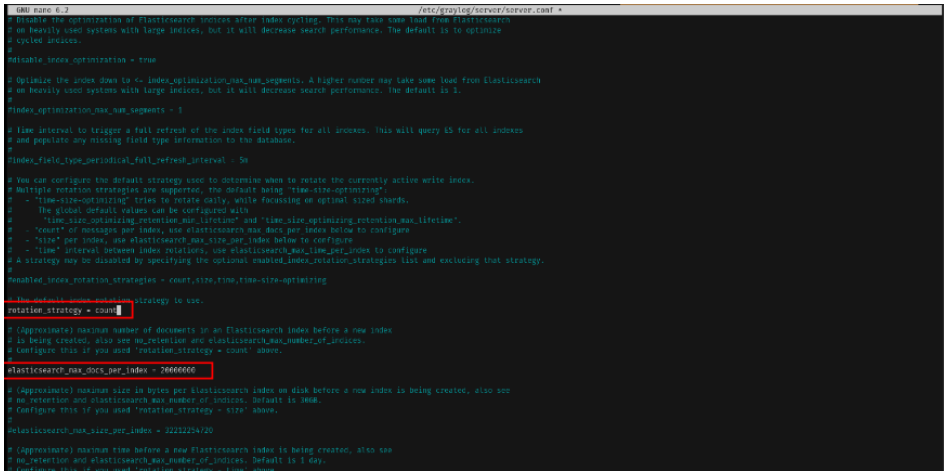


Figure 67 – Update Graylog Server Configuration – Index and Rotation

The `elasticsearch_hosts` setting in the Graylog server configuration file (`/etc/graylog/server/server.conf`) is used to specify the connection details for one or more Elasticsearch nodes that Graylog will use for storing and searching log data. Elasticsearch is a distributed search and analytics engine that serves as the backend storage for Graylog:

```
elasticsearch_hosts
```

It's crucial to ensure that the `elasticsearch_hosts` setting is properly configured to point to the correct Elasticsearch (opensearch) node(s) and that the specified nodes are accessible from the Graylog server. Misconfiguration or connectivity issues between Graylog and Elasticsearch can prevent Graylog from storing and searching log data effectively:

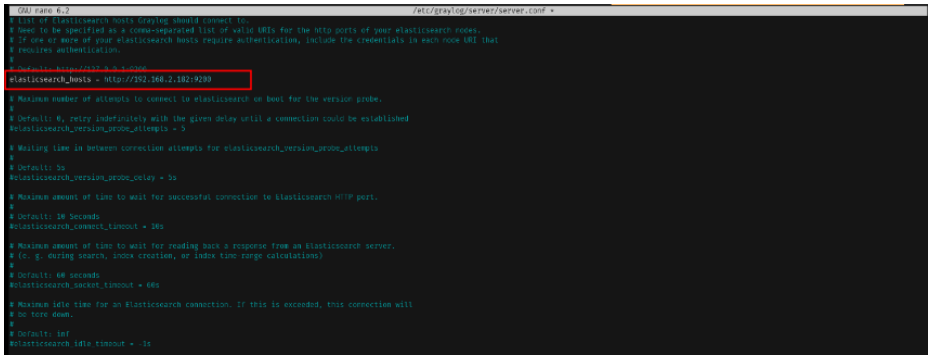


Figure 68 – Update Graylog Server Configuration – Elasticserach (opensearch) Hosts

The `elasticsearch_index_prefix`, `elasticsearch_analyzer`, `elasticsearch_shards`, and `elasticsearch_replicas` settings in the Graylog server configuration file (`/etc/graylog/server/server.conf`) are used to control various aspects of how Graylog interacts with Elasticsearch and manages the indexing and storage of log data:

```
elasticsearch_index_prefix

elasticsearch_analyzer

elasticsearch_shards

elasticsearch_replicas
```

When configuring these settings, consider your specific requirements, data volume, performance needs, and Elasticsearch cluster setup. It's recommended to consult the Elasticsearch documentation and best practices to make informed decisions based on your deployment scenario.

By properly tuning the `elasticsearch_index_prefix`, `elasticsearch_analyzer`, `elasticsearch_shards`, and `elasticsearch_replicas` settings, you can optimize how Graylog interacts with Elasticsearch, ensuring efficient indexing, searching, and storage of log data while meeting your performance and availability goals:

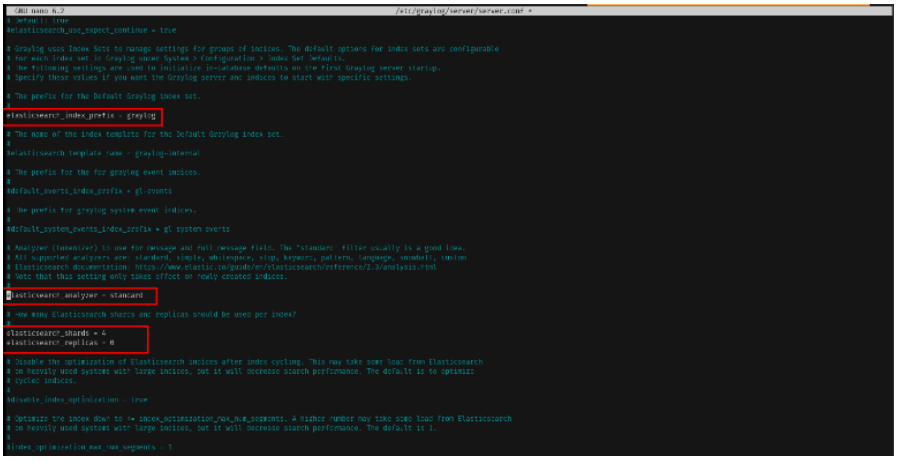


Figure 69 – Update Graylog Server Configuration – Elasticserach (opensearch) configuration

The `retention_strategy` and `retention_strategy` settings in the Graylog server configuration file (`/etc/graylog/server/server.conf`) are used to define how Graylog handles the retention and deletion of old log data. These settings allow you to control the lifecycle of your log data, ensuring that your storage space is efficiently utilized and that you retain log data for the desired duration:

```
retention_strategy

retention_period
```

By carefully configuring the `retention_strategy` and `elasticsearch_retention_period` settings, you can effectively manage the lifecycle of your log data in Graylog, ensuring that you retain data for the necessary duration while optimizing storage utilization and performance:

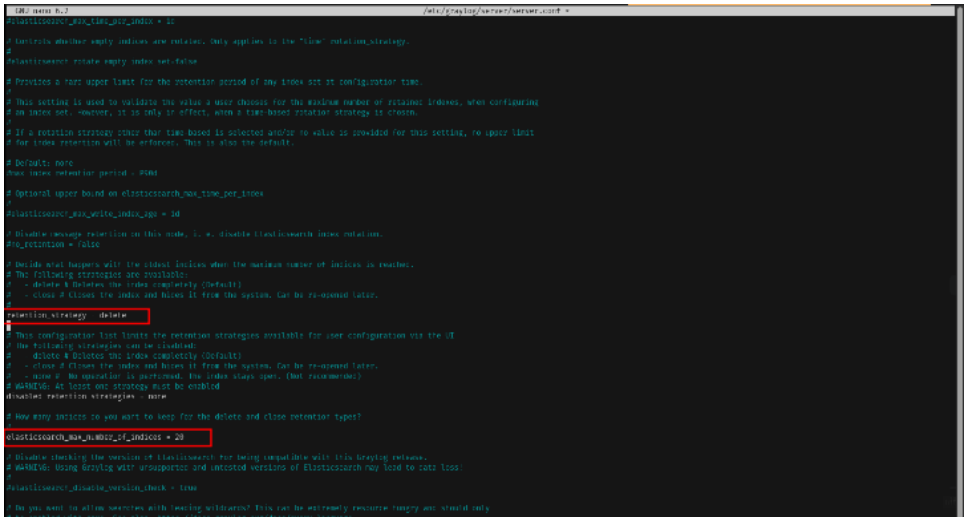


Figure 70 – Update Graylog Server Configuration – Elasticserach (opensearch) Retention Configuration

Running the command `sudo systemctl daemon-reload` after making updates to the Graylog server configuration is important because it ensures that the systemd manager is aware of the changes made to the configuration files:

```
sudo systemctl daemon-reload
```

Note that Graylog does not start automatically after installation.

27. To configure Graylog to start on system boot, run these commands:

```
    sudo systemctl enable graylog-server.service

    sudo systemctl start graylog-server.service

sudo systemctl --type=service --state=active | grep graylog
```

Let's go through each command and explain its importance:

A. `sudo systemctl enable graylog-server.service`: This command enables the Graylog server service to start automatically at system boot.

B. `sudo systemctl start graylog-server.service`: This command starts the Graylog server service immediately.

C. `sudo systemctl --type=service --state=active | grep graylog`: This command is used to verify the status of the Graylog server service. It lists all the active systemd services and filters the output to include only the lines containing the word "graylog".

The combination of these commands ensures that the Graylog server service is properly enabled, started, and verified:

If there are any issues with the installation or configuration, starting the service may result in errors or failures, which would need to be addressed separately:

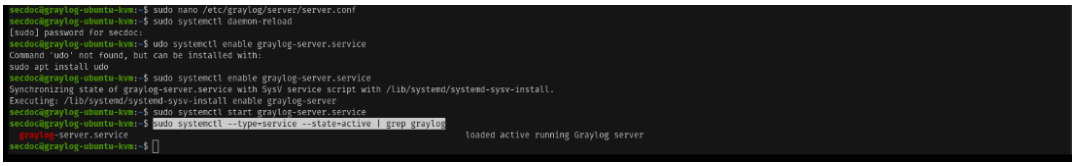


Figure 71 – Restart Graylog Service

28. Now, onto accessing web interface. Open the Graylog web interface by navigating to http://[your_graylog_ip]:9000:

Figure 72 – Graylog Web Login Screen

Once your Graylog instance or cluster is running, you can access the web interface for searching and analyzing indexed data and managing your Graylog

configuration. By default, the interface is available at https://<graylog-server>:9000/.

29. If you run into the issue where the web interface is not loading, run the command `netstat -nl` within the terminal on the Ubuntu server to make sure that the system is listening on port `9000`:

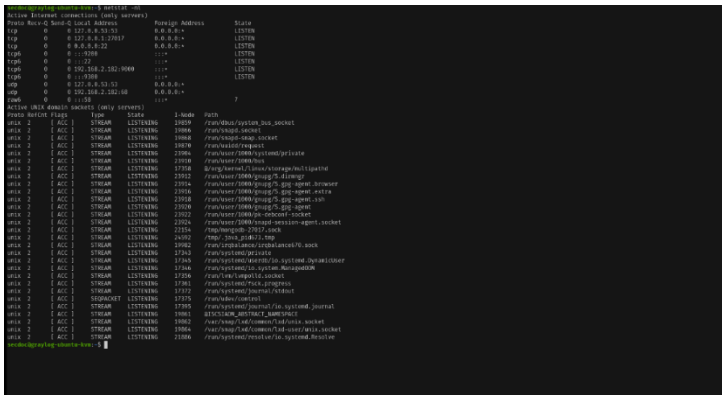


Figure 73 – Netstat System Listening Ports

30. To log into the web interface, open a browser and navigate to https://<ip-address>:9000, substituting your Graylog server's IP address.

31. Sign in as an admin user and enter the password secret set during Graylog installation:

Figure 74 – Login to Graylog

32. After logging in, you will be brought to the initial welcome screen:

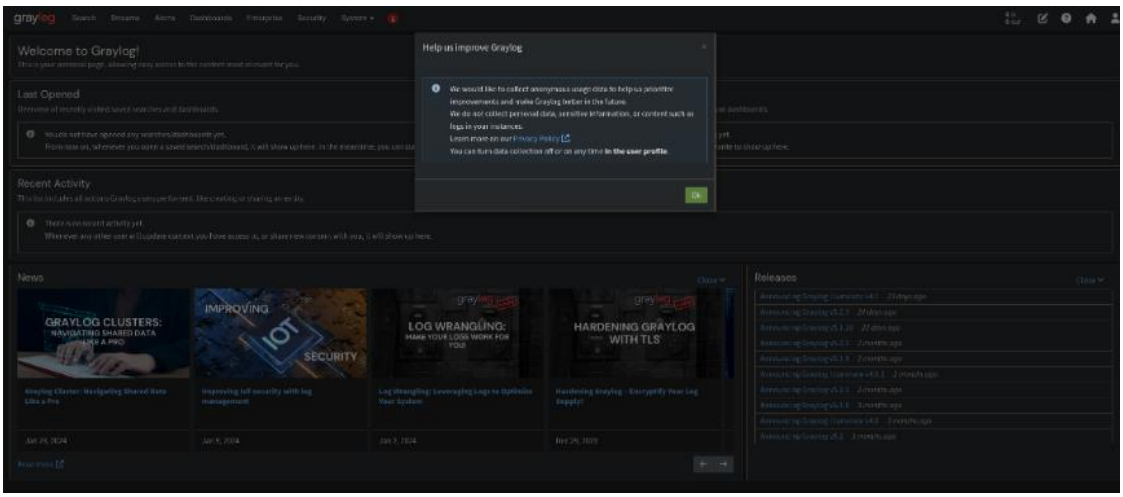


Figure 75 – Graylog Welcome Screen

Now that the hard part is over, you can get started with other aspects of getting data into Graylog:

- **Input configuration**: Set up an input in Graylog to receive data. For example, create a Syslog UDP input to listen for incoming logs.
- **System integration**: Configure systems to send logs to Graylog. For Snort, you may set up Barnyard2 to forward Snort logs to Graylog or syslog from OPNsense.
- **Creating dashboards**: Within the Graylog interface, create dashboards to visualize the incoming log data and analyze events.
- **Alert configuration**: Set up alerts in Graylog to notify you of potential security incidents based on the logs.

I have placed sample content packs for Graylog in my Github page which can be found at https://github.com/secdoc, which can be a starting point and enable you further in getting Graylog configured. This includes dashboards, inputs and pipelines. Here is the example of pipeline JSON:

```
{
```

```
      "v": "1",
      "type": {
        "name": "pipeline_rule",
        "version": "1"
      },
      "id": "d983c394-563a-4d22-b52b-f77f8553ea56",
      "data": {
        "title": {
          "@type": "string",
          "@value": "src-ip threat intel"
        },
        "description": {
          "@type": "string",
          "@value": "src-ip threat intel"
        },
        "source": {
          "@type": "string",
          "@value": "rule \"src-ip threat intel\"\nwhen\n
has_field(\"nf_src_address\") && !
in_private_net(to_string($message.src_ip))  \nthen\nlet
src_addr_intel =
threat_intel_lookup_ip(to_string($message.nf_src_address),
\"nf_src_address\");\nset_fields(src_addr_intel);\n\nlet
dns_question_intel =
threat_intel_lookup_domain(to_string($message.dns_question)
,
\"dns_question\");\nset_fields(dns_question_intel);\n\nlet
whois_intel =
whois_lookup_ip(to_string($message.nf_src_address),
\"nf_src_addressr\");\nset_fields(whois_intel);\nend"
```

```
            }

        },
```

Each lab session should be followed by a cleanup procedure to reset the environment if needed. Additionally, by running these labs in virtual environments, you not only ensure a controlled and replicable setup for each lab iteration but also provide an opportunity to simulate a more realistic network environment with multiple interacting systems. This approach offers a safer and more scalable method for cybersecurity training and experimentation.

By combining controls for prevention, detection, analysis, and alerting, network security tools provide pervasive visibility and protection across environments. Architects must carefully evaluate options to balance risk coverage and TCO.

Endpoint protection tools

Let us look at the labs.

Lab 1: Antivirus Software Implementation Using ClamAV

ClamAV is a popular open-source antivirus software designed to detect and prevent malware infections on various operating systems, including Linux, Windows, and macOS. Developed by Cisco Talos, ClamAV provides a comprehensive toolkit for scanning files, email attachments, and web traffic for viruses, trojans, malware, and other security threats. Its versatility and reliability have made it a go-to solution for system administrators, security professionals, and individuals seeking to protect their systems from malicious software. ClamAV offers both command-line and graphical user interfaces, making it accessible to users with different levels of technical expertise. It features regular database updates to ensure protection against the latest threats, and its modular architecture allows for integration with other security tools and platforms. Whether used as a standalone antivirus solution or as part of a larger security infrastructure, ClamAV plays a crucial role in maintaining the integrity and security of computer systems in today's threat landscape.

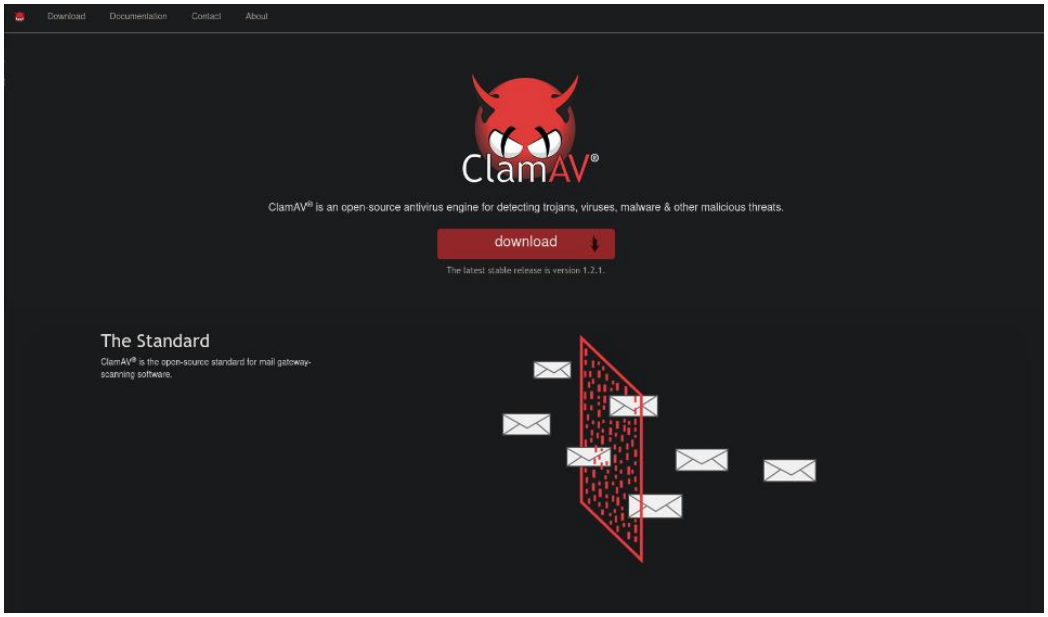


Figure 76 – ClamAV Website

You can access this site at https://www.clamav.net/.

ClamAV is an open-source antivirus engine for detecting trojans, viruses, malware, and other malicious threats.

The prerequisites include:

- A **virtual machine** (**VM**) running a Linux distribution (e.g., Ubuntu Desktop for a GUI).
- Internet access for downloading software.
- Administrative privileges within the VM.

If you have followed the lab in *Chapter 2* for the installation of ClamAV, this lab can be skipped, but if you bypassed the lab, go back to *Chapter 2* and follow the instructions.

Lab 2: Endpoint Detection and Response (EDR) Solution Implementation Using Wazuh

Wazuh is a powerful and open-source security monitoring solution that provides threat detection, integrity monitoring, and incident response capabilities for a wide range of operating systems and platforms. It is designed to help organizations protect their infrastructure from security threats, detect intrusions, and ensure compliance with security policies and regulations. Wazuh combines the benefits of a **host-based intrusion detection system** (**HIDS**) and a **security information and event management** (**SIEM**) solution, offering a comprehensive and centralized approach to security monitoring. With its agent-based architecture, Wazuh collects and analyzes security data from multiple sources, including log files, system events, and network traffic, to identify potential security issues and anomalies. It utilizes a rule-based approach and machine learning algorithms to detect threats in real-time and generate alerts for further investigation. Wazuh also provides a web-based user interface for managing and monitoring the security status of the entire infrastructure, making it easier for security teams to respond to incidents and maintain a strong security posture. Its open-source nature and active community support make Wazuh a cost-effective and flexible solution for organizations of all sizes looking to enhance their security monitoring capabilities.

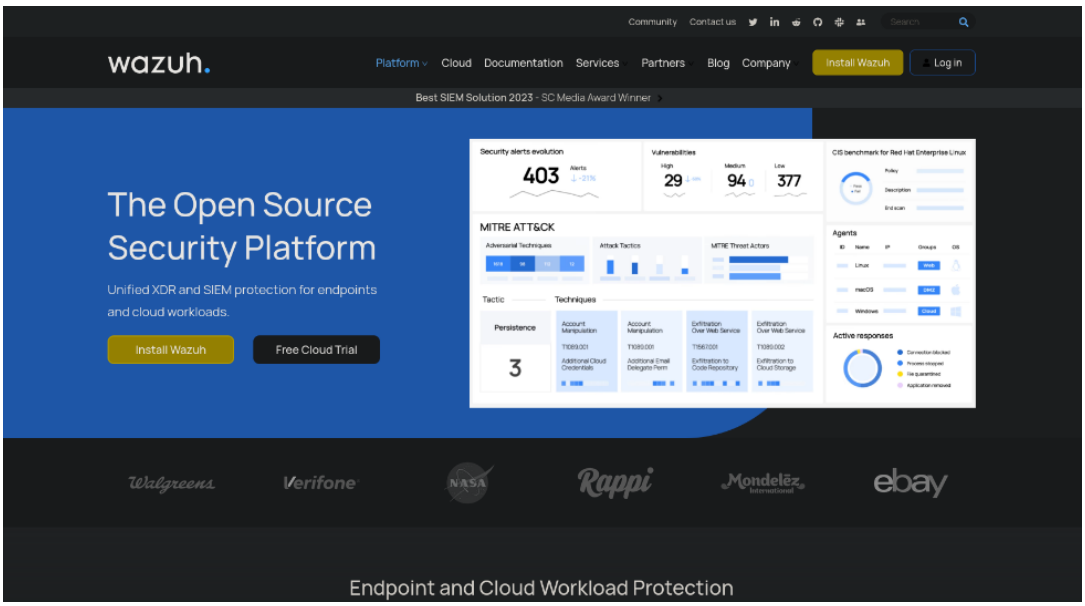


Figure 77 – Wazuh Website

Wazuh is a free, open-source EDR solution that provides host-based intrusion detection, system monitoring, and incident response.

The prerequisites are:

- Based on the Wazuh website (https://documentation.wazuh.com/current/quickstart.html), following is the requirements based on the number of agents that will be deployed:

Agents	CPU	RAM	Storage (90 days)
1-25	4 vCPU	8 GiB	50 GB
25-50	8 vCPU	8 GiB	100 GB
50-100	8 vCPU	8 GiB	200 GB

Table 1 – Wazuh System Requirements

- A VM with a Linux distribution.
- Internet access for downloading software, https://wazuh.com/.
- Administrative privileges within the VM.

Let us look at the steps:

1. Create a new VM to serve as the Wazuh server, indexer and dashboard. Optionally, create additional VMs to act as Wazuh agents.
2. Download and run the Wazuh installation assistant:

```
curl -sO https://packages.wazuh.com/4.7/wazuh-install.sh &&
sudo bash ./wazuh-install.sh -a
```

By following these steps, you create a dedicated VM to host the Wazuh server components and optionally set up additional VMs as Wazuh agents. The installation assistant simplifies the installation process by automating the necessary tasks and configurations.

After completing these steps, you will have a functional Wazuh security monitoring system in place. The Wazuh server will be ready to receive and analyze security data from the agents, and you can access the web-based dashboard to monitor and investigate security events in your infrastructure.

Remember to configure the Wazuh agents on the systems you want to monitor and ensure proper network connectivity between the agents and the Wazuh server for seamless data collection and analysis:

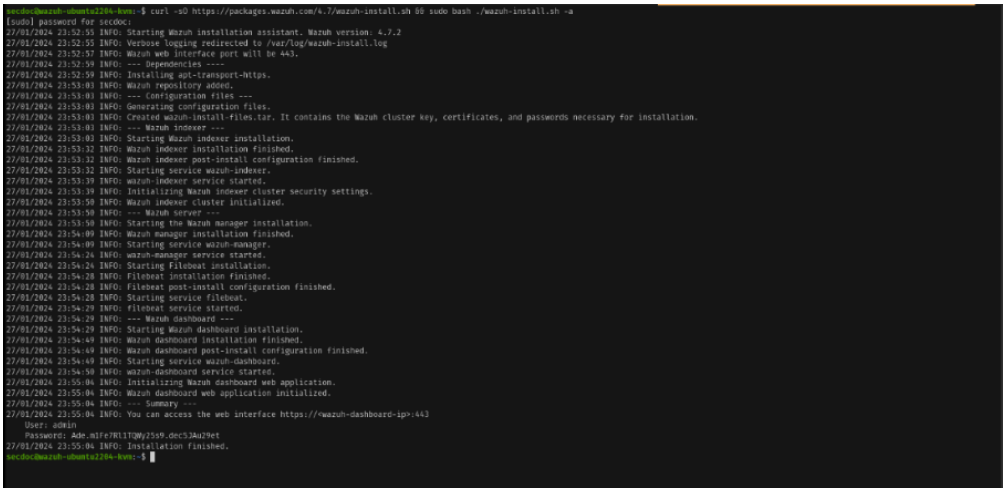


Figure 78 – Wazuh Installation

After the assistant completes the installation, it prints the access credentials and a confirmation message indicating the installation succeeded.

3. Access the Wazuh web interface at https://<wazuh-dashboard-ip> using these credentials:

   Username: admin

   Password: <ADMIN_PASSWORD> (the password provided in the terminal after a successful login)

On first login, the browser may display a warning that the certificate is untrusted since it was not issued by a known authority. You can accept the certificate as an exception or replace it with a trusted certificate.

4. The passwords for the Wazuh indexer and API users are stored in the `wazuh-passwords.txt` file inside the `wazuh-install-files.tar` archive. To display them, extract and print the file:

```
tar xvf wazuh-install-files.tar wazuh-passwords.txt

cat wazuh-passwords.txt
```

5. To uninstall the Wazuh central components, run the Wazuh installation assistant again with the `-u` or `--uninstall` option:

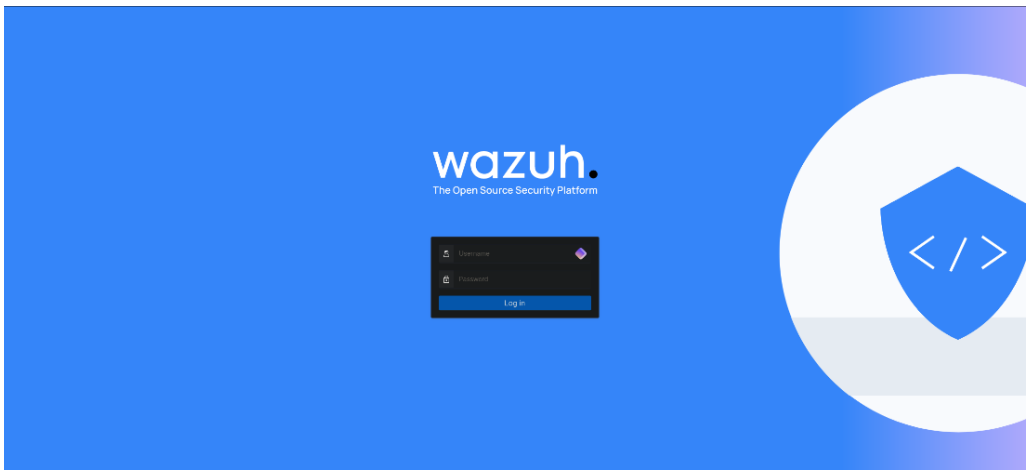


Figure 79 – Wazuh Login Screen

Now that your Wazuh installation is ready, you can start deploying the Wazuh agent. This can be used to protect laptops, desktops, servers, cloud instances, containers, or virtual machines. The agent is lightweight and multi-purpose, providing a variety of security capabilities.

Instructions on how to deploy the Wazuh agent can be found in the Wazuh web user interface:

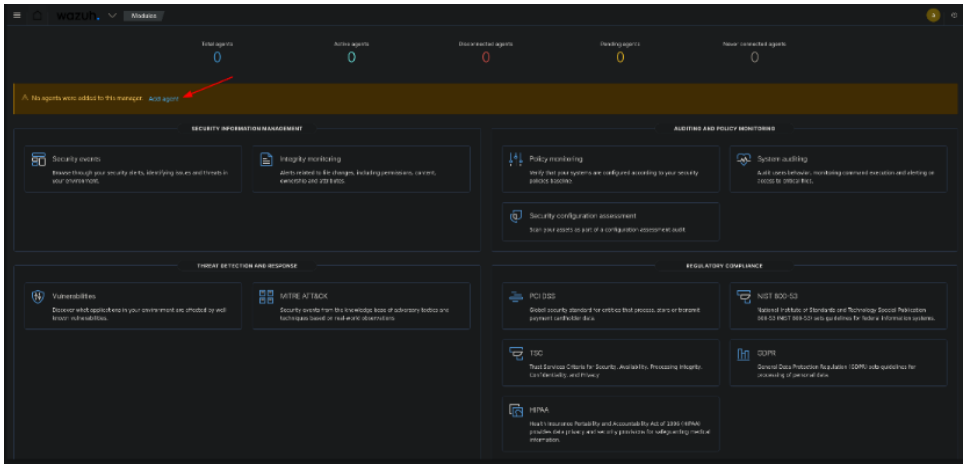


Figure 80 – Wazuh Dashboard

6. On a different VM, install the Wazuh agent by following similar steps or using a pre-built package for the specific OS:

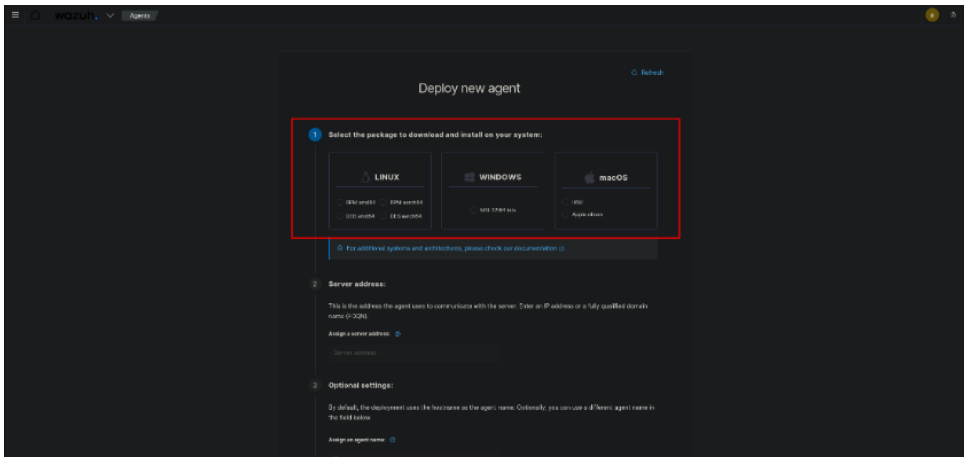


Figure 81 – Deploy Wazuh Agent

7. In this lab deploy the agent on a Windows 10 system.

8. Now, let us look at agent registration. Select the desired target agent operating system:

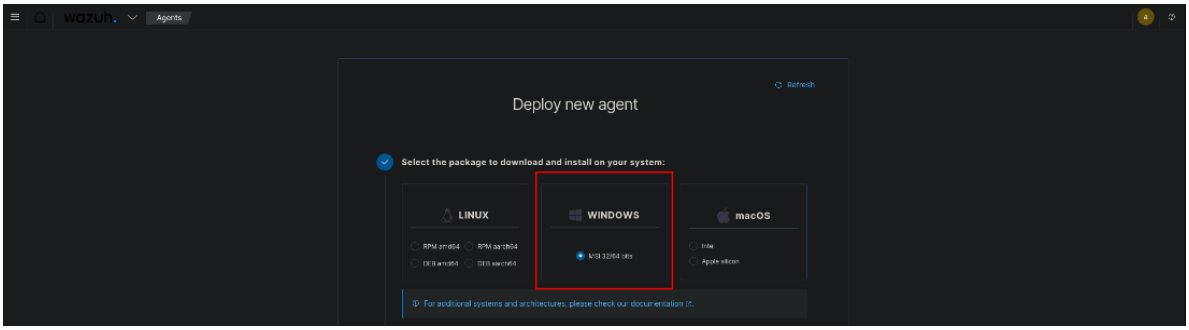


Figure 82 – Wazuh Windows Agent

9. Provide the Wazuh server address the agent uses to communicate with the server. Enter an IP address or a **fully qualified domain name** (**FDQN**):

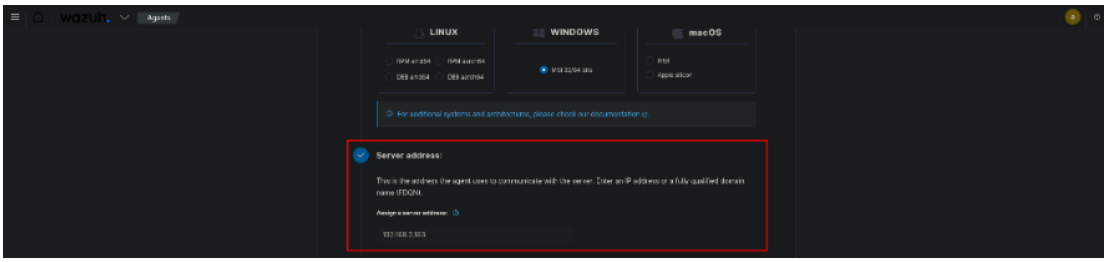


Figure 83 – Wazuh Agent Configuration – Server Address

10. By default, the deployment uses the hostname as the agent name. Optionally, you can use a different agent name in the field:

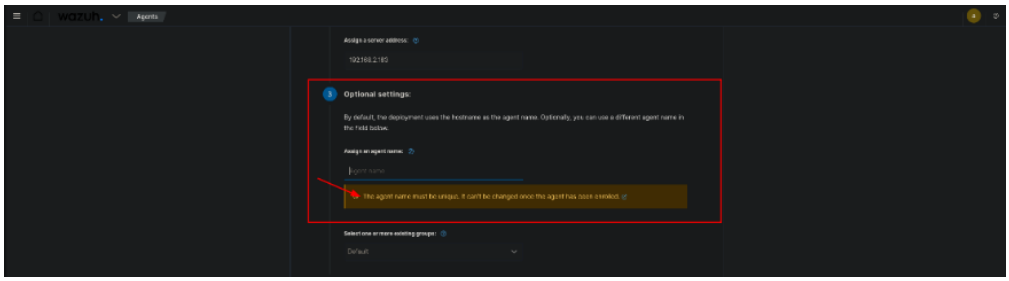


Figure 84 – Wazuh Agent Configuration – Assign Agent Name

11. To download and install the Wazuh agent on Windows:

```
$uri = "https://packages.wazuh.com/4.x/windows/wazuh-agent-4.7.2-1.msi"

$outputPath = "$env:temp\wazuh-agent.msi"


Invoke-WebRequest -Uri $uri -OutFile $outputPath


msiexec.exe /i $outputPath /q `
    WAZUH_MANAGER="<WAZUH IP ADDRESS>" `
    WAZUH_REGISTRATION_SERVER="< WAZUH IP ADDRESS>"
```

This will download the agent installer, save it to the temp folder, and silently install the agent by specifying the Wazuh manager and registration server:

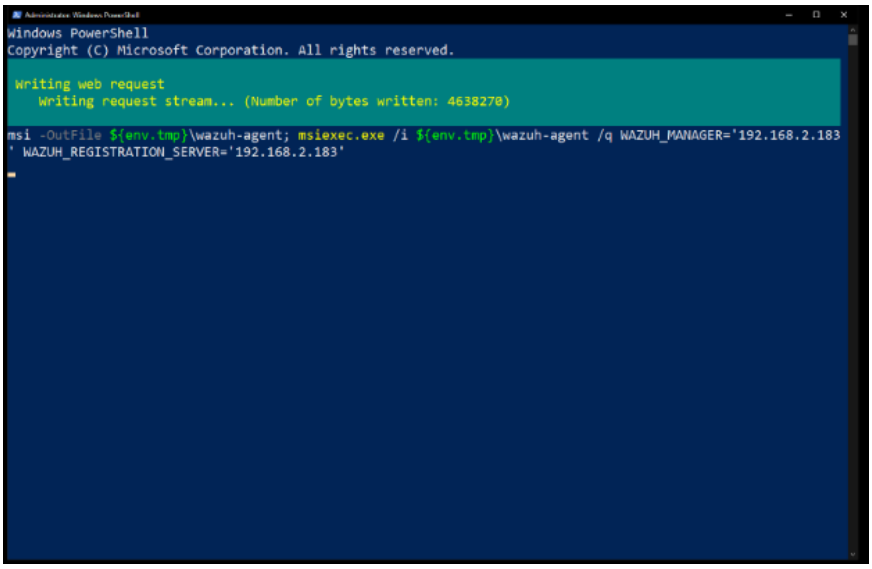


Figure 85 – Wazuh Agent Installation

12. Once the script has completed the download and installation, you can start the agent by running the following command:

```
NET START WazuhSvc
```

It's important to note that you may need administrator privileges to run the `NET START` command and start the Wazuh agent service. If you encounter any permission-related issues, make sure you are running the command with the necessary privileges.

After starting the Wazuh agent service, it will continue to run in the background, performing its security monitoring tasks until it is manually stopped or the system is shut down:

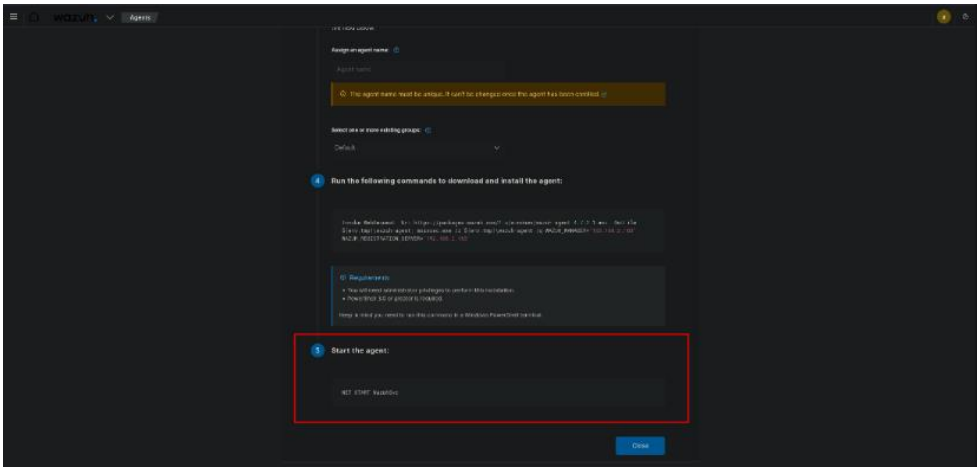


Figure 86 – Wazuh Agent Creation Final Details

By running the `NET START WazuhSvc` command, you are essentially activating the Wazuh agent on the Windows system, enabling it to perform its security monitoring functions and contribute to the overall security visibility provided by the Wazuh system:

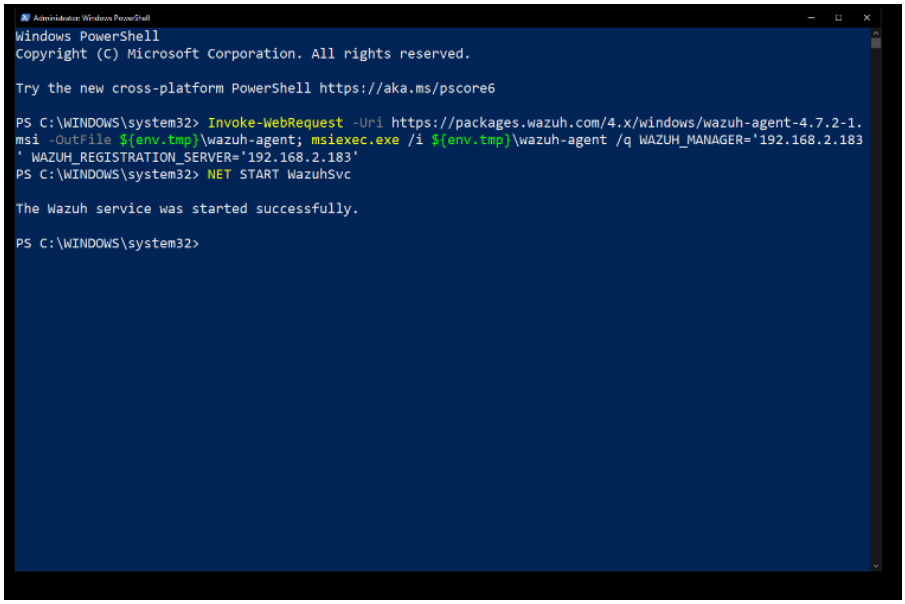


Figure 87 – Run Wazuh Agent

13. Then verify the installation within the Wazuh dashboard:

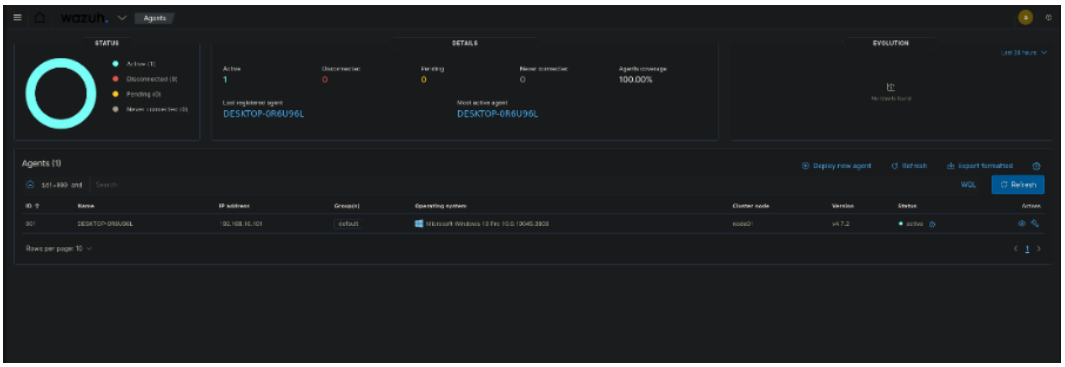


Figure 88 – Wazuh Server Agent Dashboard

14. Now, let us move to testing and validation. Generate security events on the agent VM (e.g., create and delete files in critical directories).

15. Validate that the events are detected by the agent and reported to the manager.

16. Monitor alerts and analyze the data collected by Wazuh:

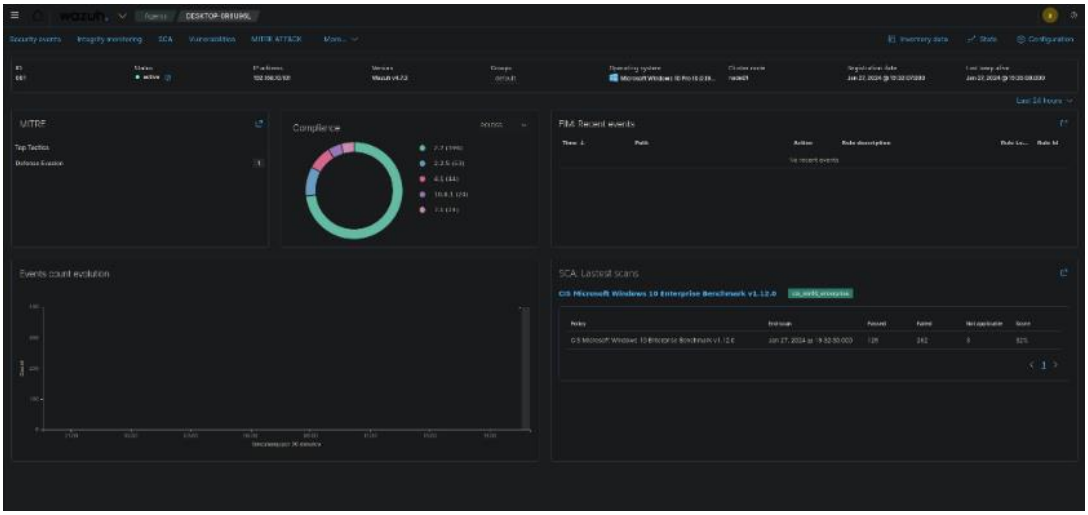


Figure 89 – Wazuh Agent Monitor Dashboard

17. Configure active response in Wazuh to automatically take action in response to certain triggers:

Figure 90 – Wazuh Event Triggers

18. Generate and review reports through the Wazuh:

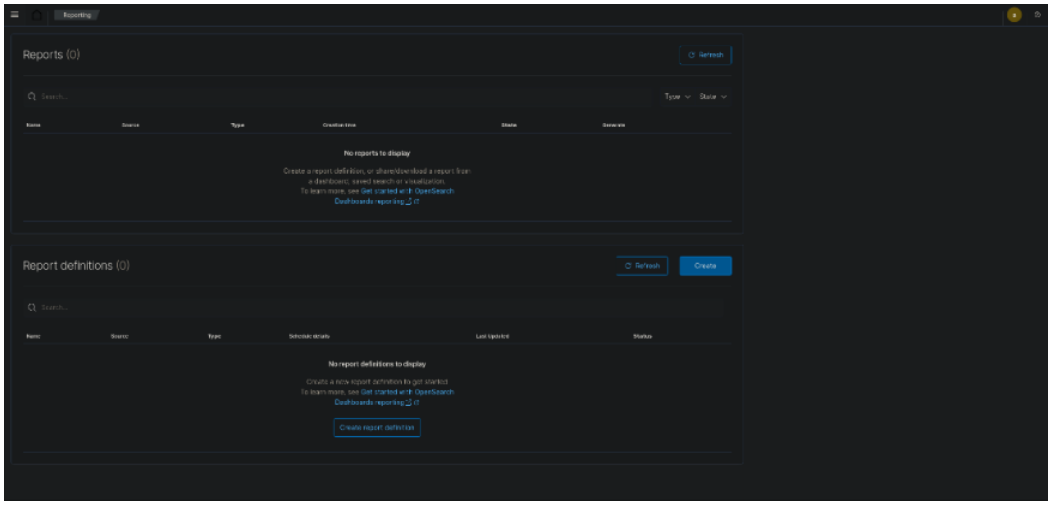


Figure 91 – Wazuh Event Report Creation

Both labs emphasize the importance of performing operations within a controlled environment, understanding configuration and output, and ensuring that the system's detection and response capabilities are functioning as expected. Additionally, these labs facilitate familiarity with real-time incident response scenarios and threat hunting practices.

By combining malware prevention, advanced threat analytics, and unified visibility, endpoint protection delivers in-depth security for devices that often represent a prime attack vector for infiltrating enterprise networks.

Identity and access management (IAM) tools

Let us look at the exercise.

Exercise: Setting Up and Configuring Keycloak for IAM

IAM encompasses tools and processes for ensuring that the right individuals have access to the appropriate resources for the right reasons. Here, I will outline an exercise that utilizes Keycloak, a popular open-source IAM solution. Packt has a book detailing Keycloak in detail called *Keycloak - Identity and Access Management for Modern Applications*, consider picking it up and turning the exercise into a lab and if you want to work more with Keycloak.

Keycloak is an open-source **identity and access management** (**IAM**) solution that provides authentication, authorization, and **single sign-on** (**SSO**) capabilities for modern applications and services. Developed by Red Hat, Keycloak offers a comprehensive and flexible platform for securing and managing user identities across diverse environments, including on-premises, cloud, and hybrid infrastructures. With its rich set of features, such as user federation, RBAC, and support for various identity protocols (OAuth 2.0, OpenID Connect, SAML), Keycloak simplifies the implementation of secure and scalable authentication and authorization mechanisms. It provides a centralized user management system, allowing administrators to create, manage, and authenticate users across multiple applications and services. Keycloak's extensible architecture and customization options make it adaptable to different security requirements and integration scenarios. Whether you are building a single application or a complex microservices architecture, Keycloak empowers developers and administrators to focus on their core business logic while relying on a robust and secure identity management solution.

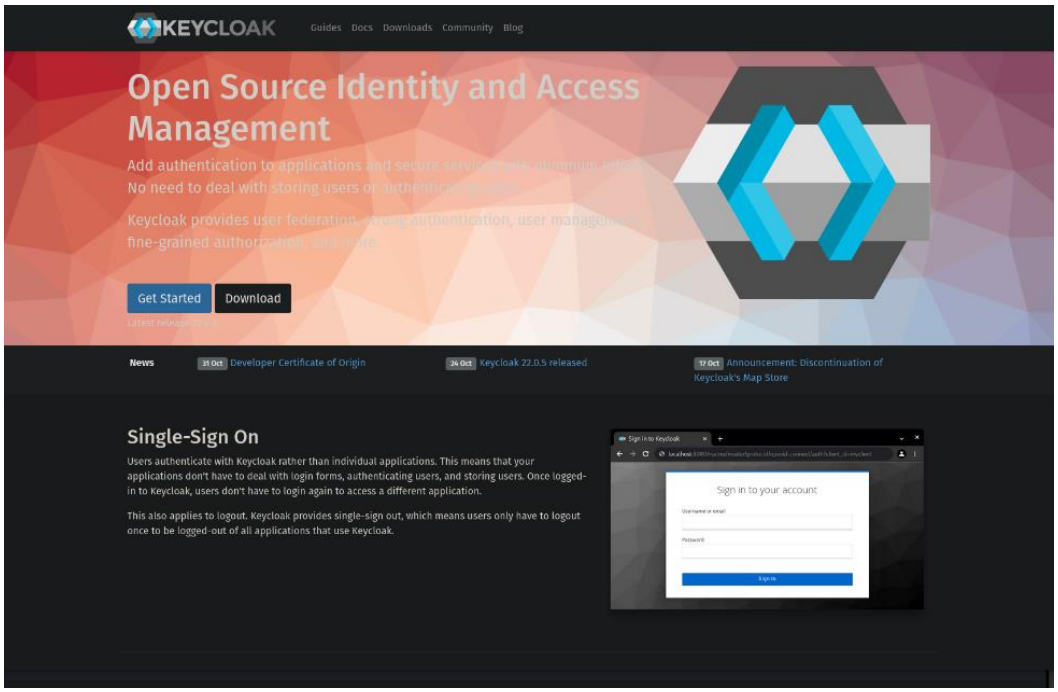


Figure 92 – Keycloak Website

The prerequisites are:

- A virtualization platform such as VMware, VirtualBox, or a cloud service capable of hosting VMs.
- A VM with at least 2 GB RAM and 2 CPU cores.
- A supported operating system installed on the VM, such as Ubuntu Server.
- Internet access for downloading software packages.

Let us look at the steps:

1. Prepare a VM with the chosen operating system, following best practices for setting up a secure VM environment.
2. Ensure the VM is connected to the network with proper firewall rules to allow HTTP/HTTPS traffic.
3. Download and install Java JDK which is a prerequisite for Keycloak. For Ubuntu, use:

```
sudo apt install default-jdk
```

4. Download Keycloak from the official website, https://www.keycloak.org/, using `wget` or `curl`.
5. Unzip the Keycloak archive to an appropriate location, for example, `/opt/keycloak`.
6. Navigate to the Keycloak bin directory and run the standalone script with `./standalone.sh` to start the server.
7. Access the Keycloak admin console via a web browser at http://[VM_IP]:8080/auth/.
8. Complete the initial setup by creating an admin account.
9. Once logged in, create a new realm by clicking on **Add realm**. Give it a meaningful name that represents your organization or project.

10. Within the realm, configure necessary tokens, session settings, and other realm-specific settings.
11. Navigate to the **Users** section and add users manually or by importing a user list.
12. Assign credentials to users and manage their roles and group memberships.
13. In the **Clients** section, register a new client (application) that will be secured by Keycloak.
14. Configure the client with correct protocol (e.g., OpenID Connect), access type, and valid redirect URIs.
15. Define roles under the **Roles** section that will be used to grant access to resources.
16. Create groups under the **Groups** section and map them to the roles.
17. Under the **Authorization** section within clients, set up resource-based policies, permission scopes, and access policies.
18. Use Keycloak's built-in tools to test user authentication and token generation.
19. Verify that users can log in and are granted access according to their roles and group memberships.
20. Configure Keycloak to act as an **identity provider** (**IdP**) by setting up identity brokering with external providers (if needed).
21. Configure Keycloak to act as a **service provider** (**SP**) by integrating with external IdP services (if needed).
22. Install Keycloak adapters on applications that should be secured by Keycloak for **single sign-on** (**SSO**) capability.
23. Examine the logs for any authentication or authorization issues.
24. Set up audit logging to track user sessions and operations.

25. Create a backup of the Keycloak database and the configuration files.
26. Document a recovery process in case of failure.
27. Monitor Keycloak's performance and make necessary adjustments to **Java Virtual Machine** (**JVM**) settings or database configurations.
28. Document the entire setup process, configurations made, and policies implemented.
29. Include diagrams and flowcharts that visualize the authentication and authorization flows.

By following these steps, you will set up a working IAM environment using Keycloak that can manage users, authenticate and authorize client applications, and integrate with other IdP services if necessary. The lab's ultimate goal is to provide hands-on experience with IAM best practices, tools, and configurations in a controlled, virtualized setting.

By centralizing identity, access, and privileges, IAM limits the attack surface and enforces least privilege, providing accountability and auditability for access.

Data protection tools

Let us look at the lab.

For the purpose of this lab, we will focus on encryption using VeraCrypt, an open-source disk encryption software.

Lab: Data Encryption with VeraCrypt

VeraCrypt is a powerful open-source disk encryption software that provides a high level of security for protecting sensitive data on your computer. It is a fork of the discontinued TrueCrypt project and builds upon its strong foundation while adding enhanced security features and addressing known vulnerabilities. VeraCrypt allows you to create encrypted volumes or containers, which can be mounted as virtual disks, and encrypt entire partitions or storage devices, such as hard drives or USB drives. With VeraCrypt, you can safeguard your confidential files, documents, and personal information from unauthorized access, theft, or data breaches. It supports various encryption algorithms, including AES, Twofish, and Serpent, and offers multiple cascading encryption modes for added security. VeraCrypt also provides plausible

deniability through hidden volumes, allowing you to create a decoy system within an encrypted volume to further protect your sensitive data. Whether you are an individual, business, or organization dealing with confidential information, VeraCrypt is a reliable and user-friendly solution for ensuring the privacy and integrity of your digital assets.

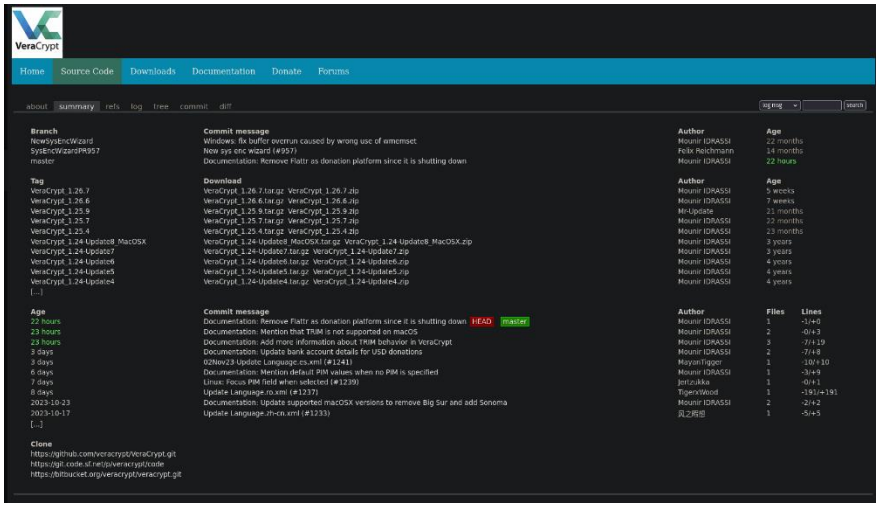


Figure 93 – Veracrypt Website

The prerequisites are:

- A virtualization platform like VirtualBox, VMware, or similar, where VMs can be created.
- A VM with a Windows or Linux operating system installed.
- Internet access for downloading VeraCrypt.
- Basic knowledge of disk encryption and file systems.

Let us look at the steps:

1. Create a new VM using your virtualization software.
2. Install your chosen operating system on the VM.
3. Within the VM, download VeraCrypt from the official website, https://www.veracrypt.fr/code/VeraCrypt/.

4. On Windows, run the installer and follow the on-screen instructions:

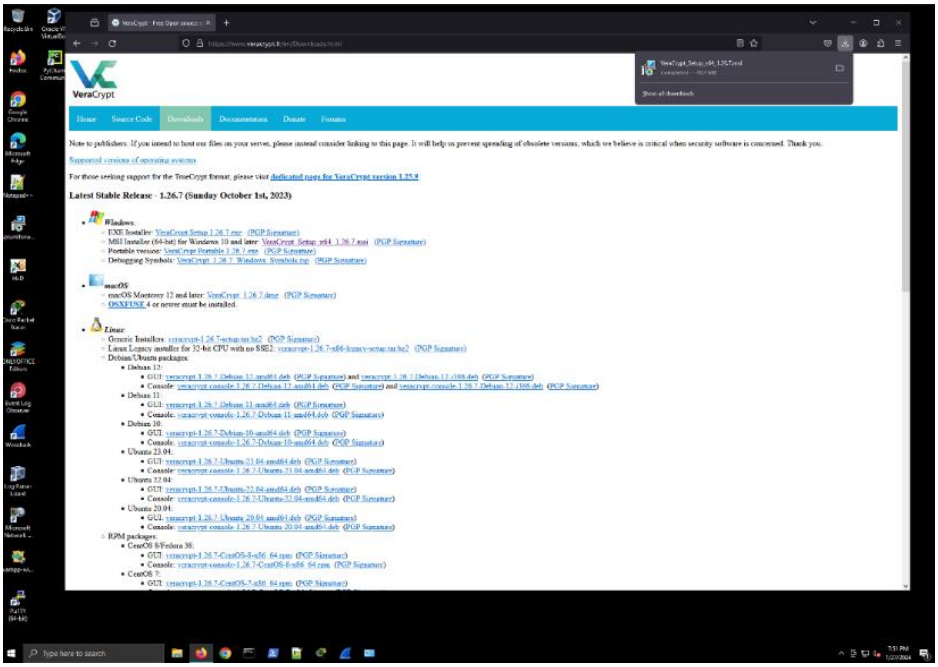
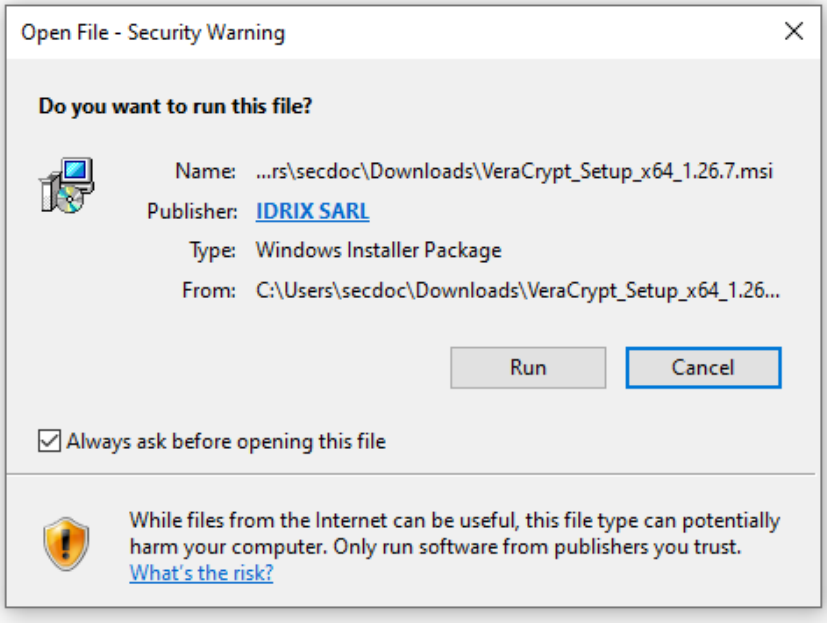


Figure 94 – Veracrypt Download

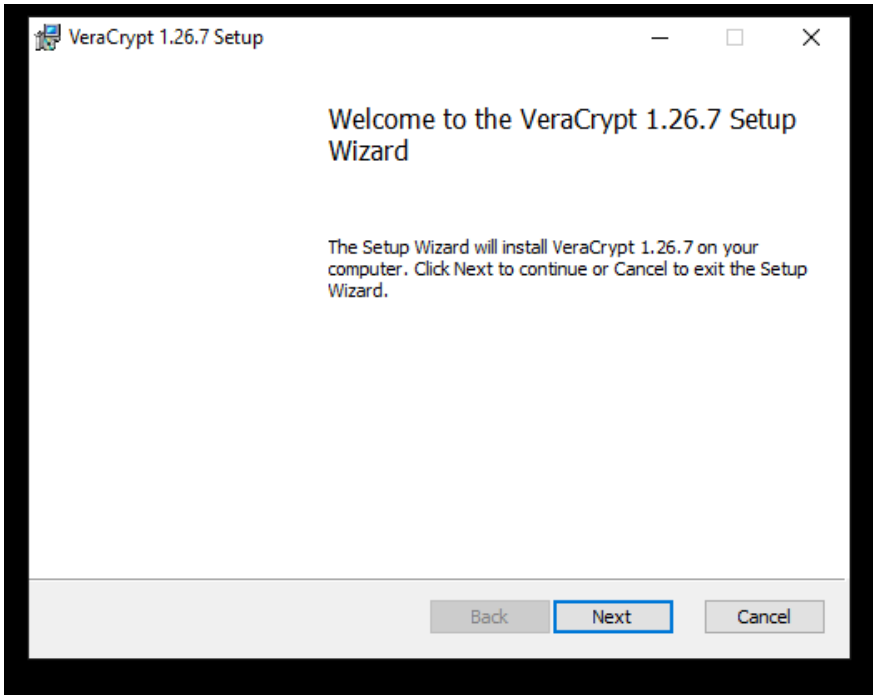
5. On Linux, extract the downloaded package and run the installation script.

   This lab provides instructions for creating, mounting, and using a VeraCrypt volume stored in a file container. It is recommended to read other sections of the VeraCrypt manual for important additional information.

6. Download and install VeraCrypt if you have not already. Launch VeraCrypt by double-clicking `VeraCrypt.exe` or the Start menu shortcut.


Open File - Security Warning
Do you want to run this file?
Name: ...rs\secdoc\Downloads\VeraCrypt_Setup_x64_1.26.7.msi
Publisher: IDRIX SARL
Type: Windows Installer Package
From: C:\Users\secdoc\Downloads\VeraCrypt_Setup_x64_1.26...
Run
Cancel
Always ask before opening this file
While files from the Internet can be useful, this file type can potentially harm your computer. Only run software from publishers you trust.
What's the risk?

(a)


VeraCrypt 1.26.7 Setup
Welcome to the VeraCrypt 1.26.7 Setup Wizard
The Setup Wizard will install VeraCrypt 1.26.7 on your computer. Click Next to continue or Cancel to exit the Setup Wizard.
Back
Next
Cancel

(b)

Figure 95 – Installing Veracrypt Windows Installer

7. In the main VeraCrypt window, click **Create Volume** to open the Volume Creation Wizard:

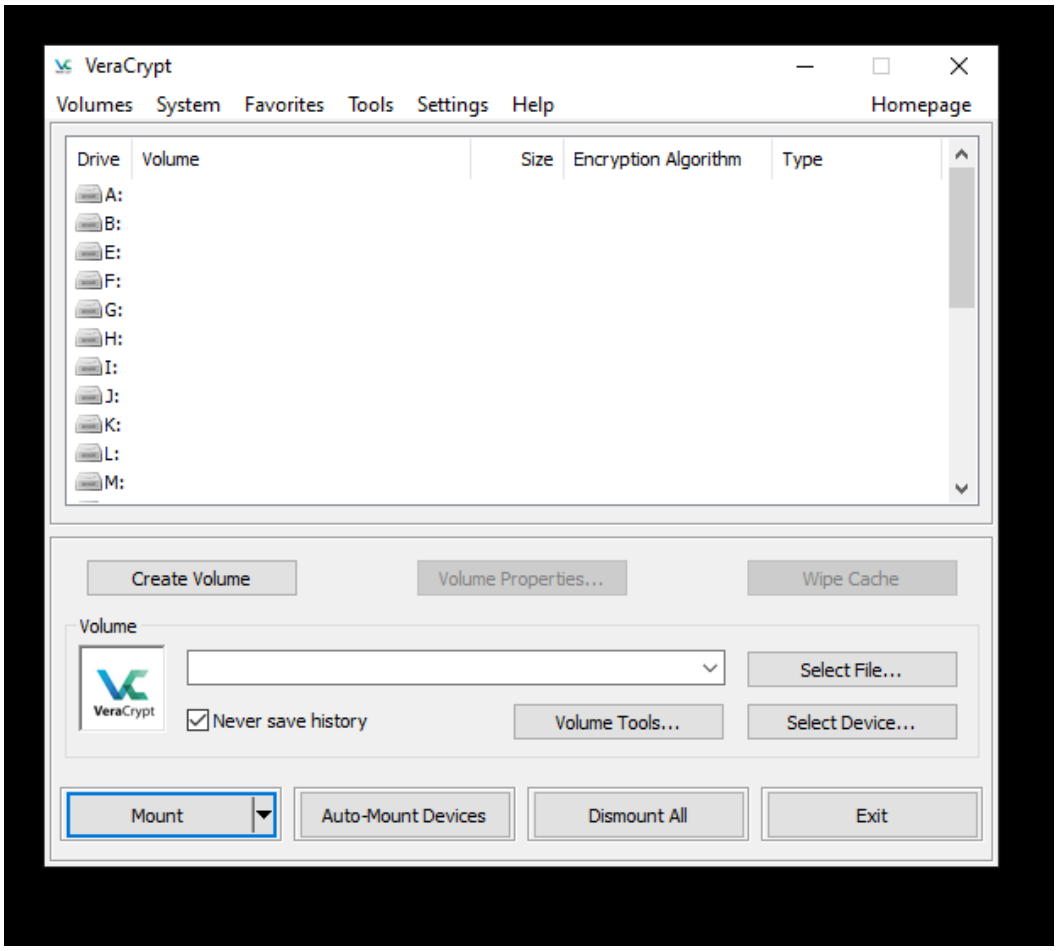


Figure 96 – Veracrypt Dialog Window

8. In the Wizard, choose **Create an encrypted file container** as the volume type. Click **Next**:

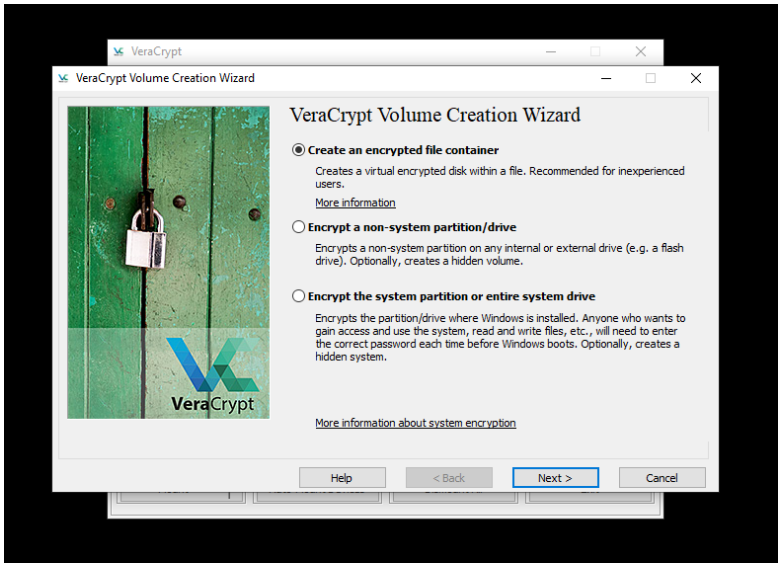


Figure 97 – Veracrypt Volume Creation Wizard

9. Select **Standard VeraCrypt volume** as the volume format and click **Next**:

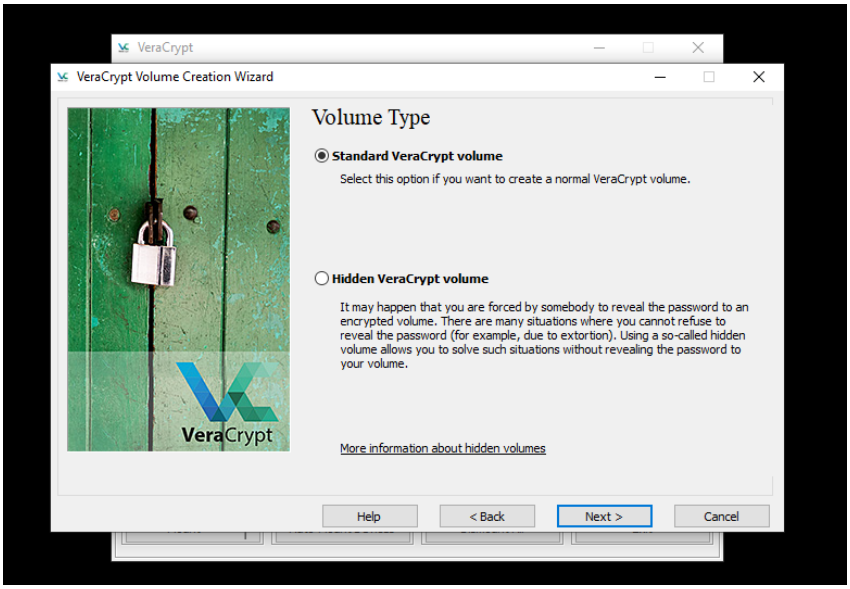


Figure 98 – Veracrypt Volume Type

10. Click **Select File** to choose where to create the VeraCrypt container file:

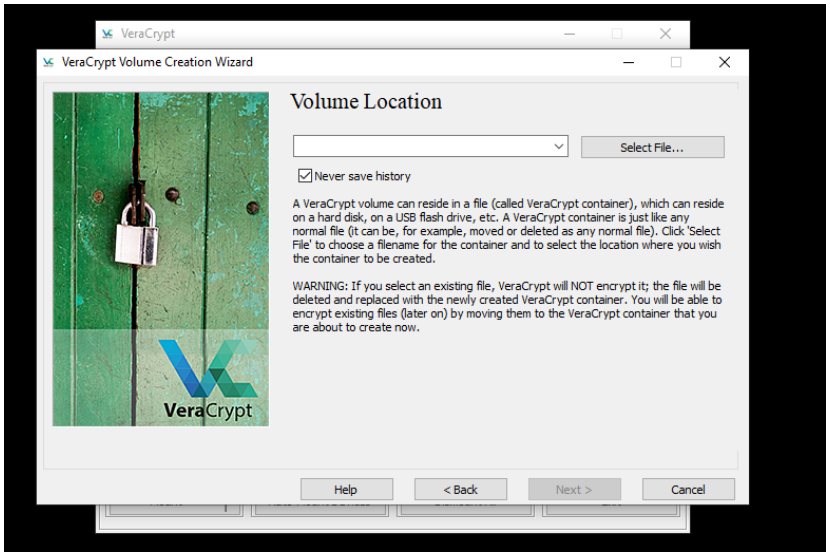


Figure 99 – Veracrypt Volume Location

11. Browse to the desired folder (e.g. `F:\Data`) and enter a filename for the container (e.g. `MyVolume.hc`). Click **Save** to return to the Wizard:

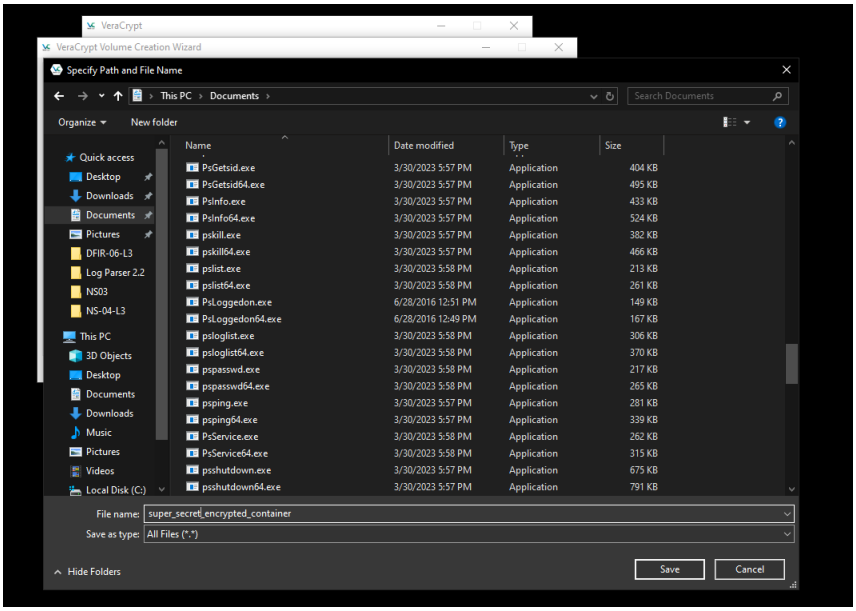


Figure 100 – Veracrypt Volume Location Selection

12. Click **Next** to continue in the Wizard:

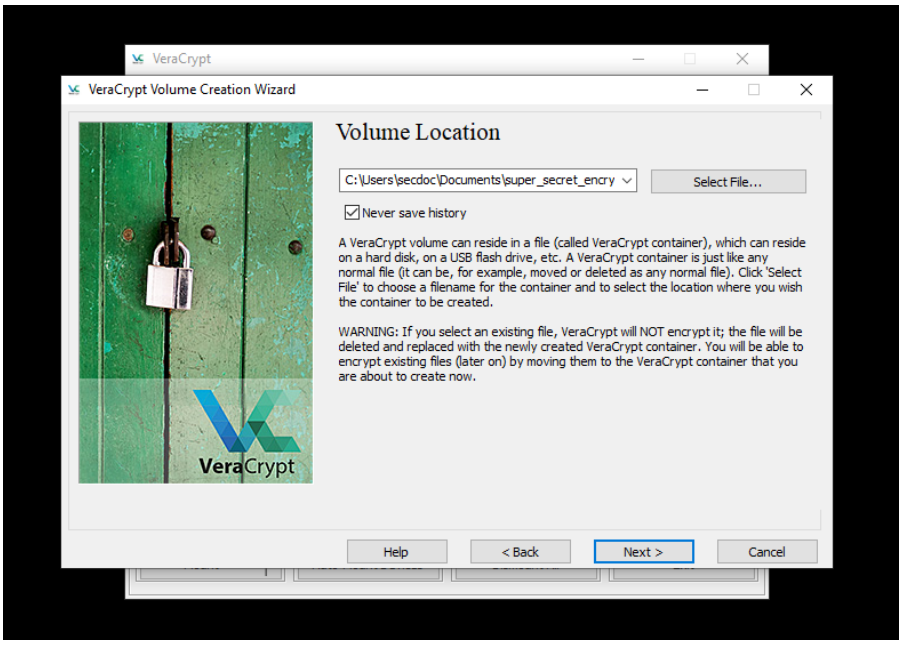


Figure 101 – Veracrypt Volume Location Wizard

13. Keep the default encryption and hash algorithms or choose your own preferences. Click **Next**:

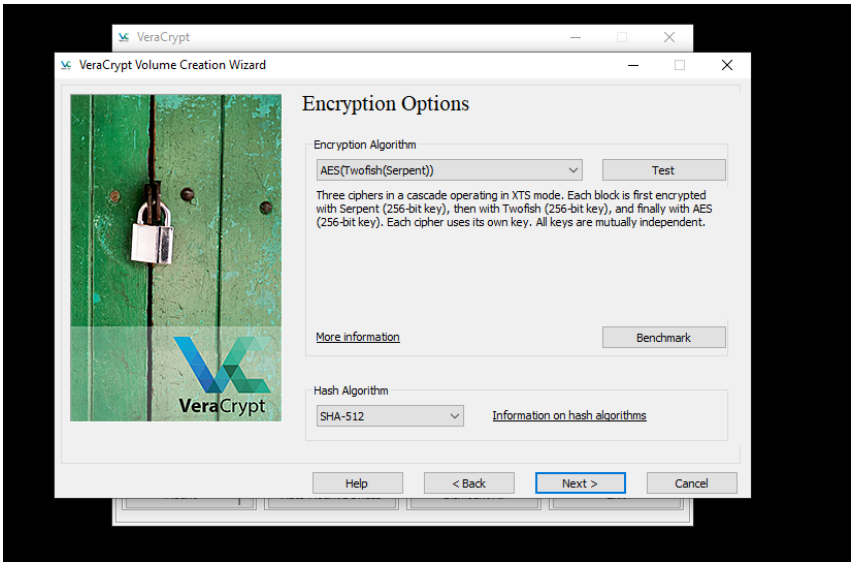


Figure 102 – Veracrypt Wizard Encryption Options

14. Enter the desired size of the container (e.g. 250 MB). Click **Next**:

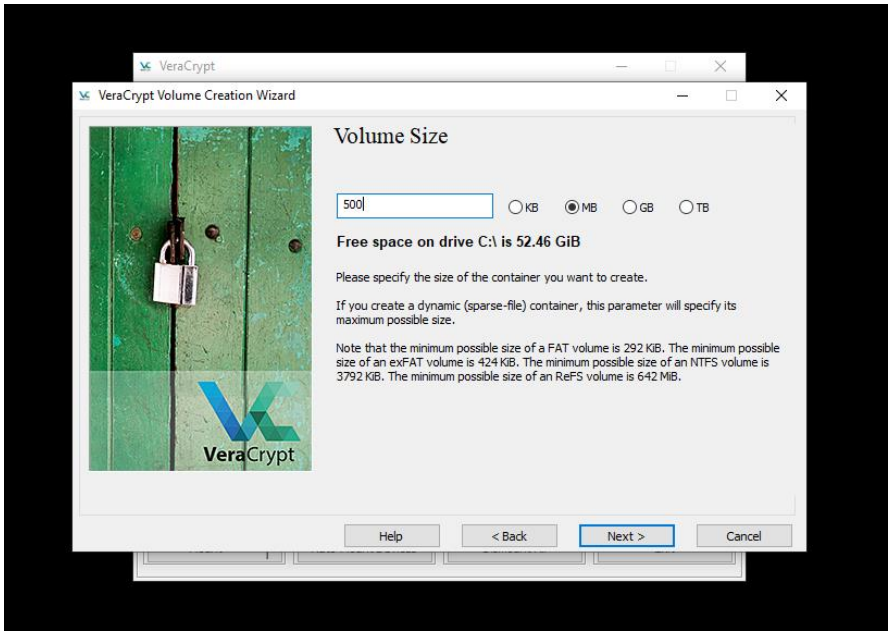


Figure 103 – Veracrypt Volume Size

15. Choose a secure password and enter it twice to continue:

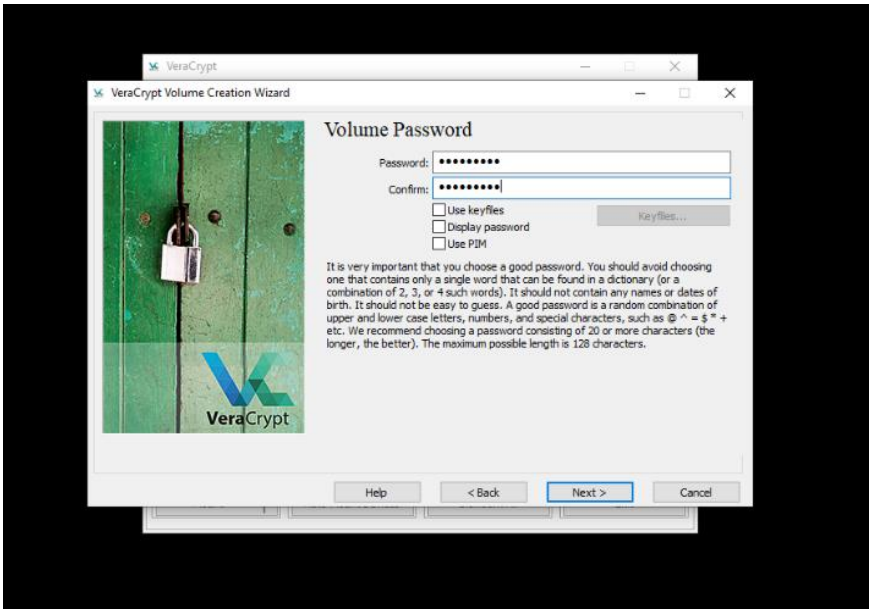


Figure 104 – Veracrypt Volume Password

16. Move your mouse randomly within the Wizard for 30+ seconds to generate encryption keys.

17. Click **Format** to create the container file. Click **OK** when done and **Exit** to exit:

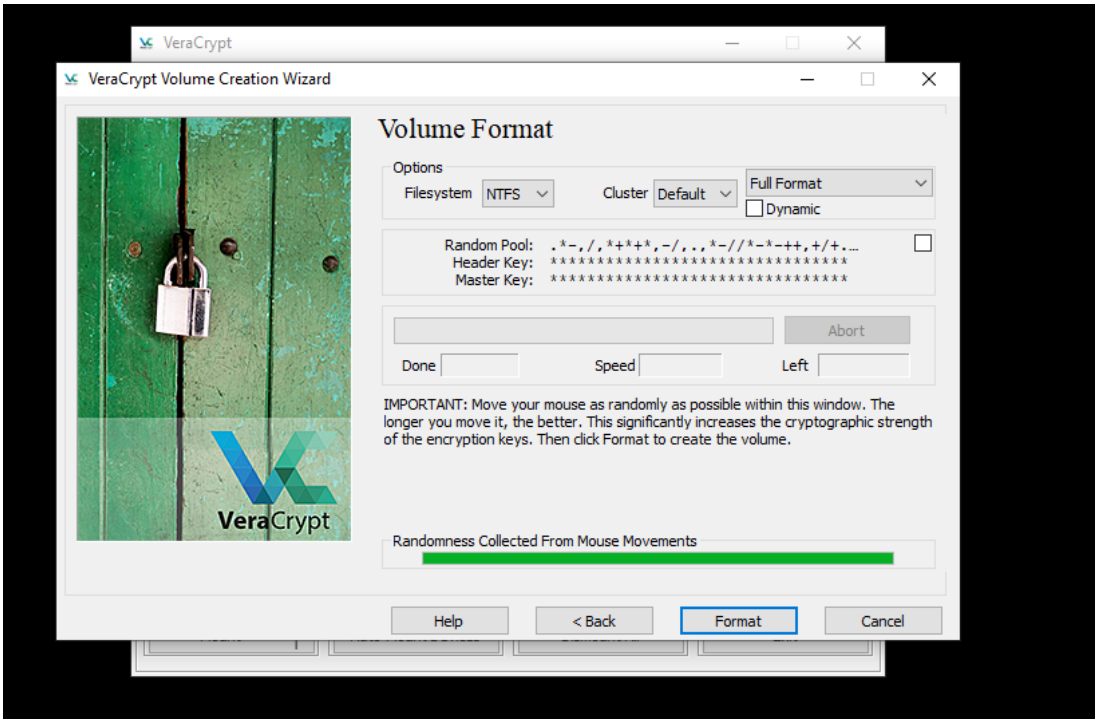


Figure 105 – Veracrypt Volume Format

18. In the main VeraCrypt window, choose a drive letter to mount the container.
19. Click **Select File** and browse to the container file created earlier. Click **Open**.
20. Click **Mount** and enter the container password. Click **OK**:

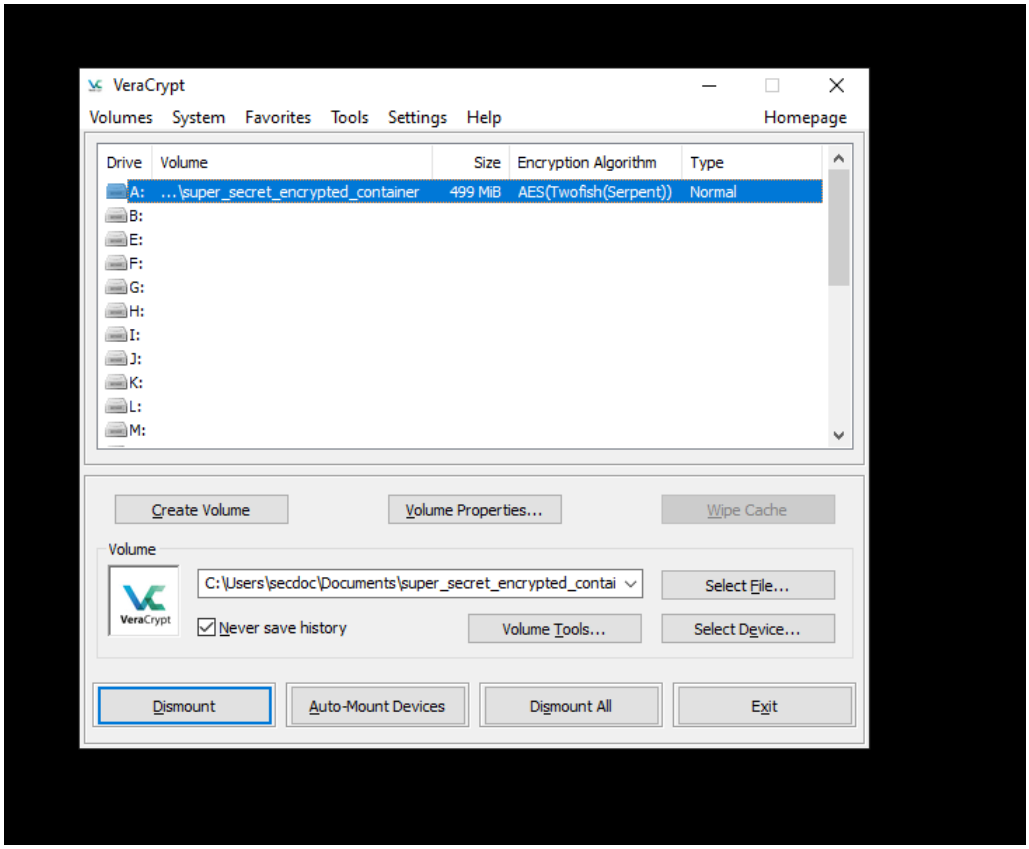


Figure 106 – Veracrypt Volume Status Window

21. The container is now mounted as an encrypted virtual disk. To close, click **Dismount** in VeraCrypt.
22. Now, let's move ahead to understanding VeraCrypt. Open VeraCrypt and familiarize yourself with the user interface.
23. Review the documentation to understand concepts like volumes, mount points, and encryption algorithms.
24. Now, onto creating an encrypted volume. Once mounted, the encrypted volume behaves like any other drive. Store sensitive files inside this volume. When unmounted, the files are secured with the chosen encryption.

25. To secure the data, dismount the volume in VeraCrypt. Verify that the volume is inaccessible without mounting it again with the correct password.
26. Copy the encrypted volume file to another secure location as a backup. Document the volume details, password, and keyfile locations for recovery purposes.
27. Understand the importance of strong passwords, backup strategies, and the implications of encryption on system performance.
28. Explore other VeraCrypt features such as hidden volumes, system encryption, and creating a VeraCrypt Rescue Disk.
29. Test the performance of your system with the encrypted volume mounted.
30. Practice recovery scenarios, including mounting the volume on a different system or VM.
31. Document the process, configuration choices, and any issues encountered.
32. Prepare a guide for end-users on how to access and use the encrypted volume.

Through this lab, participants will acquire practical skills in using encryption as a data protection tool, understanding the balance between security and usability, and the importance of comprehensive documentation and user education. This hands-on experience is vital for cybersecurity professionals tasked with safeguarding sensitive information.

Given growing data volumes and increasingly sophisticated threats, architects must integrate robust controls for data discovery, access, transmission, encryption, and analytics. Doing so limits exposure while enabling data utility across infrastructure.

Vulnerability management tools

Vulnerability management is a critical component in the cybersecurity domain, focusing on the identification, classification, remediation, and mitigation of various software vulnerabilities. One of the prominent open-source tools in this arena is OpenVAS (Open Vulnerability Assessment System). This lab will guide you through setting up and using OpenVAS within a virtual environment.

Lab: Vulnerability Scanning with OpenVAS

OpenVAS (**Open Vulnerability Assessment System**) is a powerful open-source vulnerability scanning and management framework that helps organizations identify and assess security vulnerabilities in their networks, systems, and applications. It is a widely used tool for performing comprehensive vulnerability scans and generating detailed reports to aid in the remediation process. OpenVAS consists of a scanner, a manager, and a web-based user interface, providing a centralized platform for managing and executing vulnerability scans. With its extensive database of **network vulnerability tests** (**NVTs**), OpenVAS can detect a wide range of security issues, including missing patches, misconfigurations, and known vulnerabilities. It supports various scan types, such as authenticated and unauthenticated scans, and can be customized to fit specific security requirements. OpenVAS integrates with other security tools and frameworks, allowing for seamless integration into existing security workflows. Its regular updates and active community support ensure that OpenVAS stays up-to-date with the latest vulnerability information, making it an essential tool for proactively identifying and mitigating security risks in today's dynamic threat landscape.

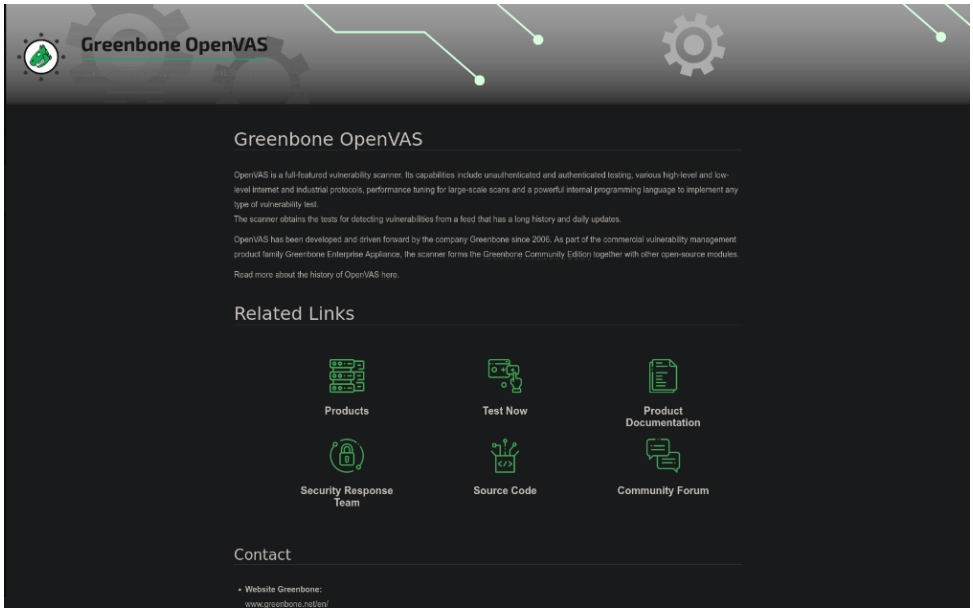


Figure 107 – OpenVAS/Greenbone Website

You can access this here: https://openvas.org/.

The prerequisites include:

- A virtualization solution such as VirtualBox, VMware, or a cloud-based platform capable of deploying virtual machines.
- A VM with at least 4 GB RAM and 2 CPU cores.
- A supported Linux distribution installed on the VM, such as Kali Linux, which comes with OpenVAS pre-installed. Understand the while Kali comes with OpenVAS installed it is not the version maintained by OpenVAS and there are known issues with the installation in Kali.
- Internet access for updates and downloading plugins.

Let us look at the steps:

1. Set up a new VM on your virtualization platform with your chosen Linux distribution. The documentation at the end of this bullet provides instructions for installing Greenbone Community Edition from the native Kali Linux repository. The Greenbone install packages are maintained by Offensive Security. Any issues found during installation or usage should be reported through the Kali Linux Bug Tracker, following the guidelines for submitting Kali bugs. If you would like to install OpenVAS on another Linux Distro, you can follow the guide at the following OPenVAS link: https://greenbone.github.io/docs/latest/22.4/container/index.html.
2. Configure the network settings for the VM to ensure it can reach the internet and the internal network for scanning purposes. Before installing Greenbone Community Edition, first update the local package lists for all configured repositories and **personal package archives** (**PPAs**) on your Kali Linux system. As a rolling release distribution, Kali continuously updates system software to the latest versions without requiring OS reinstallation. Rolling releases typically provide new software soon after release. It is highly recommended to also perform a full package upgrade beforehand since Greenbone requires the newest PostgreSQL version. Upgrading proactively avoids potential issues configuring PostgreSQL later in the installation process.

See the troubleshooting section if having problems with the PostgreSQL upgrade or configuration while installing Greenbone.

3. For distros that do not come with OpenVAS, install it via the package manager. For example, on Debian-based systems: `sudo apt-get install openvas`:

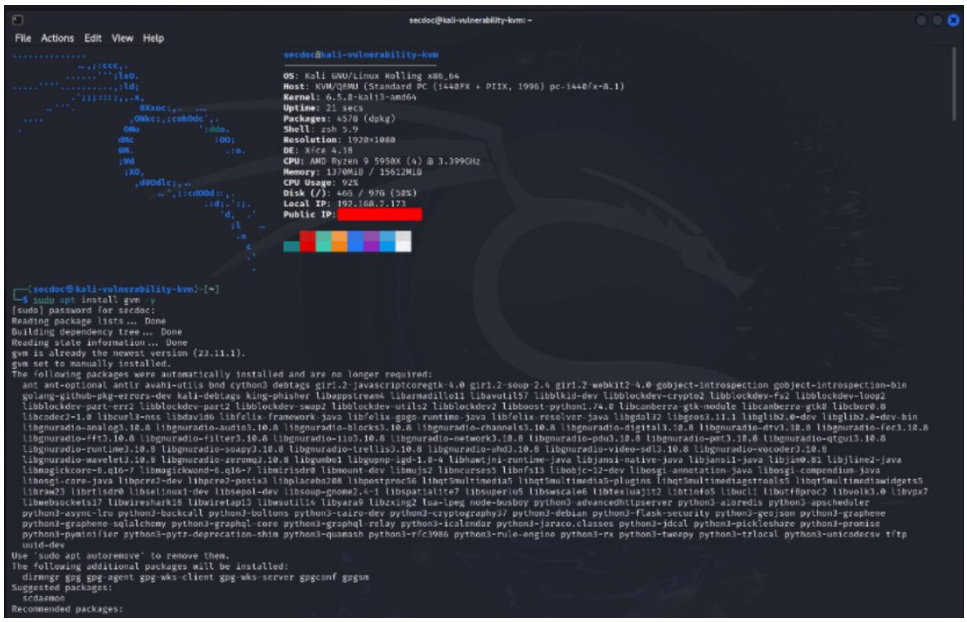


Figure 108 – OpenVAS Installation on Kali

On Kali Linux, OpenVAS (now known as GVM) can be set up by running `sudo gvm-setup`.

4. Run the setup script if necessary, which will download the latest vulnerability feeds and configure the various components of OpenVAS:

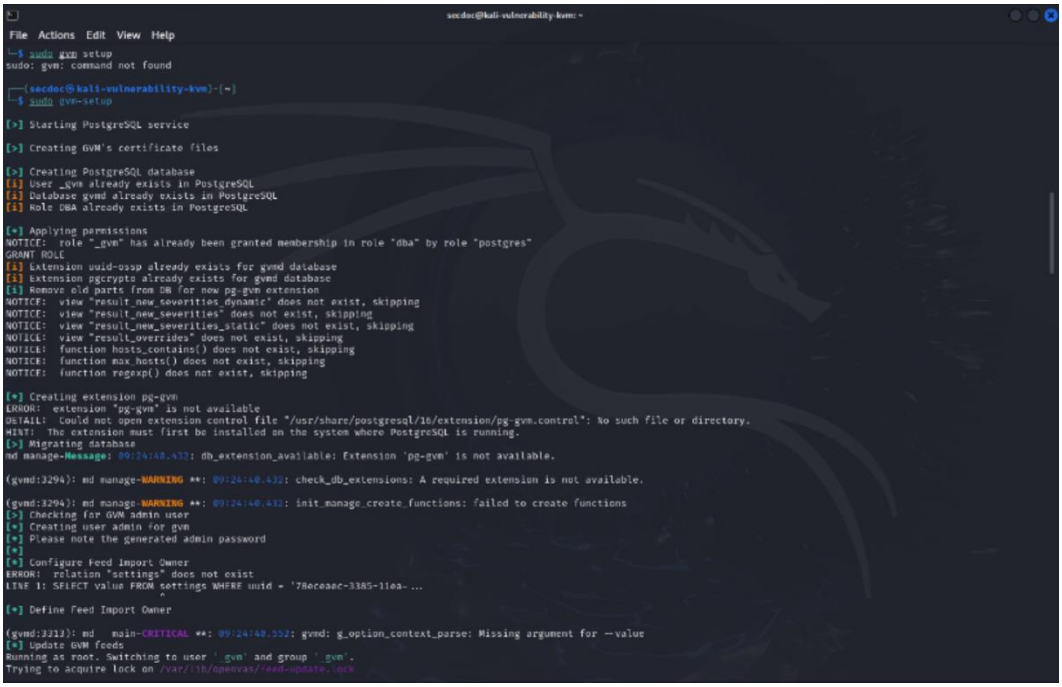


Figure 109 – OpenVAS Setup

Note

There is a known error with some of the installation modules and separate installation of those modules may be required. Such as the installation of the `pg-gvm` module. To correct run the following commands: `sudo apt install postgresql-16-pg-gvm` and `sudo runuser -u postgres -- /user/share/gvm/create-postgresql-database`.

5. After running the setup script run `sudo gvm-check-setup` for validation of the installation and default configuration:

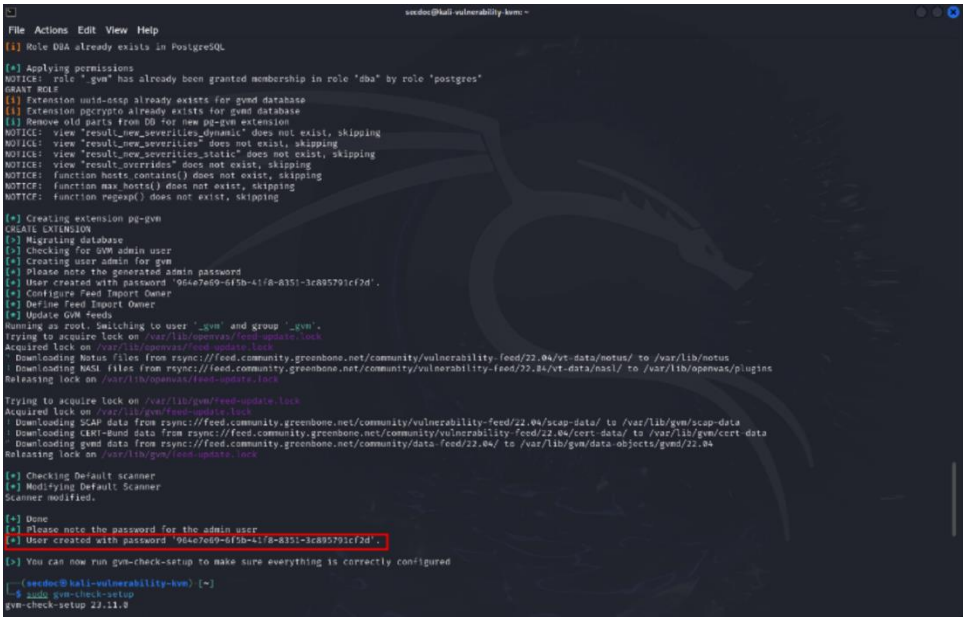


Figure 110 – OpenVAS Setup Validation and initial password

Note

Take note of the admin user generated password on completion of a successful setup. This is a randomly generated key at first installation.

6. The Kali Linux installation of Greenbone uses the same components and configuration options as compiling the source code directly. Here are some common customizations:

   First, let us look at enabling remote web interface access.

   By default, Greenbone is configured for local-only access to the web interface on `127.0.0.1`. To allow external access, edit `/usr/lib/systemd/system/gsad.service` and update the `--listen` argument:

   ```
   -ExecStart=/usr/local/sbin/gsad --foreground --
   listen=127.0.0.1 --port=9392
   ```

```
+ExecStart=/usr/local/sbin/gsad --foreground --
listen=0.0.0.0 --port=443
```

This opens the web interface on all interfaces. Optionally change the port to `443` for default HTTPS:

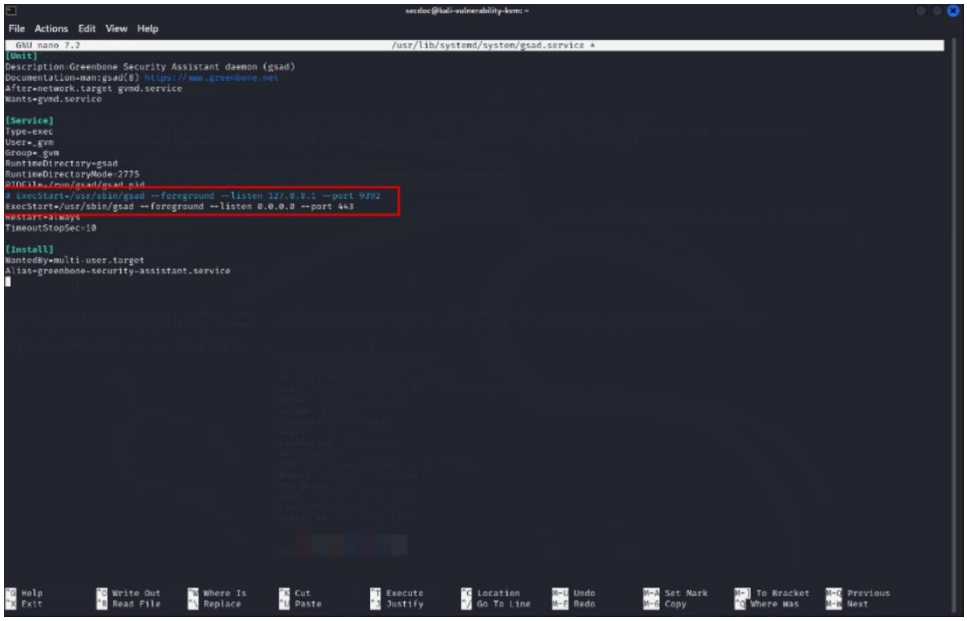


Figure 111 – OpenVAS Listening IP and Port Configuration

7. On Kali, ensure all services related to GVM are started with `sudo gvm-start`:

Figure 112 – Starting OpenVAS

8. By default, OpenVAS serves its web interface on port `9392`. Access this via a web browser by navigating to https://[VM_IP]:9392.

9. Now, let us look at customizing your Greenbone Community Edition installation. The Kali Linux installation of Greenbone uses the same components and configuration options as compiling the source code directly. Here are some common customizations:

   Let us enable remote web interface access.

   By default, Greenbone is configured for local-only access to the web interface on `127.0.0.1`. To allow external access, edit `/usr/lib/systemd/service/gsad.service` and update the `--listen` argument:

```
-ExecStart=/usr/local/sbin/gsad --foreground --
listen=127.0.0.1 --port=9392
+ExecStart=/usr/local/sbin/gsad --foreground --
listen=0.0.0.0 --port=443
```

10. This opens the web interface on all interfaces. Optionally change the port to `443` for default HTTPS:

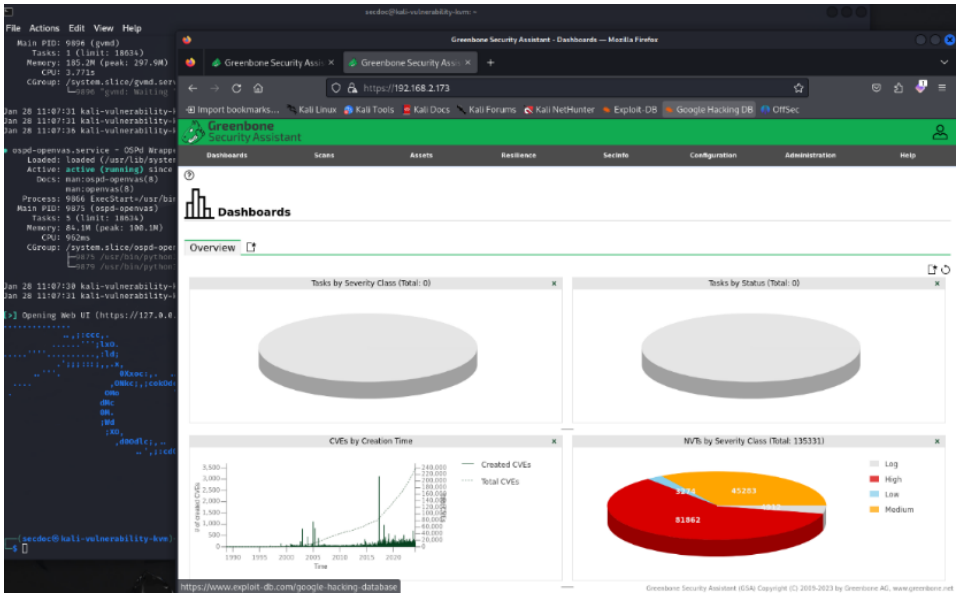


Figure 113 – OpenVAS Dashboard using LAN Interface

11. Login using the credentials set up during the installation or the default provided by the system:

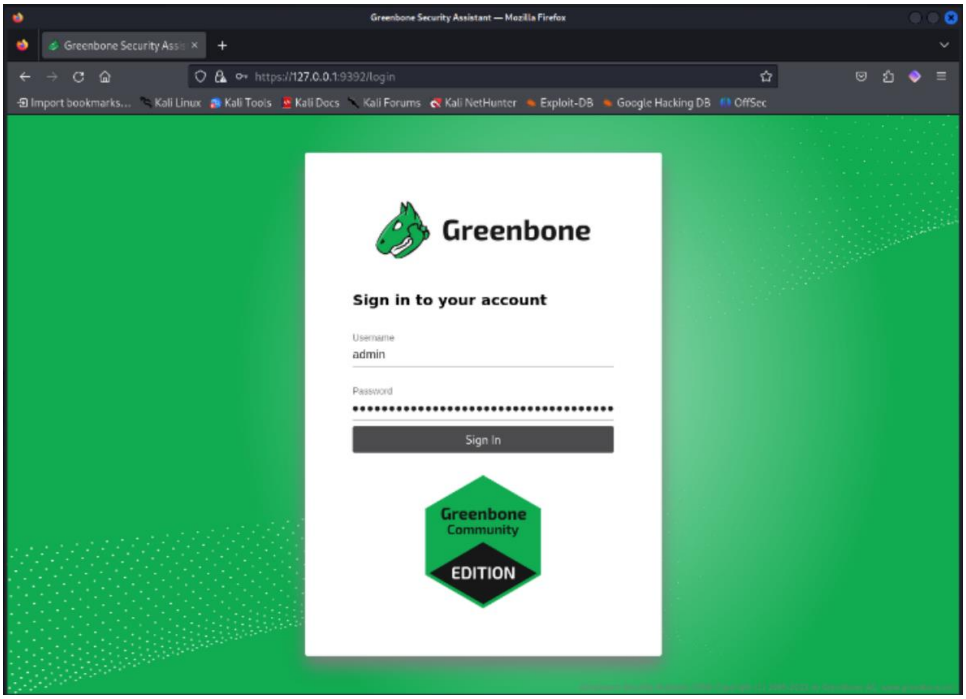


Figure 114 – OpenVAS Login

12. Prior to running initial scans, Greenbone parses vulnerability feed data into the gvmd PostgreSQL database. Without populated vulnerability data, scans cannot initialize or complete without errors. The feed population process begins during Greenbone setup but commonly requires a few minutes to multiple hours to finish based on system resources. The feed status can be monitored on the **Feed Status** page which is located under the **Configuration** menu section. Scanning should not be started until the feeds show as synchronized and finished updating:

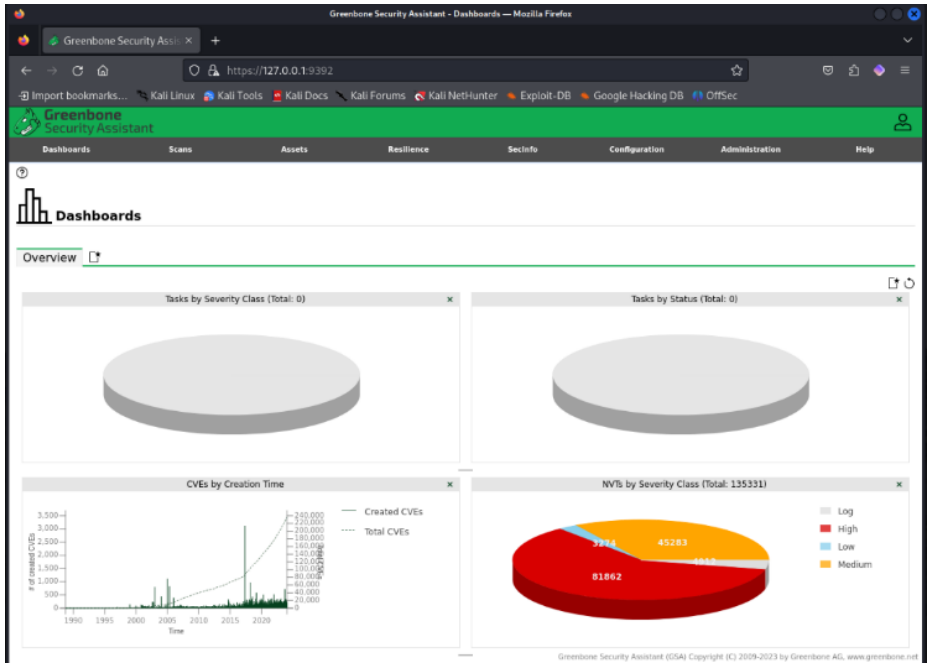


Figure 115 – OpenVAS Dashboard

13. In the web interface, go to the **Scanners** section to ensure your scanner is properly set up:

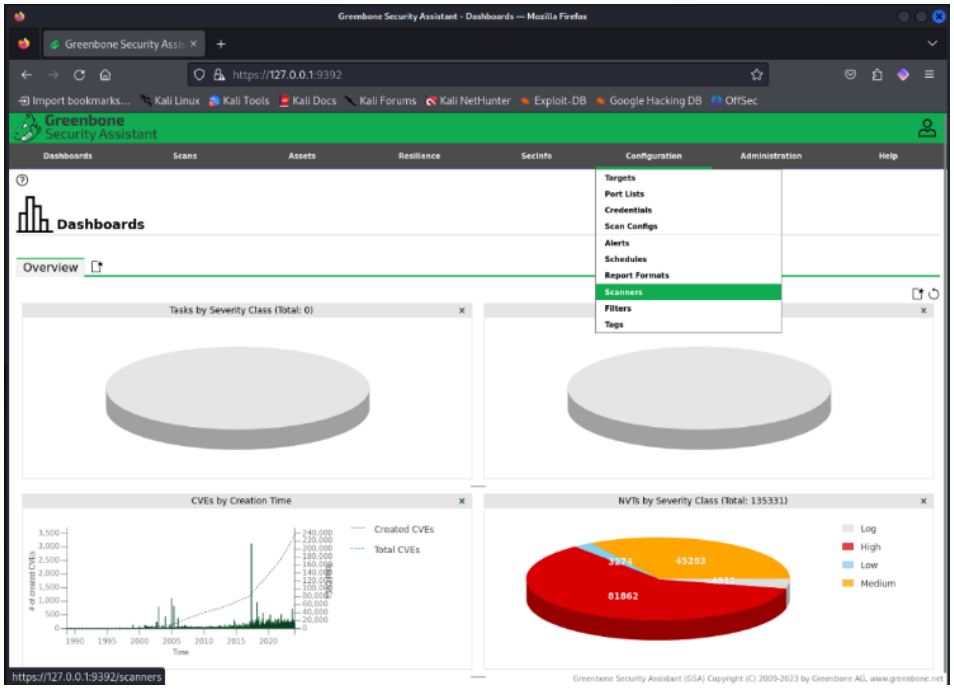


Figure 116 – Selecting Scanner Configuration

14. Navigate to **Configuration** and then **Targets** to define the IP range or specific hosts you wish to scan:

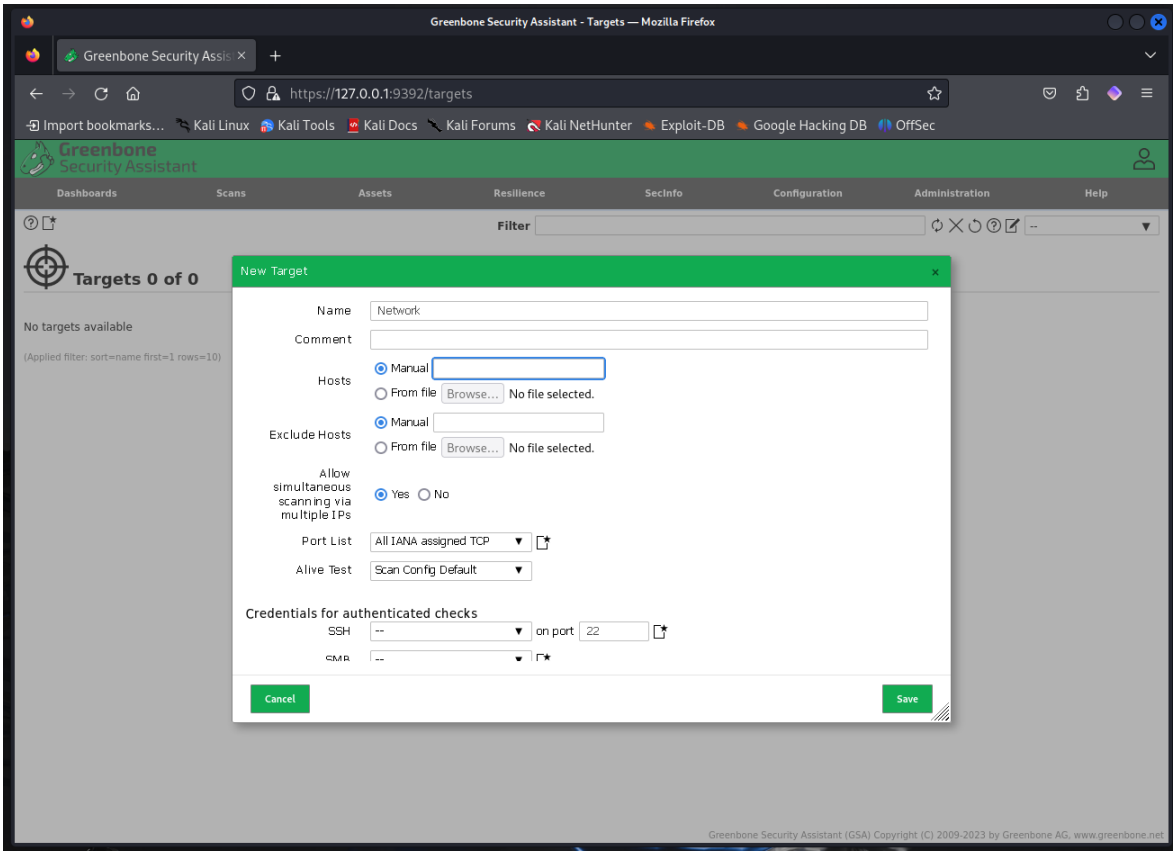


Figure 117 – Defining Targets

15. Go to **Scans** and then **Tasks** to create a new task.

16. Assign a name, select the target previously created, and select the scanning configuration (there are several default configurations available):

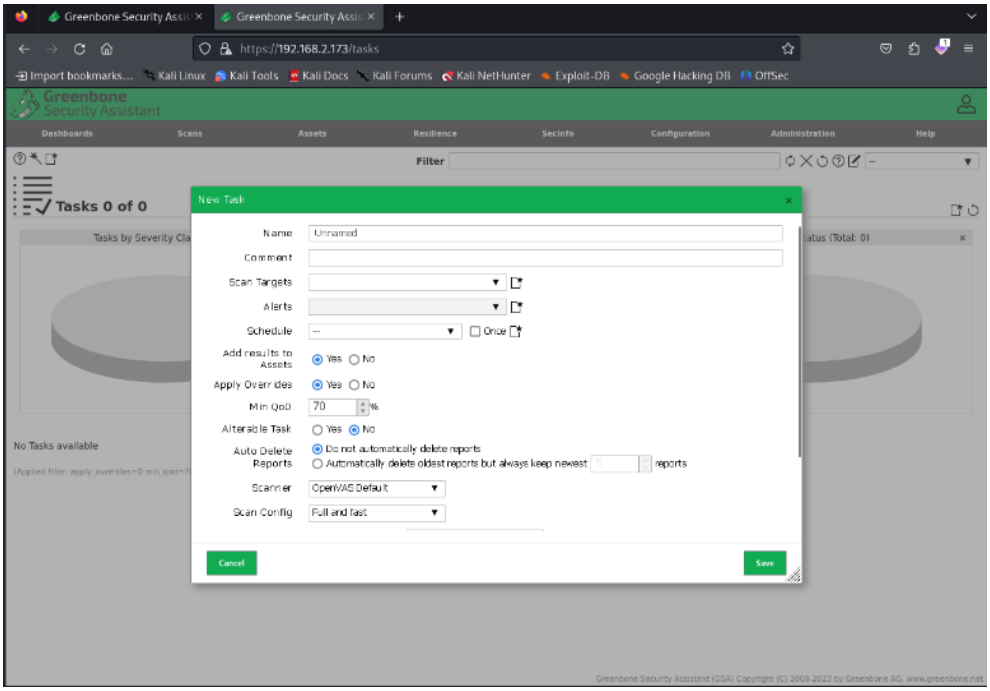


Figure 118 – Defining Tasks

17. Start the scan task and monitor its progress:

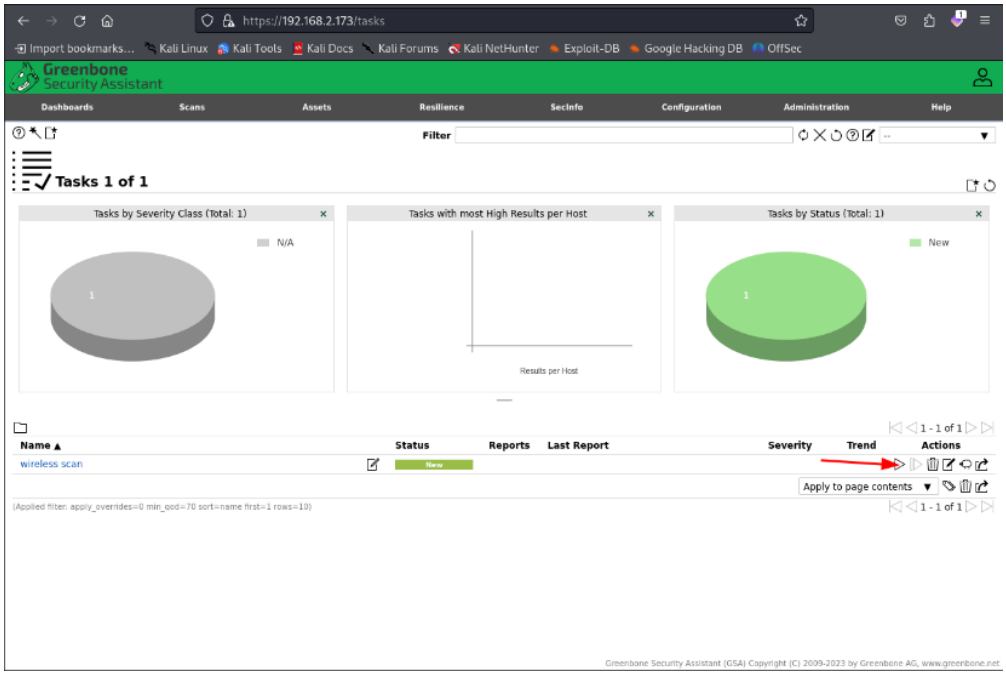


Figure 119 – Starting a Scan

Once the scan is complete, review the results for any identified vulnerabilities.

18. Analyze the reported vulnerabilities, reviewing the severity, description, and recommended actions:

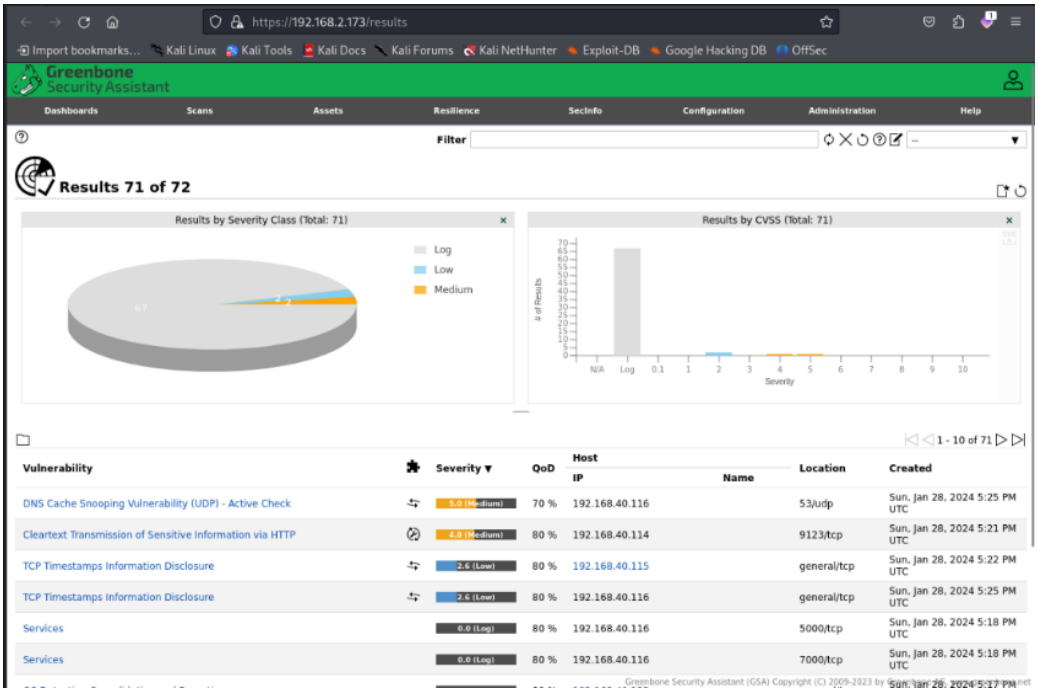


Figure 120 – Scan Results

19. Use the filtering tools to sort and prioritize the vulnerabilities.

20. Generate a report by going to **Reports** in the web interface:

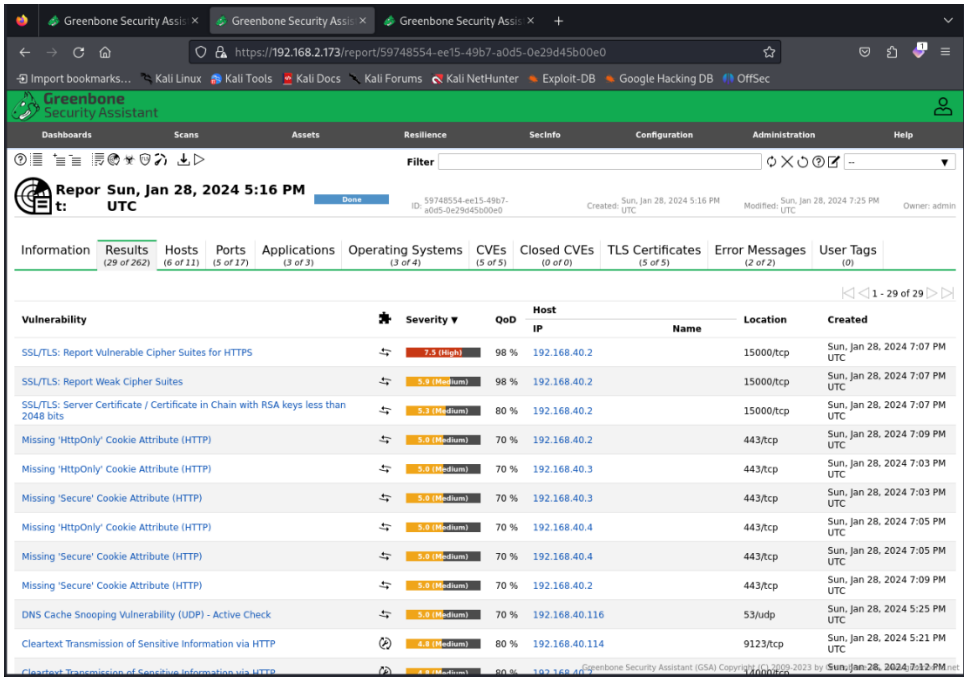


Figure 121 – Scan Report

21. Select the desired format (HTML, XML, CSV, etc.) and download the report for offline analysis or for sharing with stakeholders.
22. Address the reported vulnerabilities by applying patches, changing configurations, or implementing compensating controls.
23. Document the remediation steps and update your security posture documentation accordingly.
24. After remediation, re-run the scan to verify that the vulnerabilities have been resolved.
25. Set up a recurring scan schedule to regularly assess the security state of your targets.
26. Regularly update the OpenVAS vulnerability feed with `sudo greenbone-feed-sync`.

27. Keep the OpenVAS software updated via your Linux distribution's package manager.
28. Document your findings and the steps taken during the scan.
29. Refine your scanning process, target definitions, and remediation procedures based on lessons learned.
30. Some optional advanced configurations include exploring advanced configurations such as setting up multiple scanners or segmenting scan targets.
31. Ensure that the OpenVAS VM is secured with appropriate firewall rules, strong passwords, and access controls.
32. Consider the impact of scanning on network and host performance, scheduling scans during low-usage periods if necessary.

Through this lab, you will gain comprehensive hands-on experience with OpenVAS for vulnerability scanning, from installation and configuration to scanning, reporting, and remediation. This experience is crucial for cybersecurity professionals responsible for maintaining an organization's defensive posture against the ever-evolving threat landscape. By thoroughly scanning for vulnerabilities across the attack surface, architects gain visibility to proactively address risks before exploitation.

Security configuration and patch management tools

Security configuration and patch management are vital practices within cybersecurity management to ensure systems are up-to-date and configured according to the best security policies. For the purpose of this lab, we will focus on using Ansible for Security Configuration Management and **Windows Server Update Services** (**WSUS**) for Patch Management within a virtual environment.

Lab 1: Security Configuration Management using Ansible

Ansible is an open-source automation tool that simplifies the process of configuring, managing, and deploying systems and applications across various environments. It is an agentless platform that uses SSH or WinRM to communicate with target machines, making it lightweight and easy to set up. Ansible uses a simple, human-readable

language called YAML to define automation tasks, known as playbooks. These playbooks describe the desired state of the systems and the steps required to achieve that state, enabling consistent and repeatable deployments. With Ansible, you can automate a wide range of tasks, including configuration management, application deployment, orchestration, and provisioning. It provides a vast library of pre-built modules and plugins that allow you to interact with different systems, services, and tools, making it highly extensible and adaptable to diverse environments. Ansible's idempotent nature ensures that tasks are executed only when necessary, minimizing the risk of unintended changes. Its push-based architecture and parallel execution capabilities enable efficient management of large-scale infrastructures. Whether you are managing a handful of servers or a complex multi-tier application stack, Ansible empowers you to streamline your IT operations, reduce manual efforts, and improve the reliability and scalability of your systems.

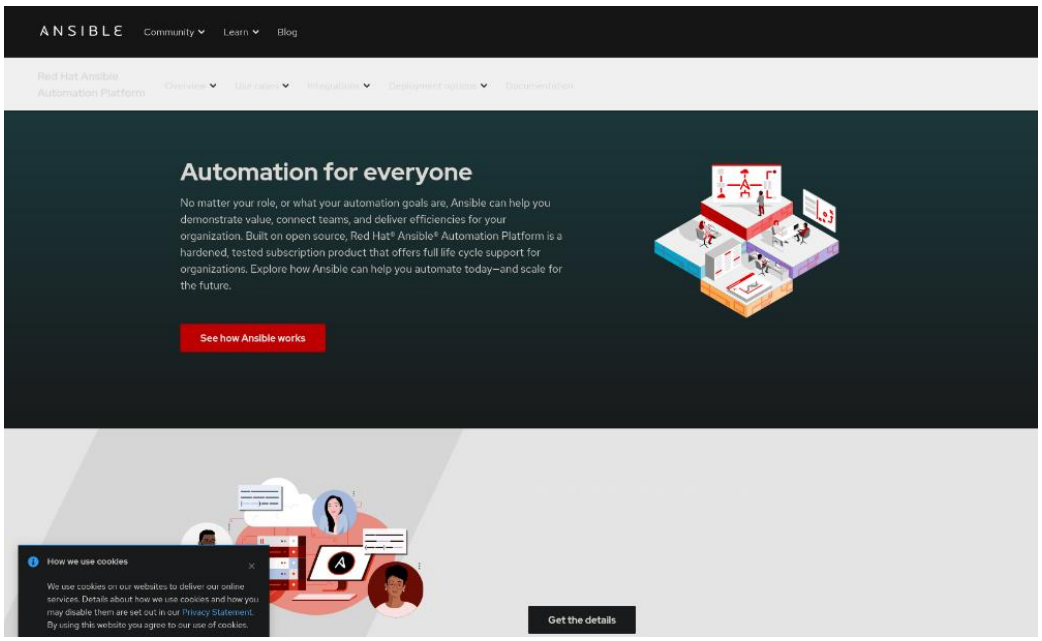


Figure 122 – Ansible Website

You can access this at https://www.ansible.com/.

The prerequisites include:

- A virtualization platform like VirtualBox or VMware.

- A Control VM (Ansible Control Node) running a Linux distribution with Ansible installed.
- One or more Target VMs (managed nodes) that you wish to configure.
- Network connectivity between the control node and the managed nodes.

Let us look at the steps.

1. Set up a Linux VM as the Ansible Control Node. Ansible is a powerful automation tool for managing multiple servers. Using Python and SSH, it can configure servers, routers, switches, and more from a central control node. This lab covers installing Ansible on Debian 12 **Bookworm** from the official repositories. First, update the system with the latest packages to enable installing Ansible. As a Debian stable release, Bookworm receives ongoing security updates but limited feature updates. Updating packages now ensures compatibility with the Ansible version available in the repos. With the system updated, we are ready to install Ansible and configure the control node. From this central management server, Ansible can then automate tasks across servers in the infrastructure through SSH.
2. Set up one or more Target VMs (Linux or Windows) that will be managed by Ansible.
3. Install Ansible on the Control Node using the package manager (for Ubuntu/Debian systems). After updating packages, install the Ansible package to set up the control node. This will install Ansible and all required dependencies on Debian 12. Additional packages may be needed in some use cases, but this covers the Ansible basics. With Ansible installed, the central management server can now automate tasks and configure infrastructure components including servers, routers, switches etc. through SSH connectivity:

```
sudo apt update && sudo apt upgrade -y

sudo apt install ansible
```

4. To verify Ansible installed correctly on Debian 12, check the version. This will print the Ansible version number along with Python and OpenSSL dependency versions. Reviewing the output confirms Ansible is present and accessible on the command line after installation completes:

```
ansible -version
```

5. To enable Ansible to connect to managed hosts, generate SSH keys on the Ansible control node and distribute the public key to each host. This creates public and private ssh key files on the Ansible server. Copy the public key to each managed host that Ansible will need to access, allowing passwordless SSH authentication between Ansible and managed nodes:

```
ssh-keygen
```

The server that will be managed through Ansible is:

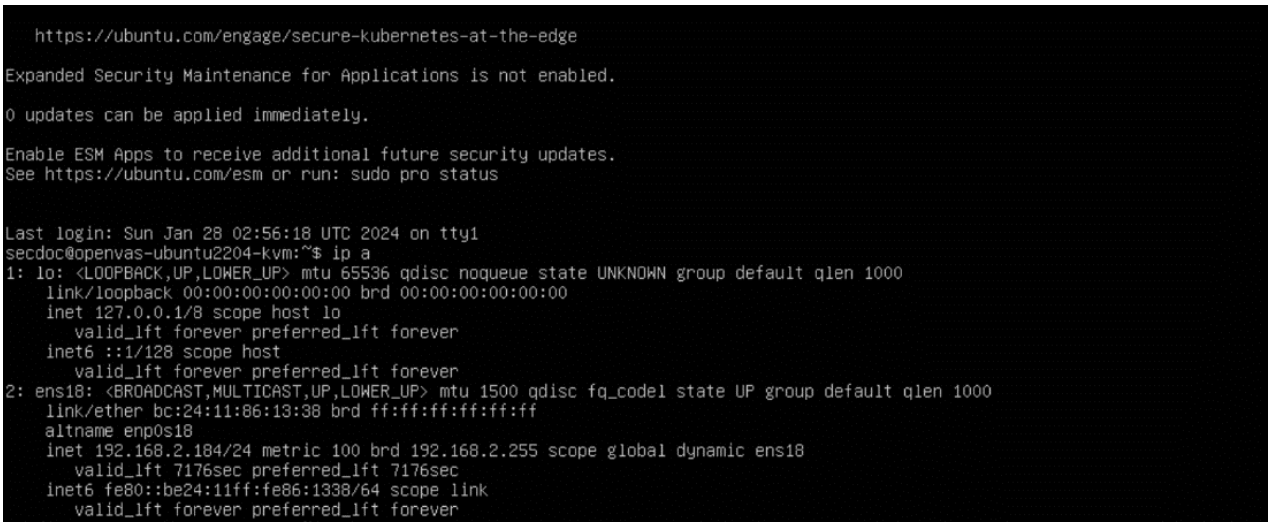


Figure 123 – ifconfig for Target System

6. Copy the public key to the Target Nodes using (in the lab, I am using an Ubuntu 22.04 server as the managed client):

```
ssh-copy-id
```

7. To allow Ansible to run `sudo` commands on managed hosts without prompting for a password, configure passwordless sudo access for the sysops user (or whichever admin user Ansible utilizes) on each host:

```
echo "secdoc ALL=(ALL) NOPASSWD:ALL" | sudo tee
/etc/sudoers.d/secdoc
```

NOTE

Make sure to replace `secdoc` with the user you created or are using on the ansible system.

8. Create a project directory for Ansible automation called **ansible** and change into it:

```
mkdir ansible

cd ansible
```

9. Next, create an `ansible.cfg` file by typing `nano ansible.cfg` to define the following core settings:

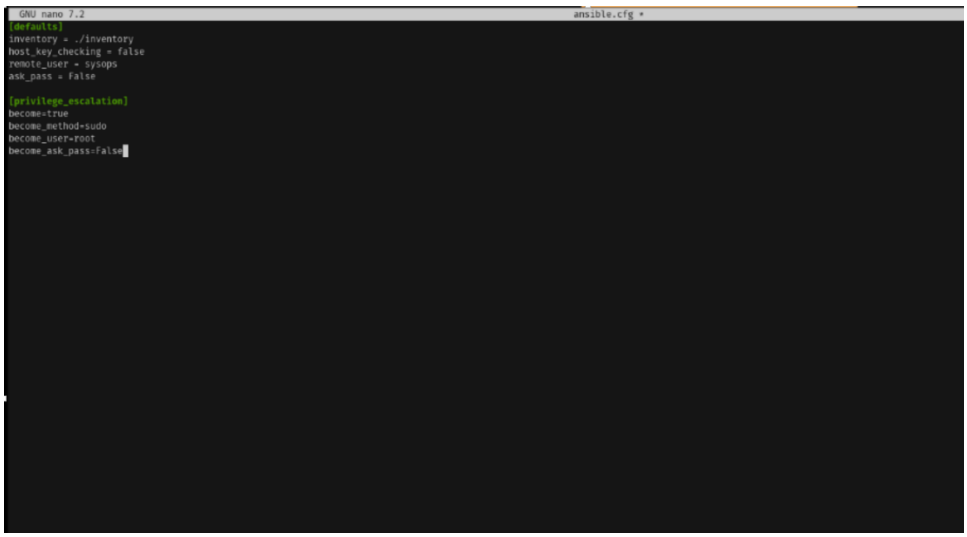


Figure 124 – Defining core settings

Here are the settings:

```
 [defaults]
inventory = ./inventory
host_key_checking = false
remote_user = sysops
ask_pass = False


[privilege_escalation]
become=true
become_method=sudo
become_user=root
become_ask_pass=False
```

10. Then press *CTRL+O* to save the file and then *CTRL+X* to exit.

    This specifies the inventory file location, disables host key checking, sets the default remote user, disables password prompting, and configures sudo privileges.

11. Finally, create an empty inventory file to define groups of managed hosts:

```
 [prod]


[dev]
192.168.1.184
```

12. Then press *CTRL+O* to save the file and then *CTRL+X* to exit.

13. The ansible directory now contains `ansible.cfg` and inventory files to automate managed nodes.

14. To test connectivity from the Ansible control node to managed hosts, run the following ad-hoc ansible ping command, `ansible all -m ping`.

15. This performs a basic ping test to each defined host. Getting a successful ping response confirms Ansible can reach the managed nodes. With basic connectivity verified, we can create a sample playbook to install an Nginx server on a dev node.

16. To create the installation config, enter the following by editing in Nano using `nano nginx.yml`:

```
---
- name: Install NGINX Web Server
  hosts: dev
  tasks:
  - name: install nginx package
    ansible.builtin.apt:
      name: nginx
      state: prese

  - name: Start nginx service
    service:
      name: nginx
      state: started
```

17. Then press *CTRL+O* to save the file and then *CTRL+X* to exit.

18. To execute the sample playbook:

```
ansible-playbook nginx.yml
```

19. This will install Nginx on the dev node based on the playbook instructions.

20. To verify Nginx was installed and started successfully:

```
ansible dev -m shell -a 'dpkg -l| grep -i nginx'

ansible dev -m shell -a 'systemctl status nginx'
```

Getting the expected output confirms the playbook achieved the desired state configuration on the target managed host.

Ansible is now fully operational for patch management, deployment automation, and orchestrating infrastructure from this central control server. Playbooks can automate regular sysadmin tasks across many machines in just minutes!

21. Document the configurations applied, any issues encountered, and remediation steps.
22. Update the playbooks as security policies evolve.

Let us move on to the next lab.

Lab 2: Patch Management with WSUS

WSUS (**Windows Server Update Services**) is a Microsoft tool that enables centralized management and distribution of updates and patches for Microsoft products, including Windows operating systems, Office applications, and other Microsoft software. It is designed to simplify the process of keeping systems up to date and secure within an organization's network. With WSUS, administrators can create a local repository of updates on a Windows server, which acts as a central point for managing and deploying updates to client computers. Instead of each client independently downloading updates from Microsoft's servers, they can retrieve them from the local WSUS server, reducing internet bandwidth usage and providing better control over the update process. Administrators can use WSUS to approve or decline specific updates, schedule update installations, and monitor the update status of client computers. WSUS integrates with Active Directory, allowing for the creation of computer groups and the application of different update policies based on organizational requirements. By using WSUS, organizations can ensure that their Microsoft systems are consistently patched, reducing the risk of vulnerabilities and maintaining a secure and stable computing environment.

Figure 125 – WSUS Website

You can access this at https://learn.microsoft.com/en-us/windows-server/administration/windows-server-update-services/get-started/windows-server-update-services-wsus.

Because of the nature of WSUS and the need for explicit Windows infrastructure, this is an exercise to help you understand the steps needed to deploy patching services within a Windows-based infrastructure. As part of this, it is recommended to read and understand the first step in the deployment of WSUS and to make important decisions regarding the deployment. As such I have provided the Microsoft link to assist in the planning: https://learn.microsoft.com/en-us/windows-server/administration/windows-server-update-services/plan/plan-your-wsus-deployment.

The prerequisites include:

- A virtualization platform like VirtualBox or VMware.
- Windows infrastructure and a Windows Server VM to act as the WSUS server.
- Windows Client VM(s) to manage patches for.

Let us look at the steps:

1. Install Windows Server on a VM and configure it with a static IP address.

2. Add the WSUS role through the Server Manager interface.
3. Configure the WSUS according to your organizational requirements, such as choosing which updates to download, when to download them, and which machines to apply them to.
4. On the Windows Client VMs, set the Group Policy or registry settings to point to the WSUS server for updates.
5. In the WSUS Administration Console, approve updates for the relevant computer groups.
6. Ensure that client VMs are checking in with the WSUS server and downloading the approved updates.
7. Use the WSUS Administration Console to monitor update deployment and client status.
8. Generate reports to track update coverage and identify any issues with deployment.
9. If updates fail to install, use logs from both the WSUS server and client machines to troubleshoot the issue.
10. Regularly clean up the WSUS database and declined updates to maintain WSUS server performance.
11. Ensure communication between the WSUS server and clients is secured.
12. Regularly update the WSUS server itself to protect against vulnerabilities.
13. Document the WSUS configuration settings, update policies, and any troubleshooting steps.

Through this lab and exercise, participants will gain hands-on experience with automating security configurations using Ansible and managing patches in a Windows environment using WSUS. These processes help ensure consistent application of security policies and timely deployment of important security updates.

Proactively hardening configurations and installing the latest patches blocks exploitation of known vulnerabilities. Architects must balance security with availability when planning change windows.

Incident response and forensics tools

Incident response and digital forensics are pivotal components of cybersecurity, focusing on addressing and managing the aftermath of security breaches or attacks and performing detailed investigations to understand the breach and recover from it. For the purpose of this lab, we will use **The Sleuth Kit** (**TSK**) and Autopsy for digital forensics analysis, and Security Onion as an incident response platform, within a virtualized environment.

Lab 1: Digital Forensics with The Sleuth Kit and Autopsy

Digital forensics is a critical field in today's digital age, and The Sleuth Kit (TSK) and Autopsy are two powerful open-source tools that have become essential in the forensic investigation of digital devices. The Sleuth Kit is a collection of command-line tools and libraries that allow investigators to analyze disk images and recover deleted files, while Autopsy is a graphical interface that sits on top of TSK, providing a user-friendly environment for conducting forensic investigations. Together, these tools enable forensic examiners to perform a wide range of tasks, including data recovery, file system analysis, timeline creation, and artifact examination. With support for various file systems and the ability to handle large volumes of data, TSK and Autopsy have become the go-to tools for digital forensics in law enforcement, corporate investigations, and incident response. Whether investigating a computer intrusion, analyzing digital evidence in a criminal case, or conducting internal investigations, TSK and Autopsy provide the necessary capabilities to uncover digital traces, reconstruct events, and gather crucial evidence to support the investigation process.

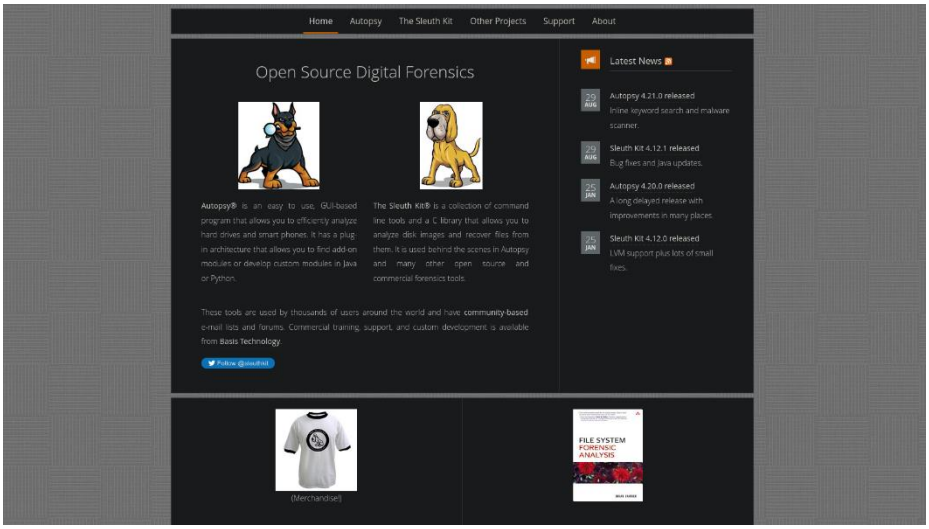


Figure 126 – The Sleuth Kit (TSK) and Autopsy Website

The prerequisites are:

- A virtualization platform like VirtualBox or VMware.
- A forensic workstation VM with a Linux OS or Security Onion which includes TSK and Autopsy.
- A disk image or a VM snapshot to investigate, simulating a suspect system.
- Network connectivity between your forensic workstation VM and other devices if remote analysis is required.

Let us look at the steps:

1. Set up a VM to serve as your forensic workstation. This could be a Linux distribution with digital forensics tools installed or a specialized distro like Security Onion. In this lab I will be using the SANS SIFT Workstation as the system to install the tools. The SIFT Workstation is available as an OVA download or with specific installation instruction at the following SANS URL: https://www.sans.org/tools/sift-workstation/.

2. Obtain a disk image (e.g., E01, dd, aff) or VM snapshot for analysis. Ensure legal permission for analysis if required. In this example I will be using a Windows Server image (WinServer.dd).

3. On a Linux VM, install The Sleuth Kit using the package manager with `sudo apt-get install sleuthkit`:

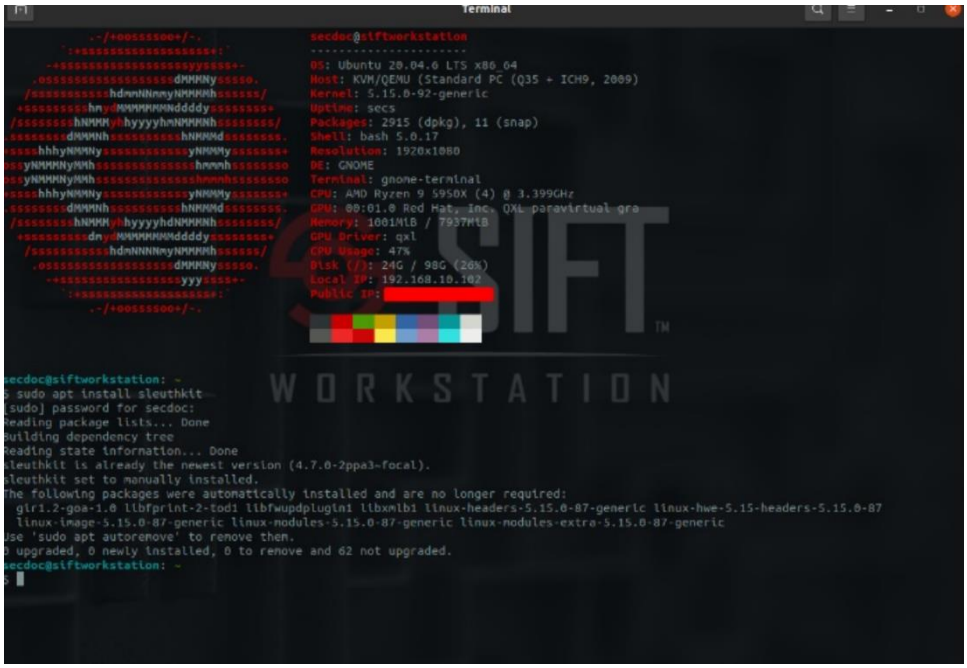
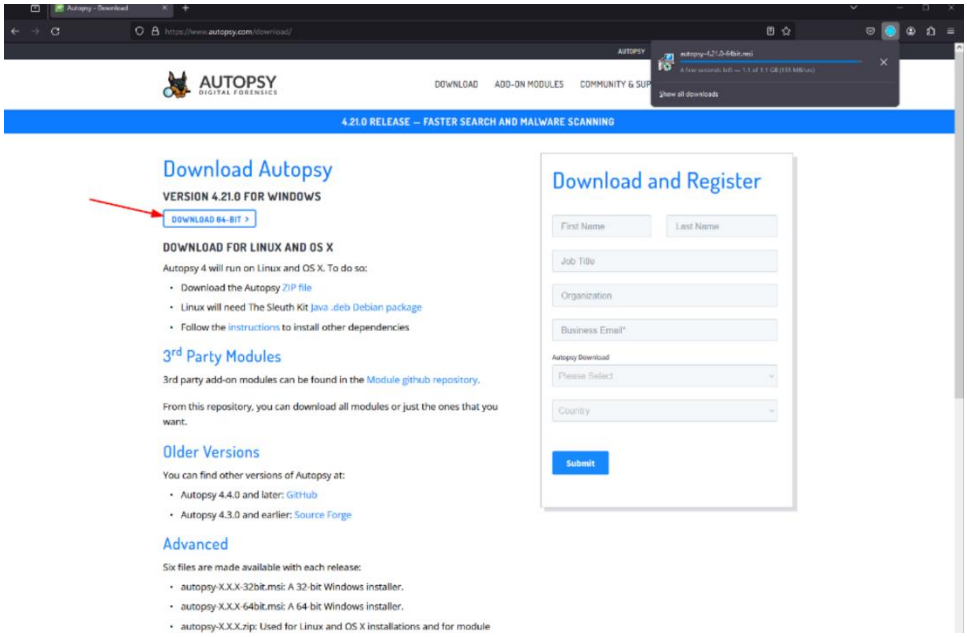


Figure 127 – Sleuthkit Install

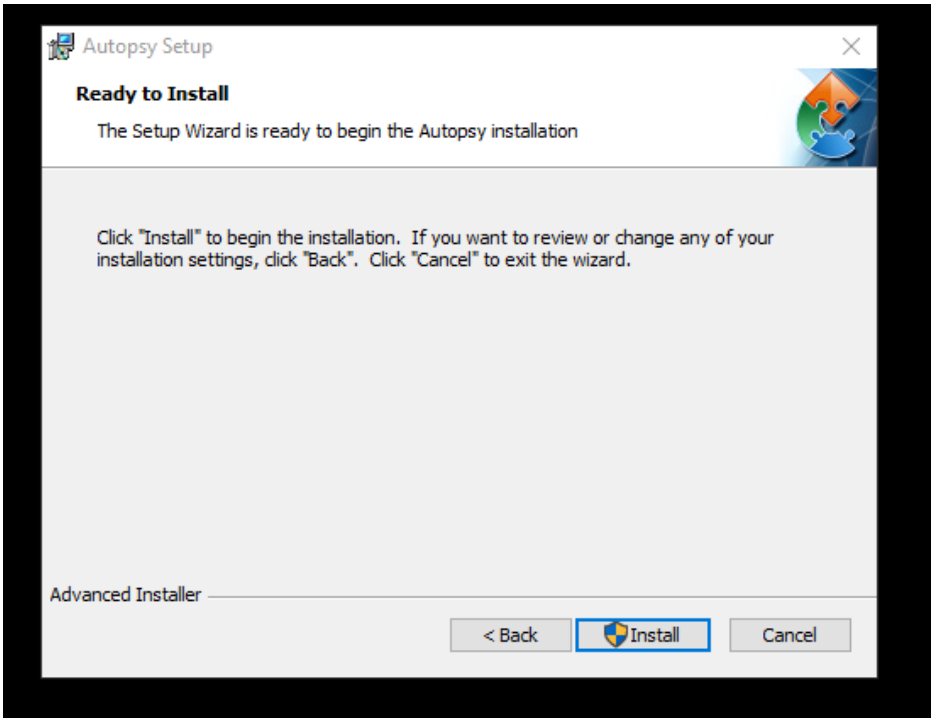
Note that Sleuthkit is a cli/terminal only application.

4. Install Autopsy by downloading the latest version from the official website, https://www.sleuthkit.org/, and following the installation instructions. Autopsy is installed on Windows:



(a) Autopsy Download

(b) Autopsy Installation Wizard



(c) Autopsy Installation

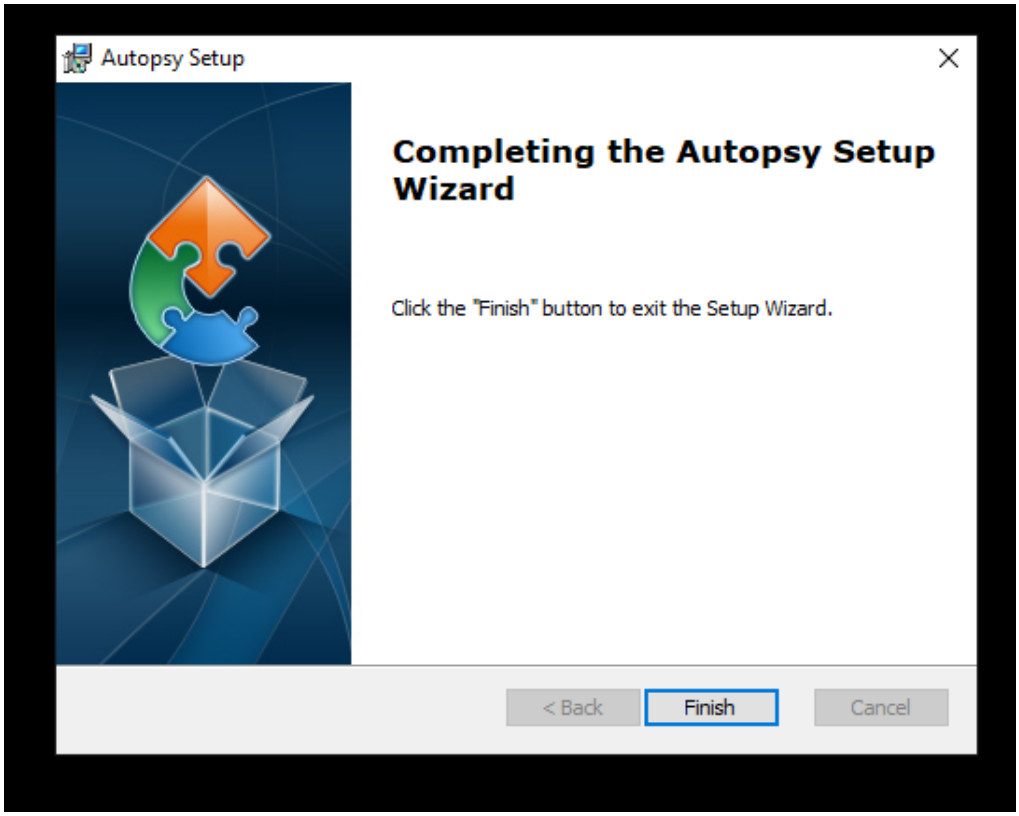
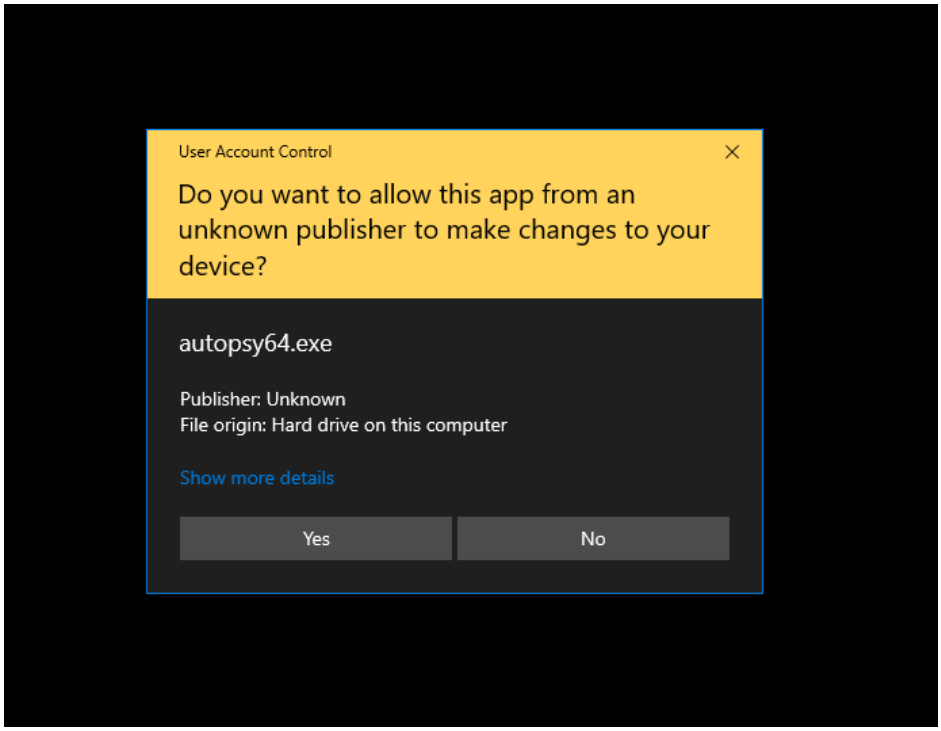


Figure 128 – Autopsy Installation Completion

5. Open Autopsy and create a new case, providing all the necessary case details. It is important to note that you should run Autopsy as Administrator:



(a) UAC Initial Autopsy Start

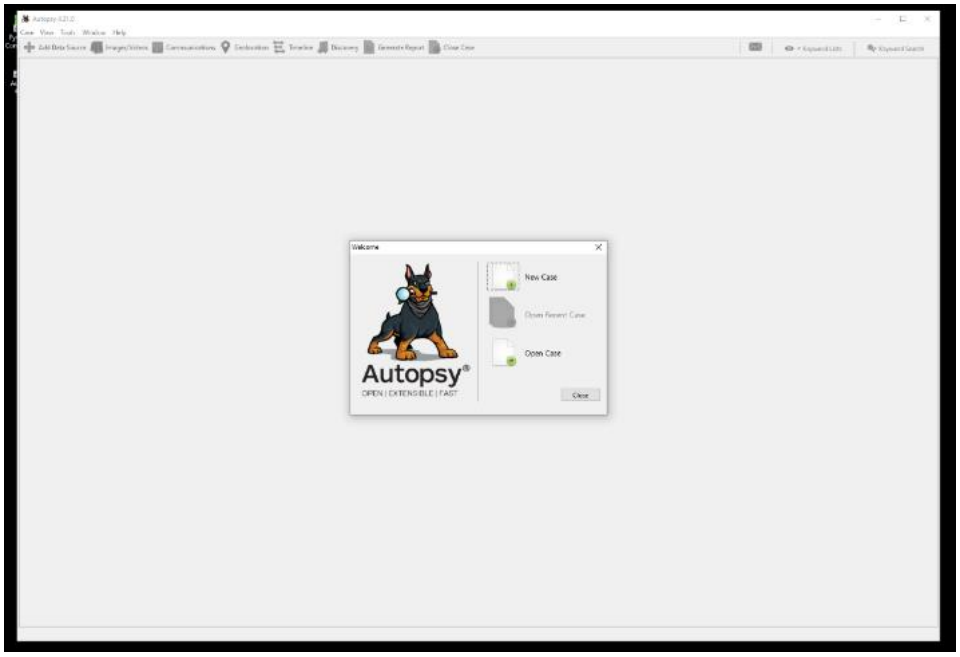
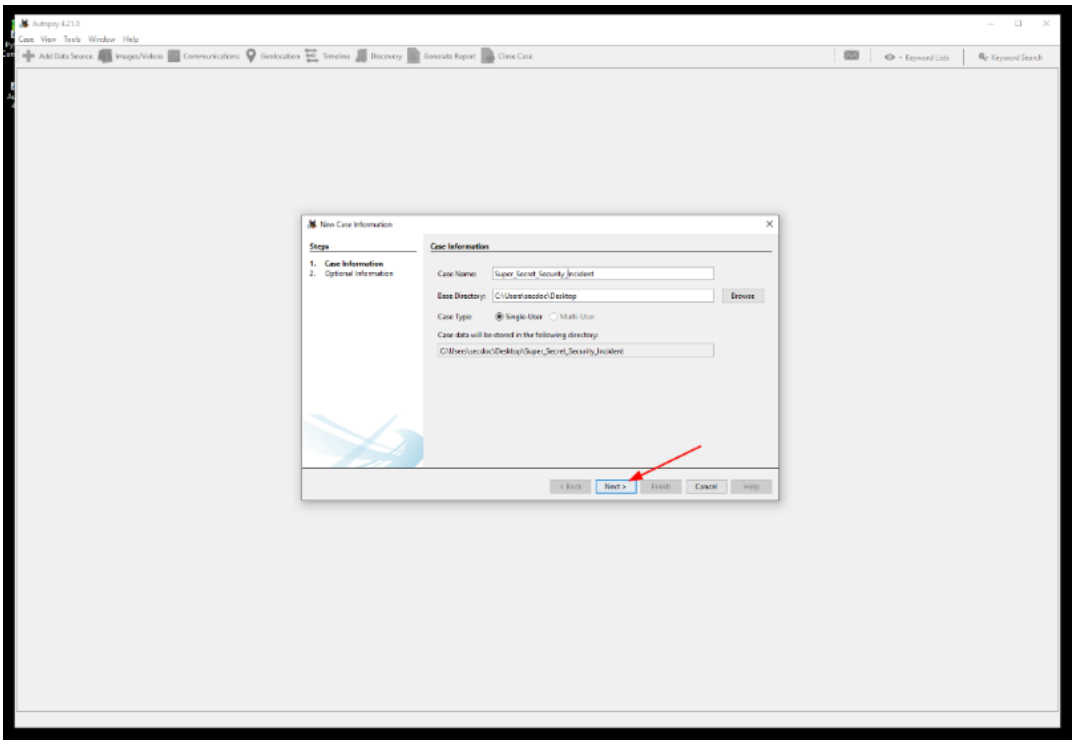


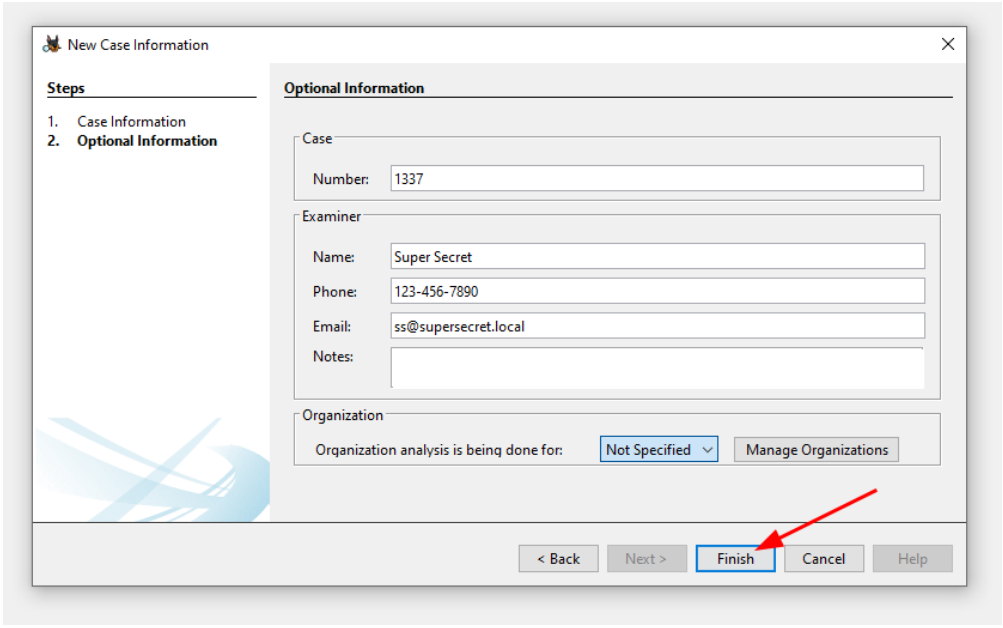
Figure 129 – Autopsy Default Dashboard

6. Add a new host to the case and select the disk image or local drive to investigate. Follow the prompts to complete case creation:

(a) Selecting New Case



(b) Autopsy Case Default Location



(c) Case Information

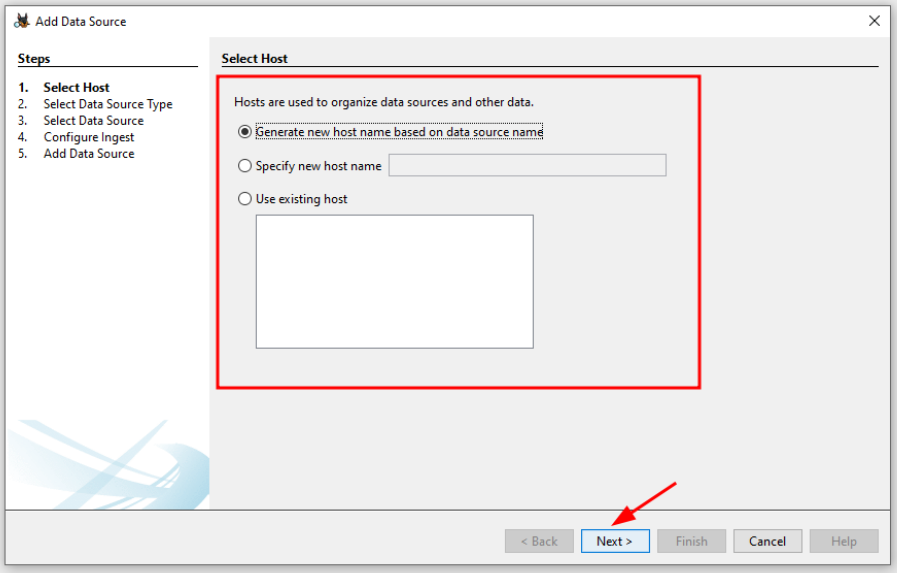
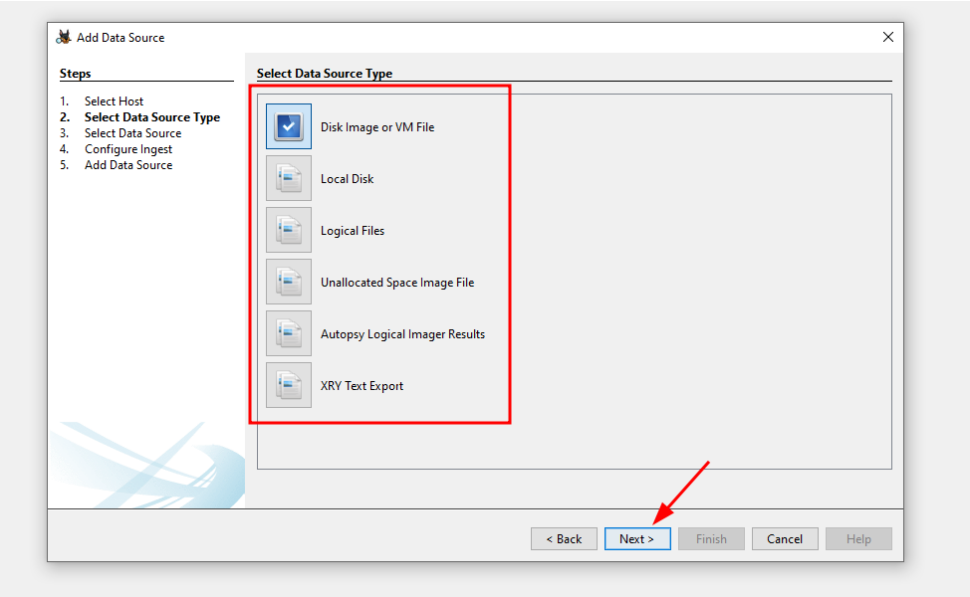
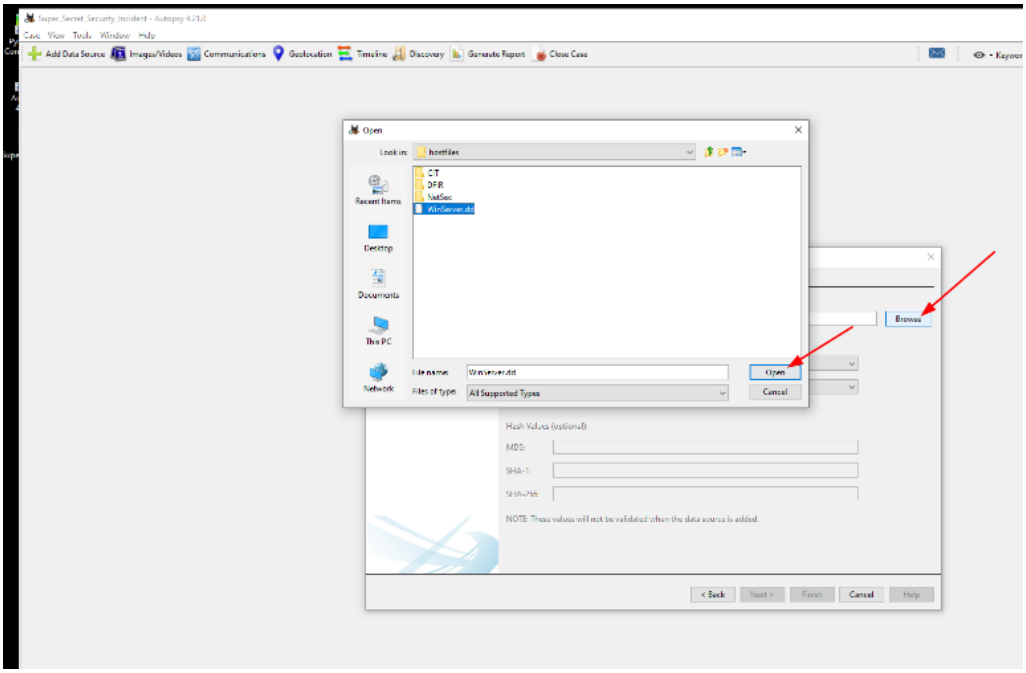


Figure 130 – Selecting Host

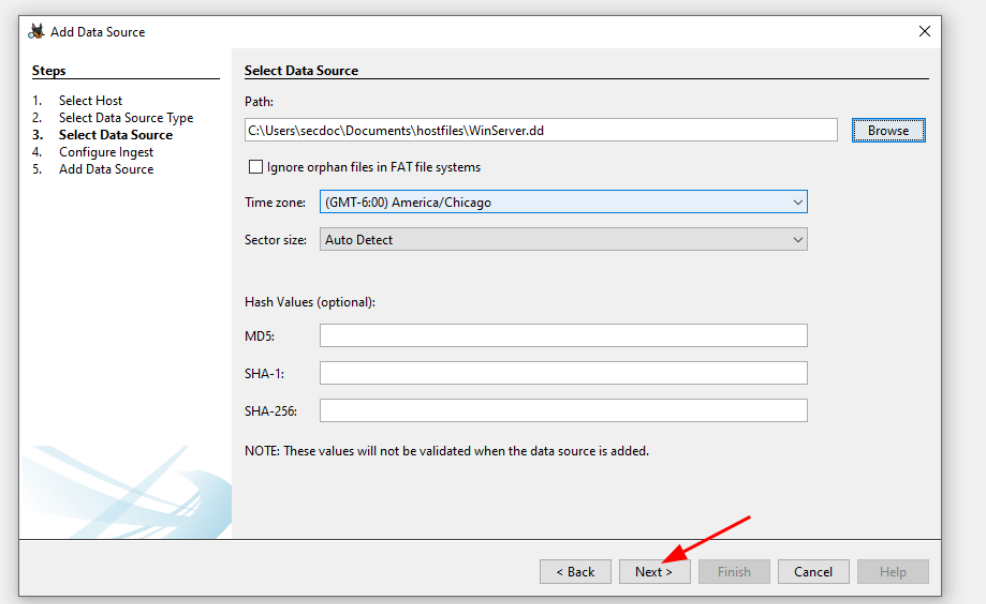
7. If working with a live system, use TSK or another acquisition tool to create an image of the suspect system's disk:



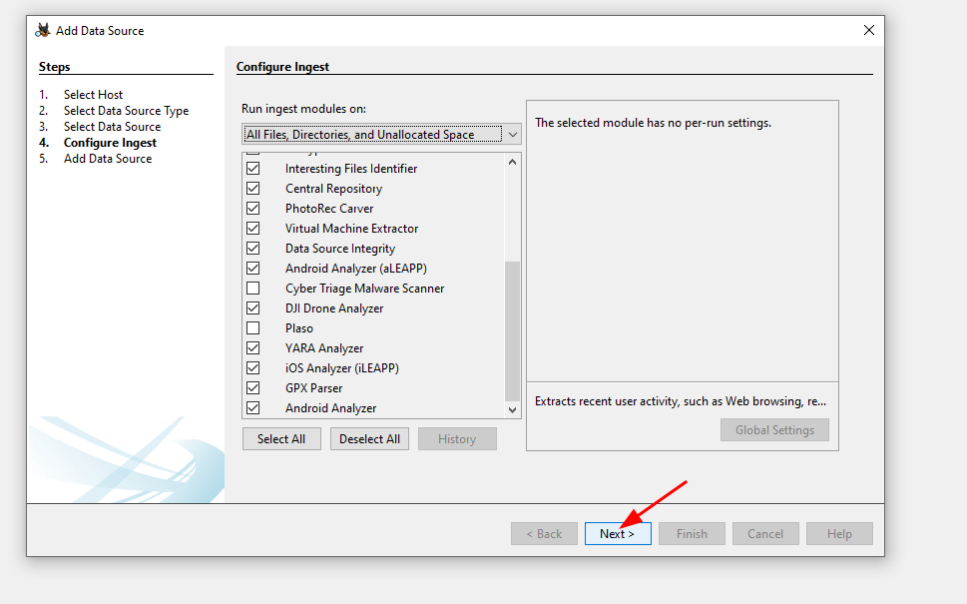
(a) Selecting Data Source



(b) Selecting Drive Image



(c) Data Source Validation



(d) Data Ingestion Configuration

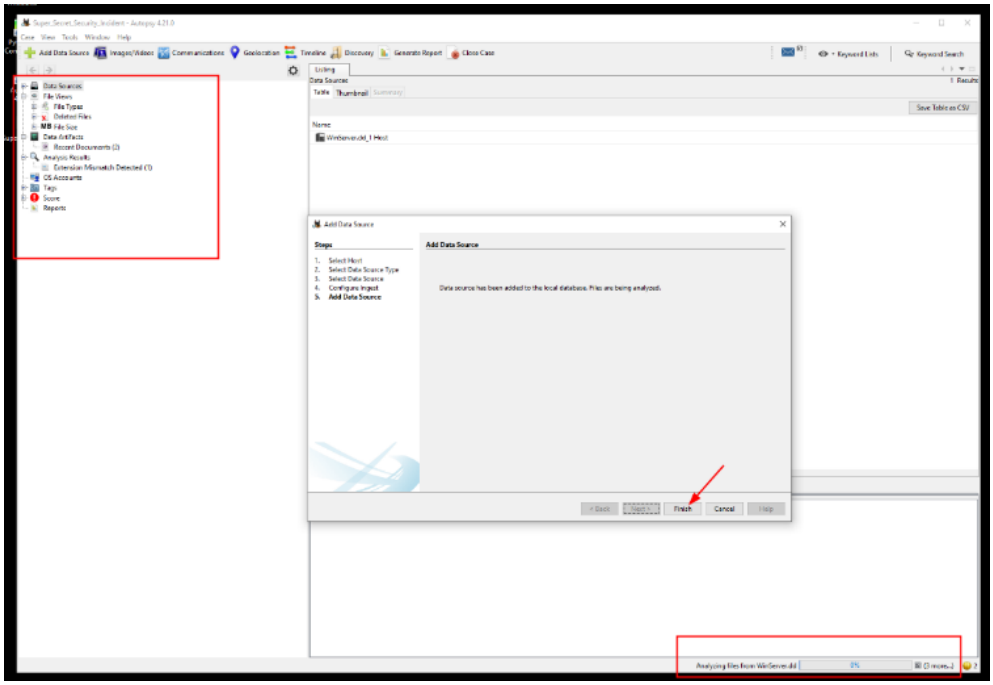


Figure 131 – Add Data Source

8. Navigate through the file system using Autopsy's interface to look for any obvious signs of compromise or artifacts of interest.

9. Use the keyword search and timeline analysis features to locate files or system activity relevant to the time frame of the incident:

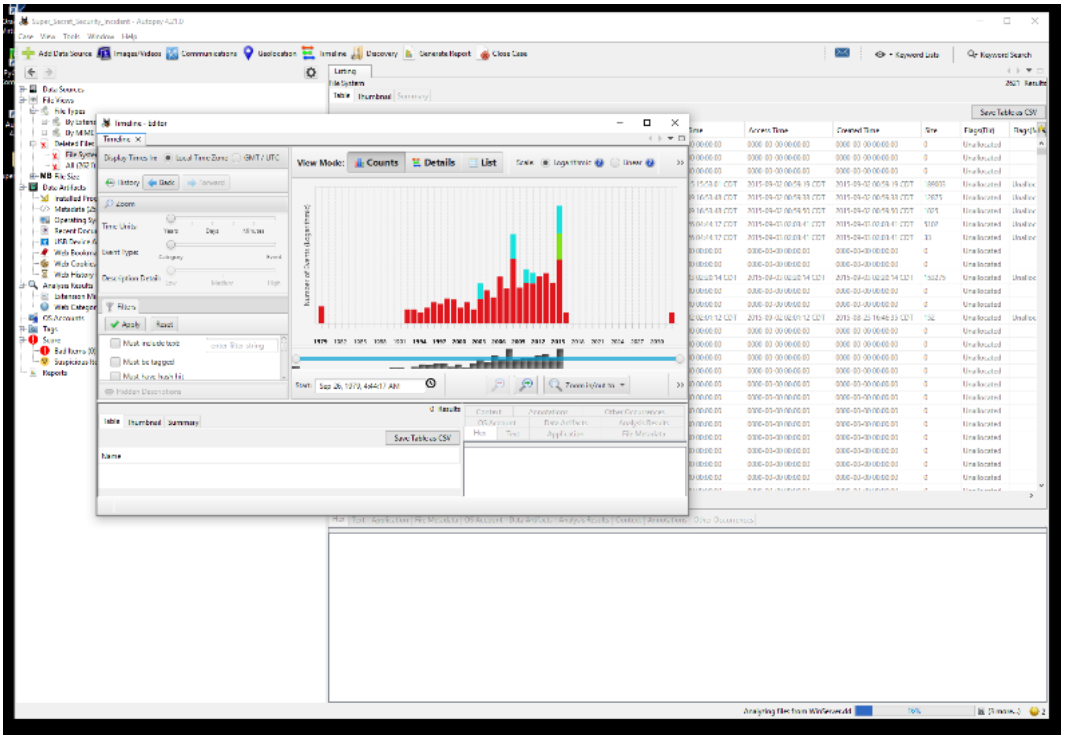


Figure 132 – Timeline Analysis

10. Now, let's do a deep dive with TSK. When a forensic investigator arrives at an incident scene, they first seize all relevant evidence and capture volatile data at risk of loss when power is removed. The evidence is then transported to a forensics lab, taking care not to alter the original data.

    At the lab, an identical copy of the evidence is created to prevent tampering of the original. Changes are documented and chain of custody is recorded detailing all personnel who handled evidence. This log is included in the final report to demonstrate integrity.

11. Use command-line tools from TSK such as fls, istat, icat, and mmls to analyze the file system structure, recover deleted files, and examine file metadata. To create a bit-for-bit forensic duplicate of a hard drive and generate an MD5 checksum,

run this command on Kali Linux: `dcfldd if=/dev/sdb1 hash=md5 of=/media/[filename].dd bs=512 noerror`.

Here:

- `if=/dev/sdb1` specifies the source drive to image
- `hash=md5` calculates an MD5 hash of the image for verification
- `of=/media/[filename].dd` is the output image file saved externally
- `bs=512` transfers 512 bytes of data at a time
- `noerror` continues reading on errors by writing zeros

This command images the entire disk bit-for-bit to an external drive, saving it as a `.dd` file while hashing the image. The external drive must have greater capacity than the source drive. The `noerror` option can be omitted if preferred. As I already have an image to analyze, this step will not be necessary.

12. To create a hash execute `md5sum [filename].dd`.
13. Use the `mmls` command to get details of the image file including the partition layout information:

```
mmls [filename].dd
```

14. After obtaining a verified image, the next step is analyzing the evidence to investigate the incident. This typically involves recovering data, extracting hidden files, and accessing protected content if technically possible and legally permitted. The fsstat tool provides extensive file system details. An offset of 2048 is used to skip either unallocated space or the partition table area. Other regions can also be examined. Hard disks often contain hidden host protected areas with vendor utilities that could conceal data if aware of their existence. The `disk_stat` Sleuth Kit tool detects if a HPA is present on the disk:

```
fsstat -o 2048 [filename].dd
```

15. Use the fls tool to view files and directories in the file system. Deleted files are denoted with a `*` prefix, like `* eula.2052.txt`. To examine a specific directory, reference it by its inode number from the fls output:

```
fls -o 2048 [filename].dd
```

16. You can use icat to read files as part of the investigation. In addition, if there are multiple files associated with the same inode, you can pipe the command to less to read them in a more manageable way:

```
icat -o 2048 [filename].dd [inode number] | less
```

    The command `icat -o 2048 [filename].dd [inode number] | less` is used in digital forensics to read the contents of a specific file associated with a given inode number from a disk image, and display the output in a more manageable way using the `less` command.

    This command is particularly useful when dealing with multiple files associated with the same inode number. In some cases, a single inode may have multiple file names or entries pointing to it, such as in the case of hard links. By using `icat` with the inode number, you can read all the files associated with that inode.

    Using `less` to view the output makes it easier to examine the file contents, especially if the file is large or contains a lot of information. You can scroll up and down, search for specific text, and navigate through the content more efficiently.

    It's important to note that the specific options and parameters used with `icat` may vary depending on the version of TSK you are using and the specifics of your investigation. Always refer to the documentation and guidelines provided by TSK for the most accurate and up-to-date information on using `icat` and other TSK tools in your forensic investigations:

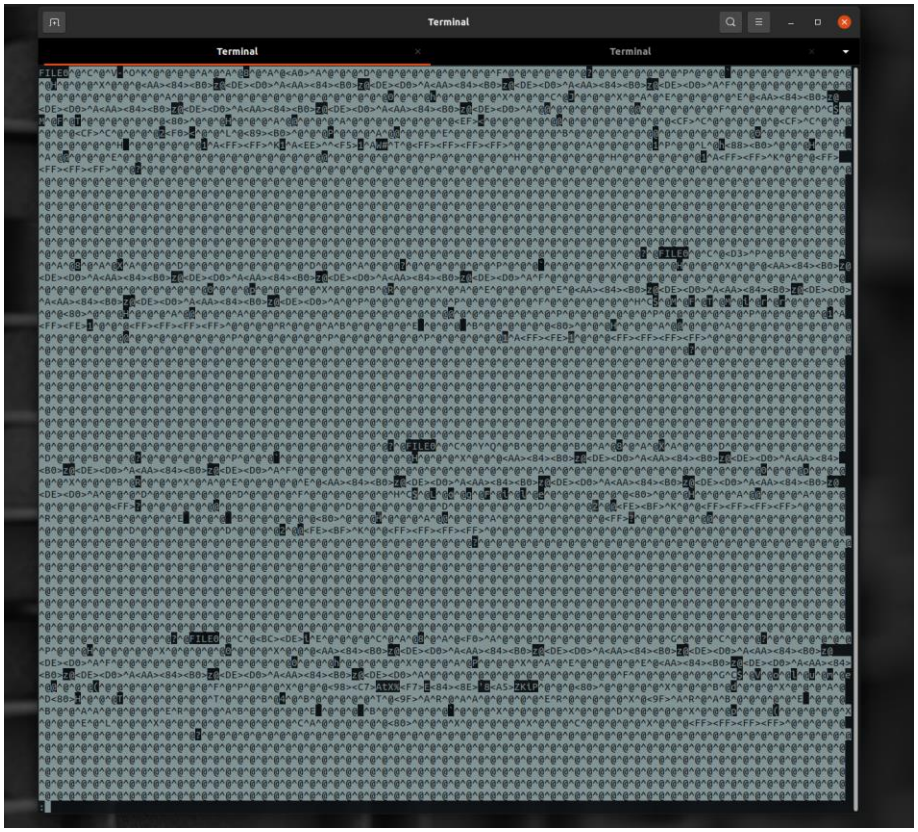


Figure 133 – inode View of Drive dd Image

17. Additional Sleuth Kit tools for metadata analysis include `ifind` and `ffind`, which locate files by searching for text string matches. Beyond keyword searches, file signature analysis is another common approach to identify files pertinent to the case by their headers rather than extensions. Focusing on relevant file types aids the investigation.

18. Document findings and extract evidence using icat to retrieve file contents based on inode numbers.

19. Document every step taken during the forensic investigation, noting how evidence was preserved and analyzed. The forensic report is the concluding

deliverable presented to the party who initiated the investigation. It contains only objective facts pertinent to the case, including at minimum:

- An executive summary outlining the case background and key findings.
- An analysis summary fully detailing evidence gathering and examination methodology. This comprehensively covers all analysis steps taken by the investigator to prove or disprove foundational allegations.
- A final summary that recapitulates closing statements and ultimate conclusions.

20. The report relays pure evidence-based conclusions, free of subjective interpretations. The process transparency provided by an exhaustive analysis summary ensures integrity that stands up to legal scrutiny.
21. Generate reports using Autopsy, including file lists, timelines, and hashes of relevant files.
22. Secure the evidence and ensure any copies of disk images are stored securely or deleted if no longer needed.
23. Reset the forensic workstation to a clean state if it will be used for future investigations.

Let us look at the next lab.

Lab 2: Incident Response with Security Onion

Security Onion is a powerful open-source intrusion detection, enterprise security monitoring, and log management solution that provides a comprehensive platform for detecting, investigating, and responding to security threats. Built on a foundation of industry-leading open-source tools, such as Zeek (formerly Bro), Suricata, Wazuh, and Elastic Stack, Security Onion offers a unified and scalable approach to network security monitoring. It combines **network intrusion detection systems** (**NIDS**), HIDS, full packet capture, and advanced analytics to give security professionals deep visibility into their network traffic and system activities. With its intuitive web-based interface, Security Onion allows users to monitor alerts, investigate suspicious activities, and

perform forensic analysis, empowering them to quickly identify and respond to potential security incidents. Whether deployed in a single server or distributed across multiple nodes, Security Onion provides a robust and flexible solution for organizations of all sizes looking to enhance their security posture and defend against evolving cyber threats.

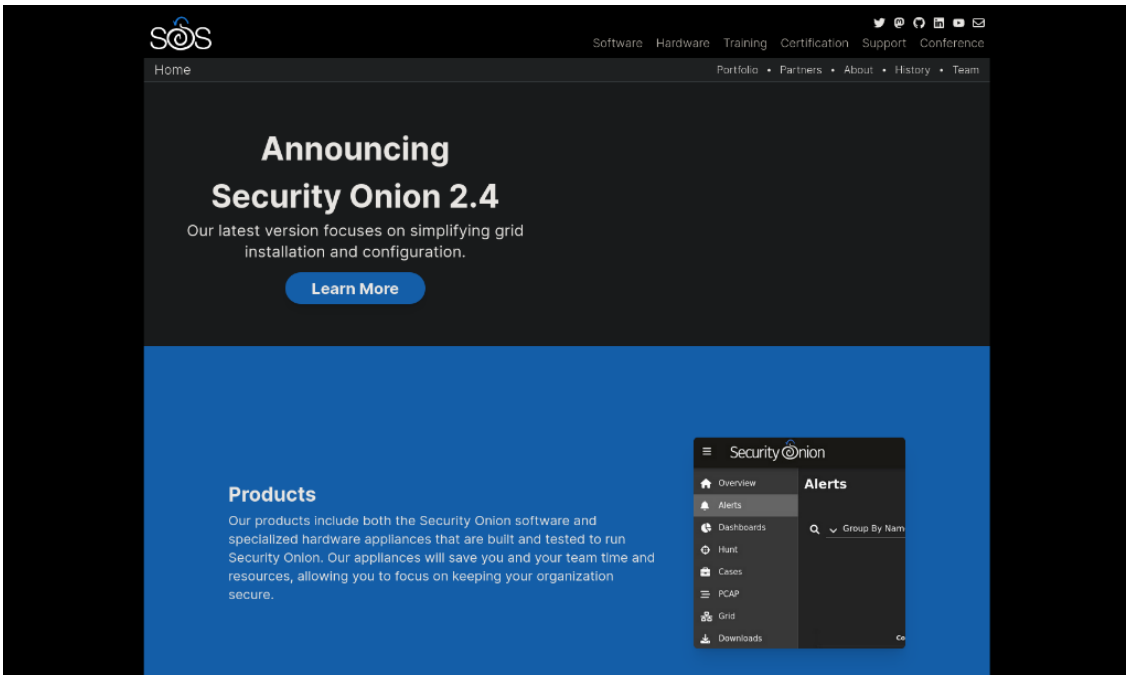


Figure 134 – Security Onion Website

The prerequisites are:

- A virtualization platform like VirtualBox or VMware.
- A VM to serve as the Security Onion monitoring server. If your only goal is to import pcaps using `so-import-pcap`, you can set up Security Onion 2 as an Import Node with these minimum specifications:
  - 4GB RAM
  - 2 CPU cores
  - 200GB storage

- For any other deployment beyond a basic import node, Security Onion 2 requires at minimum:
    - 12GB RAM
    - 4 CPU cores
    - 200GB storage
- Additional VMs or physical machines to serve as clients/network devices to be monitored.

Let us look at the steps:

24. Install Security Onion on a VM, following the official installation guide (Security Onion Uses Oracle Linux 9 as the standard Linux Distro used when installing via the ISO):

Figure 135 – Security Onion installation dialog box

Note

The installation and configuration of Security Onion can take a considerable amount of time as it installs and sets up all the various modules/packages, so do not be discouraged if it is taking a long time.

Here's a look at Security Onion's login screen:

Figure 136 – Security Onion Login

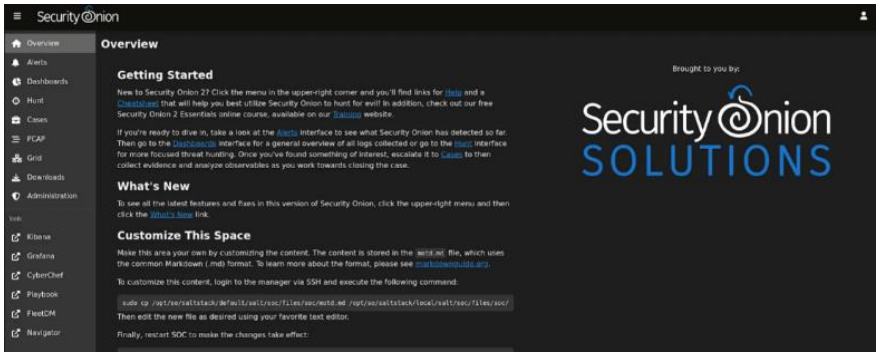


Figure 137 – Security Onion home screen

25. Configure network settings to allow the Security Onion VM to monitor traffic (e.g., in promiscuous mode or connected to a span/mirror port).

26. Deploy **network intrusion detection systems** (**NIDS**) like Snort or Suricata on Security Onion to monitor network traffic:

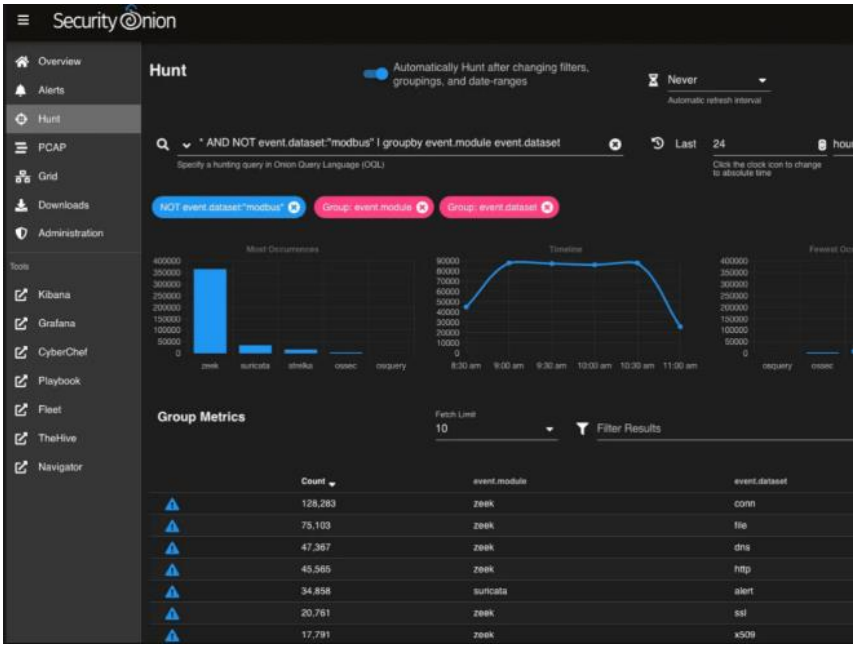


Figure 138 – Security Onion Hunt Dashboard

27. Configure log management and SIEM tools like Squert, Kibana, and TheHive for alert management and analysis. It is important that you review and use the Security Onion documentation at https://docs.securityonion.net/en/2.4/first-time-users.html through the rest of this lab. The complexity of setting up the initial Security Onion configuration can vary from environment to environment and as such, being able to walk through the documentation versus the verbatim steps in this lab will be helpful for you for a successful installation:

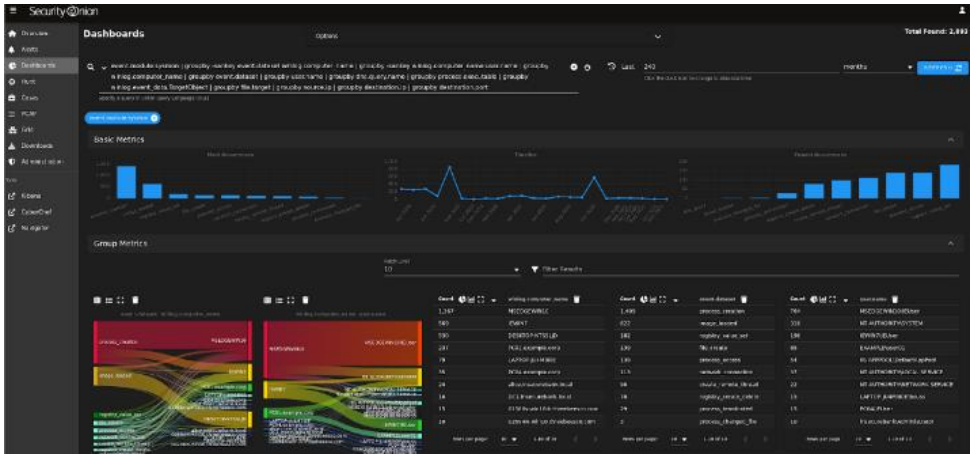


Figure 139 – Threat Dashboard

28. Generate or simulate suspicious network traffic that could indicate a security incident (e.g., using Metasploit or other penetration testing tools).
29. Review the alerts generated by the NIDS on Security Onion.
30. Use Security Onion's tools to analyze the data and determine the nature of the incident.
31. Practice containment strategies to limit the impact of the incident.
32. Follow an eradication process to remove the threat from the environment.
33. Document the incident response process, noting all alerts, actions taken, and lessons learned.
34. Generate reports detailing the incident and response actions for future reference.
35. Use lessons learned to improve the incident response plan and security posture.
36. Ensure that all monitoring tools are reset and ready for future incident detection.

Through these labs, participants will gain valuable experience in both the technical and procedural aspects of digital forensics and incident response. The iterative process of collection, examination, analysis, and reporting is essential for effective cyber investigation and response.

By combining investigative tools and platforms, security teams can rapidly analyze incidents, determine impact, and drive recovery while preserving evidence.

Application security tools

Application security tools are essential in identifying and mitigating security vulnerabilities within software applications. The exercise will look at SonarQube and will focus on **static application security testing** (**SAST**). In this lab, **dynamic application security testing** (**DAST**) using OWASP ZAP as well. Another popular tool similar to ZAP is Burp Suite. Packt has a great book on Burp Suite called *Hands-on Application Penetration Testing with Burp Suite* by Carlos A. Lozano, Dhruv Shah, and Riyaz Ahemed Walikar.

Exercise: SAST with SonarQube

SAST (**Static Application Security Testing**) with SonarQube is a powerful approach to identifying and mitigating security vulnerabilities and code quality issues early in the software development lifecycle. SonarQube is an open-source platform that performs static code analysis, which means it examines the source code without executing it, to detect potential security flaws, bugs, and code smells. By integrating SonarQube into the development process, teams can continuously scan their codebase, receive immediate feedback on code quality and security, and track the progress of issue resolution. SonarQube supports a wide range of programming languages and integrates seamlessly with popular development tools and CI/CD pipelines. It provides an intuitive web interface that displays code metrics, highlights issues, and offers detailed insights into the health and security of the codebase. With its extensive rule sets and customizable quality profiles, SonarQube helps organizations enforce coding standards, adhere to best practices, and maintain a high level of code quality. By leveraging SAST with SonarQube, development teams can proactively identify and address security vulnerabilities, reduce technical debt, and deliver more secure and reliable software.

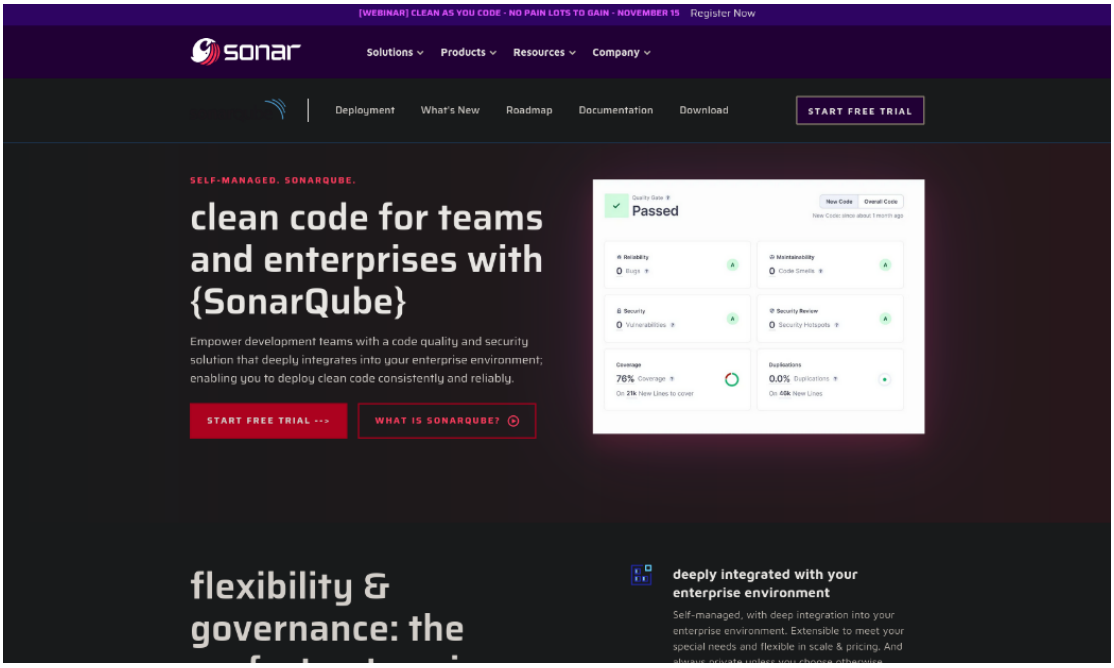


Figure 140 – SonarQube Website

The prerequisites are:

- A virtualization platform like VirtualBox or VMware.
- A virtual machine (VM) or a physical machine to serve as the analysis environment.
- The source code of the application to be analyzed.

Let us now look at the steps:

1. Set up a Linux VM which will be used to host the SonarQube server.
2. Ensure the VM has enough resources (CPU, memory, and storage) to perform analysis.
3. Install SonarQube on the VM following the official documentation, https://www.sonarsource.com/products/sonarqube/.

4. Make sure to install the required database (e.g., PostgreSQL) and connect it with SonarQube.
5. Access the SonarQube web interface through the browser on the host machine using the VM's IP address and port 9000.
6. Configure the quality profiles and rules according to the application's technology stack.
7. Install SonarScanner on the development machine where the application's source code resides.
8. Run SonarScanner against the source code repository to start the analysis.
9. Monitor the progress via the SonarQube dashboard.
10. Once the analysis is complete, review the issues flagged by SonarQube.
11. Use the detailed descriptions provided to understand the context and potential impact of each issue.
12. Address the reported issues in the application's source code.
13. Rerun SonarScanner to ensure the fixes were effective and did not introduce new issues.
14. This is optional: Integrate SonarQube with a **continuous integration** (**CI**)/**continuous deployment** (**CD**) pipeline for automated scanning on code commits.
15. Document the SAST process, findings, and remediation actions for compliance and audit purposes.

Lab: DAST with OWASP ZAP

DAST (**Dynamic Application Security Testing**) with OWASP ZAP is a powerful technique for identifying security vulnerabilities in web applications by actively interacting with them in a runtime environment. OWASP **ZAP** (**Zed Attack Proxy**) is a popular open-source web application security scanner that helps developers and

security professionals find and fix security issues in their applications. With OWASP ZAP, you can perform automated and manual security testing, mimicking the actions of a potential attacker to uncover vulnerabilities such as SQL injection, **cross-site scripting** (**XSS**), and broken authentication. By sending carefully crafted requests to the application and analyzing the responses, OWASP ZAP can detect security flaws that may not be apparent through static code analysis alone. It provides a user-friendly interface, allowing users to configure and customize the scanning process, set up authentication, and define the scope of the testing. OWASP ZAP generates detailed reports highlighting the discovered vulnerabilities, their severity, and recommendations for remediation. With its extensive community support, regular updates, and a wide range of plugins and add-ons, OWASP ZAP is a versatile and essential tool for conducting DAST and ensuring the security of web applications.

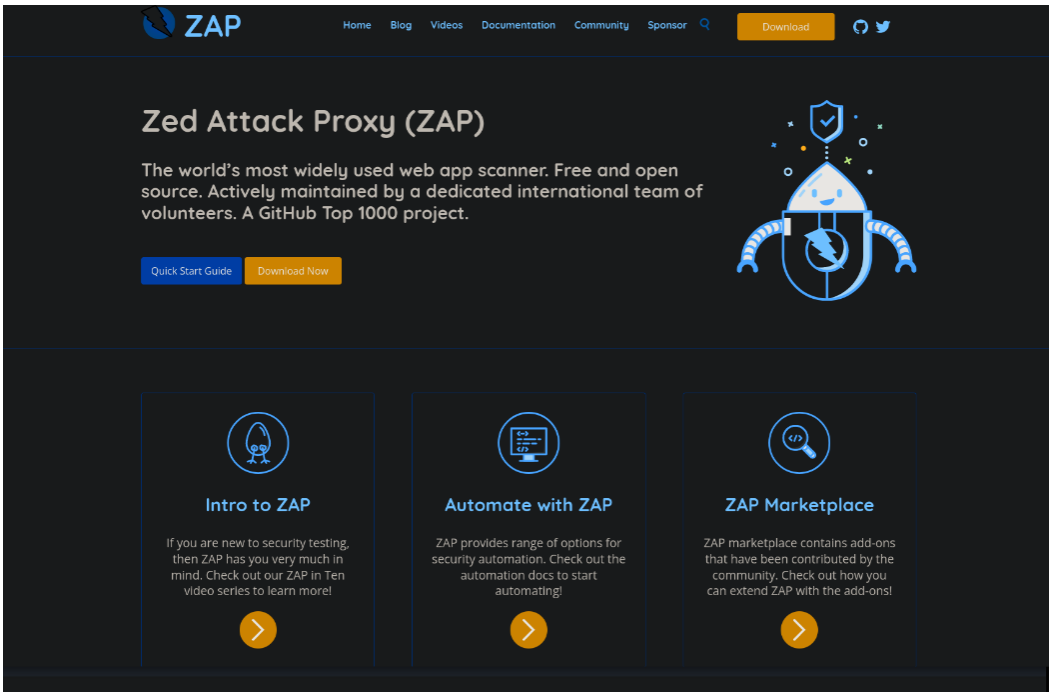


Figure 141 – OWASP ZAP

The prerequisites include:

- A virtualization platform like VirtualBox or VMware.
- A VM to serve as the testing environment with OWASP ZAP installed.

- An instance of a web application to test, which could be hosted within a VM or accessible over the network.

Let us look at the steps:

1. Set up a VM with a suitable operating system (e.g., Kali Linux which includes OWASP ZAP). You can also step through the **Getting Started** documentation at the OWASP ZAP site using the following link: https://www.zaproxy.org/getting-started/.
2. If not pre-installed, download and install OWASP ZAP from the official website, https://www.zaproxy.org/.
3. Ensure the target web application is running and accessible from the ZAP VM.
4. On initial startup, ZAP asks if you want to persist sessions. By default, ZAP saves testing sessions into HSQLDB database files locally without persistence enabled. Selecting this option preserves your session data for later access instead of deleting it upon exiting ZAP. Enabling persistence allows providing custom names and file paths to control where databases are saved. Without persistence, default unnamed database files generated at default locations are removed when closing ZAP:

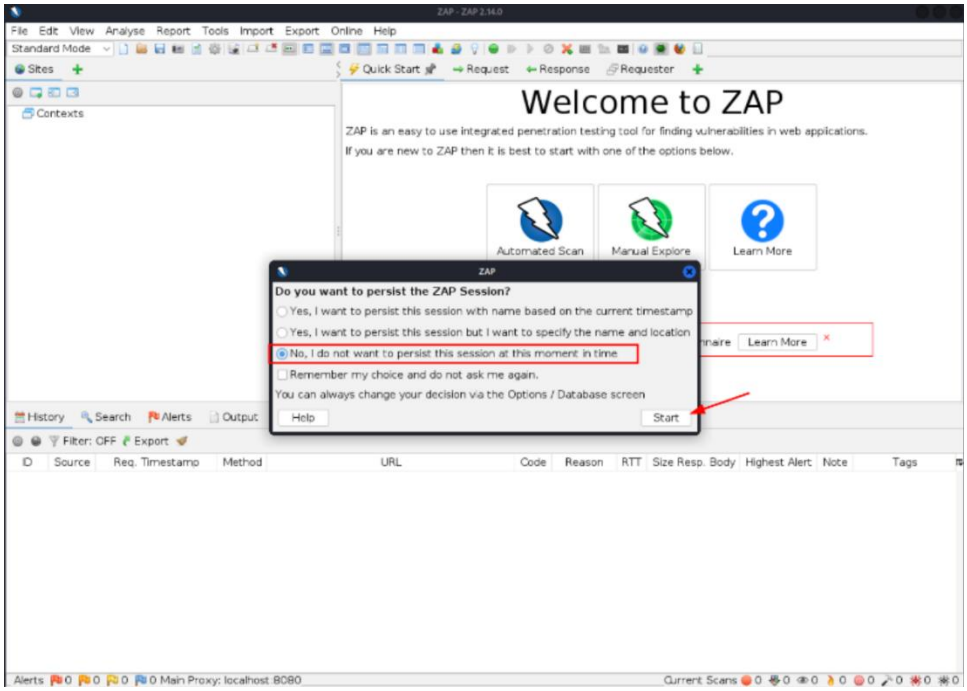


Figure 142 – OWASP ZAP Dashboard

5. Open OWASP ZAP and set up the local proxy settings.
6. Configure your web browser to use the ZAP proxy.
   A. You will need to set your browser to use ZAP as a proxy for scanning web traffic. By default, ZAP runs on localhost Address with Port 8080, changeable in **Options** | **Network** | **Local Servers/Proxies**. Here are browser-specific instructions for proxy configuration for Chrome (Windows):
      i. Click the icon in top right and select **Options**
      ii. Go to **Change proxy settings**
      iii. In LAN Settings, enable proxy server and enter ZAP's Address and Port
   B. Here are browser-specific instructions for proxy configuration for Firefox (Windows):

i. Go to **Tools** | **Options** | **General** | **Network Settings**

ii. Choose **Manual proxy configuration**

iii. Enter ZAP's Address and Port for HTTP Proxy and SSL Proxy

C. Here are browser-specific instructions for proxy configuration for Firefox (Linux):

i. Go to **Edit** | **Preferences**:

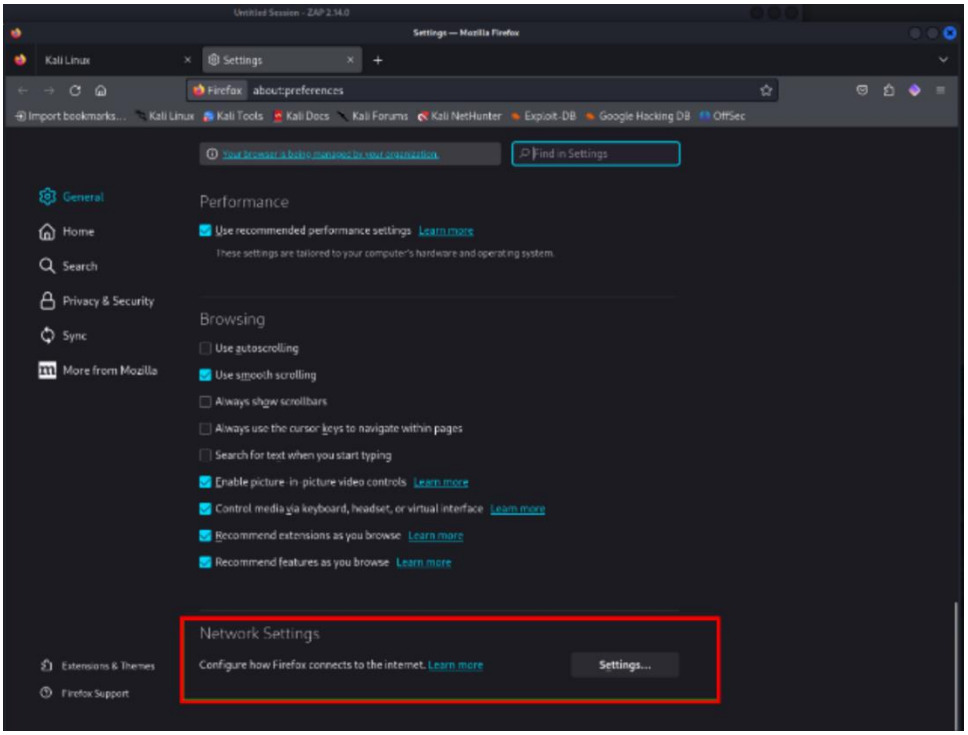


Figure 143 – Browser Network Settings

ii. Follow Windows instructions from *General* section:

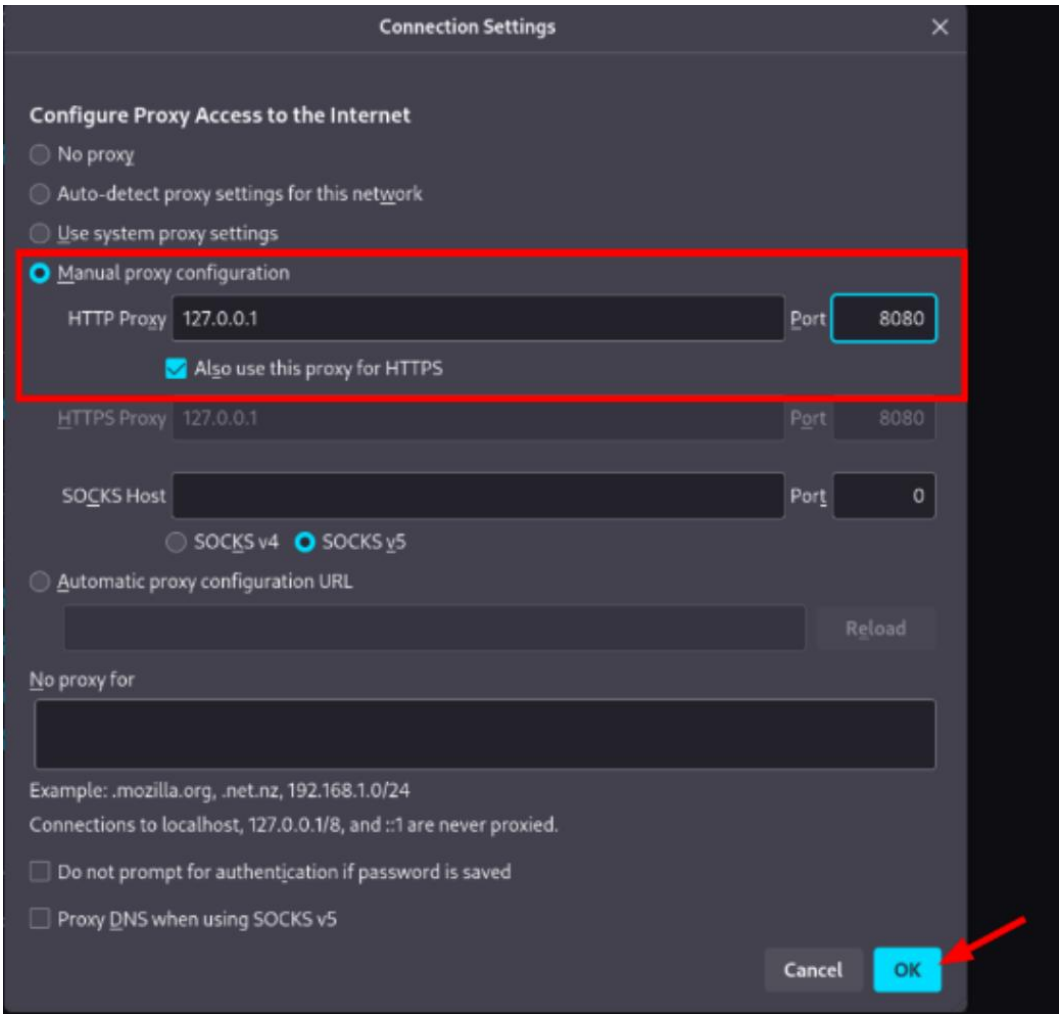


Figure 144 – Browser Proxy Settings

D. Here are browser-specific instructions for proxy configuration for Firefox (OS X):

   i. Go to **Firefox** | **Preferences**

   ii. Follow Windows instructions from *General* section

E. Here are browser-specific instructions for proxy configuration for Safari (OS X):

i. Click Safari settings icon in top right

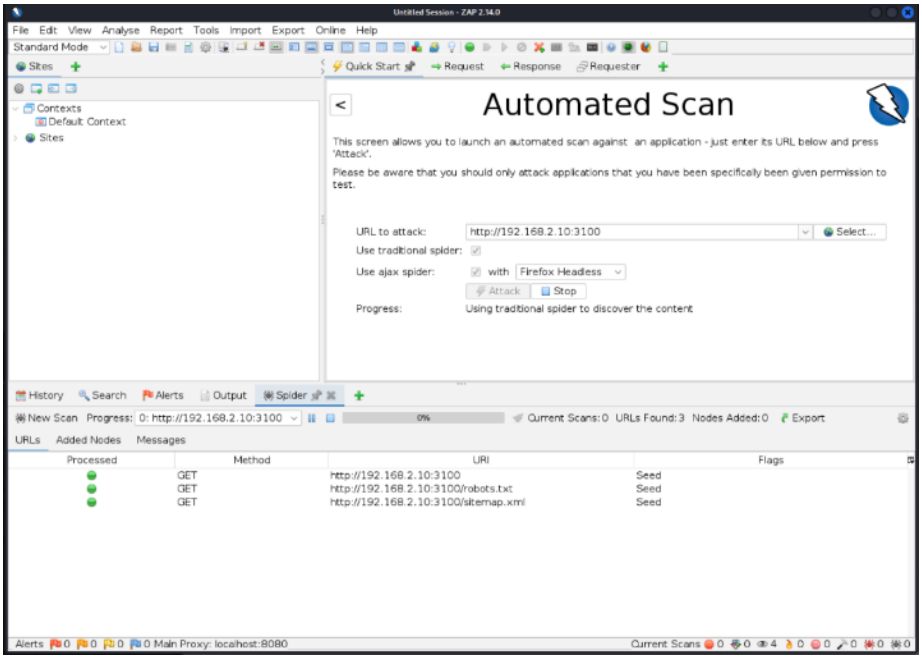
ii. Go to **Preferences** | **Advanced** | **Proxies**

iii. Click **Change Settings** and configure as system proxy

With browsers routing traffic through ZAP, you can now scan web applications.

7. Navigate through the application in your browser while OWASP ZAP passively analyzes the traffic.

8. Use the **Attack** feature in ZAP to actively scan the application for vulnerabilities:



(a) Automated Scanning

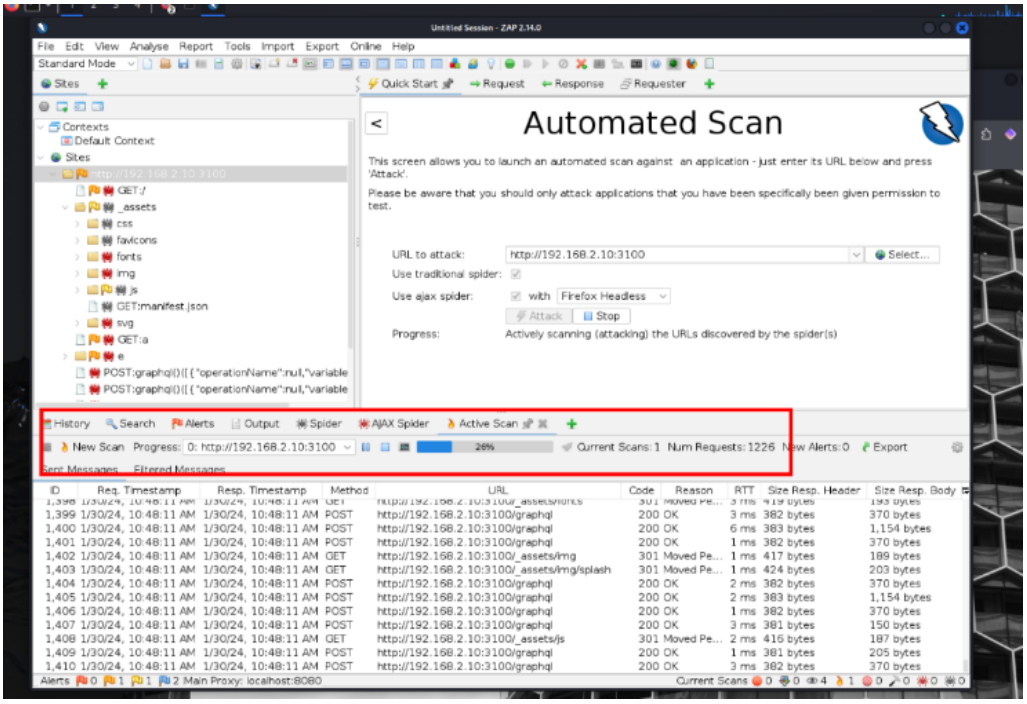


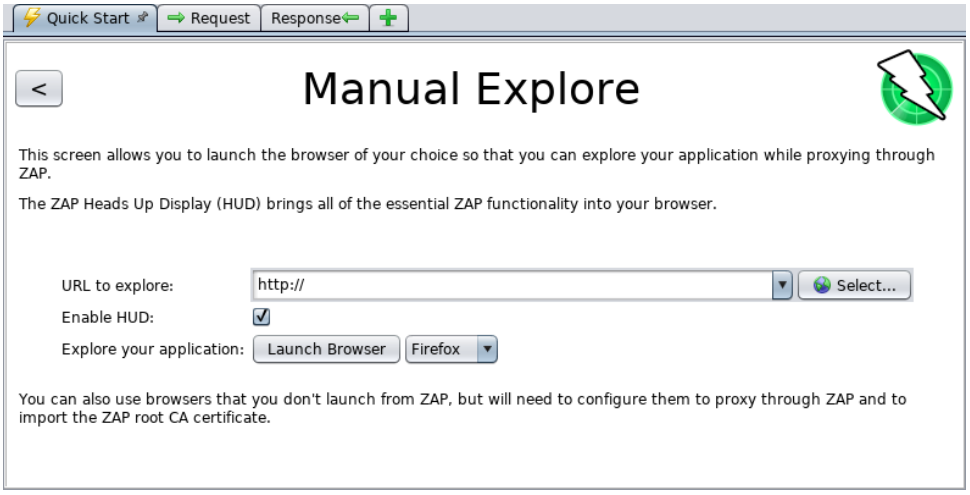
Figure 145 – Scan Progress

9. Monitor the results and identify any security issues detected.

10. Review the vulnerabilities found by OWASP ZAP, which can include issues like SQL injection, **cross-site scripting** (**XSS**), and more:

Figure 146 – Content Security Policy (CSP) Header Information

11. Analyze the risk and potential impact of each finding.
12. Explore manual testing features such as forced browsing, manual request editing, and breakpoint debugging to further investigate potential vulnerabilities. While passive scanning and automated attacks provide an initial vulnerability assessment, they have some key limitations:
13. Login-protected pages are not discoverable without configuring authentication in ZAP. There is little control over passive scan exploration or the sequence/types of automated attacks. ZAP does offer more advanced options beyond these basics. Spiders enter limited test data which may not expose complex forms. Manual exploration with real inputs reveals more of the application. Obscure pages sometimes go live without notice, so exhaustive manual site exploration is important even if pages don't link elsewhere. Obscurity is not security.
14. To thoroughly explore the application:
    - Launch a browser configured to proxy through ZAP via the Quick Start tab. This automatically handles certificates.
    - Or manually configure your browser to proxy via ZAP and import/trust the ZAP root CA certificate for sites with errors.
    - Be sure to manually visit every page, submit realistic form data, and execute all site functionality. ZAP will passively scan traffic for weaknesses.
15. Combining automated scanning with comprehensive manual exploration and real-world testing generates more accurate findings by fully exercising complex application logic and surfaces obscured pages:



(a) Manual Scan

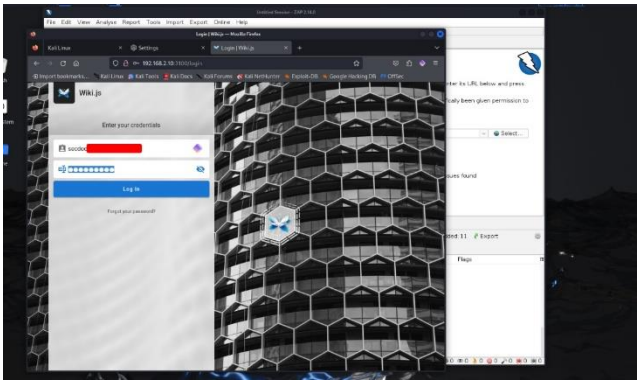


Figure 147 – Scan Target Site

16. Address the vulnerabilities in the application's code or configuration.

17. Retest the application to ensure that the remediations are effective.

18. Generate a report within OWASP ZAP detailing the vulnerabilities found and the steps taken to remediate them:

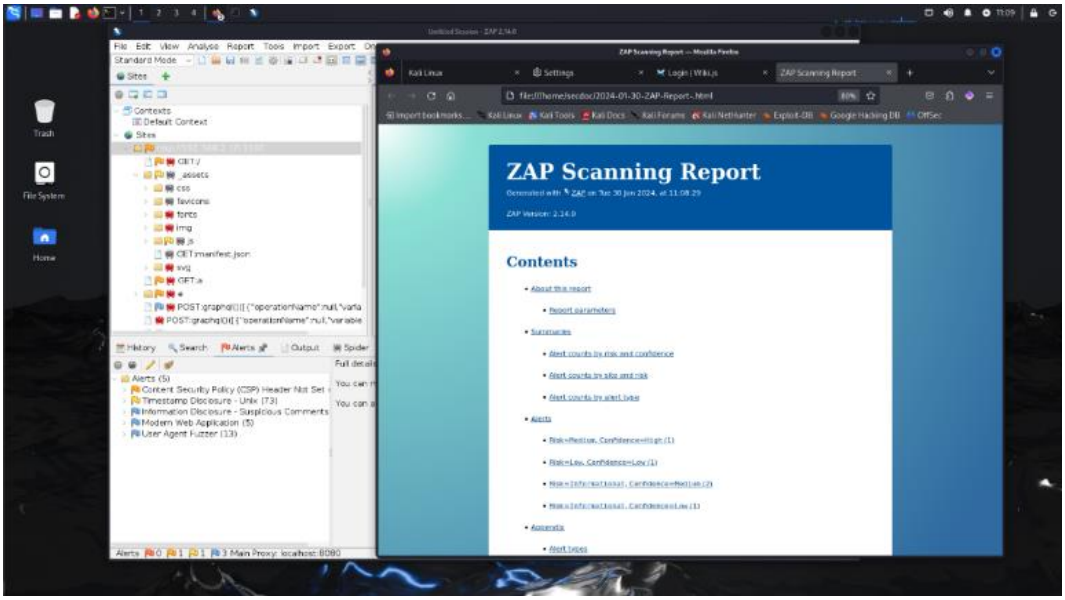


Figure 148 – San Report

19. Document the DAST process, findings, remediation actions, and any additional manual testing steps taken.

Through these labs, participants will gain practical skills in both static and dynamic analysis, covering a comprehensive approach to application security testing. Proper documentation and regular analysis are crucial for maintaining application security over time.

Layered application testing provides code level through runtime assurance of software security throughout the development lifecycle. Architects combine results to prioritize remediation efforts.

Cloud security tools

Lab exercises for cloud security tools often involve services provided by cloud service providers such as AWS, Azure, or GCP. For this example, we will focus on using AWS native tools to ensure security within the AWS cloud environment. The lab will cover AWS **Identity and Access Management** (**IAM**), AWS Security Groups, AWS Inspector, and AWS CloudTrail.

Lab: Setting up and securing an AWS environment

AWS (**Amazon Web Services**) is a comprehensive cloud computing platform provided by Amazon, offering a wide range of services and tools for building, deploying, and managing applications and infrastructure in the cloud. With AWS, organizations can leverage a global network of data centers and a robust set of cloud services to scale their applications, store and process data, and deliver content to users worldwide. AWS provides a flexible and cost-effective environment that allows businesses to focus on their core competencies while offloading the complexity of managing underlying infrastructure. From compute power and storage to databases, networking, and security, AWS offers a vast array of services that cater to different application requirements and workload patterns. With its pay-as-you-go pricing model, extensive documentation, and strong community support, AWS has become a popular choice for startups, enterprises, and government agencies looking to embrace the benefits of cloud computing and drive innovation in their respective domains. Setting up and securing an AWS environment is crucial to ensure the integrity, confidentiality, and availability of your applications and data. AWS provides a shared responsibility model, where AWS is responsible for the security of the cloud infrastructure, while customers are responsible for securing their applications and data within the cloud.

By implementing these security best practices and leveraging AWS security services, organizations can establish a robust and secure AWS environment. It's important to continuously monitor, assess, and improve the security posture of your AWS environment to stay ahead of evolving threats and maintain the confidentiality, integrity, and availability of your applications and data.

AWS offers extensive documentation, whitepapers, and resources to guide customers in setting up and securing their environments. Additionally, AWS provides professional services and support to assist customers in their cloud security journey.

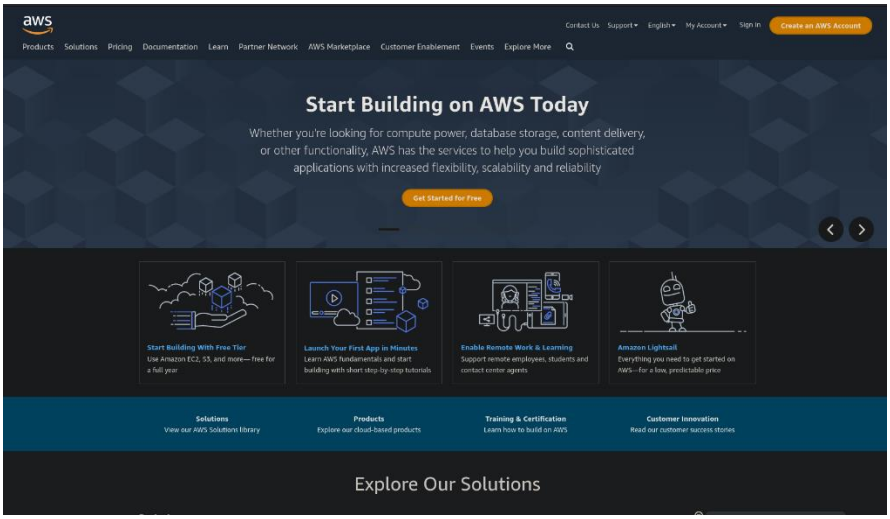


Figure 149 – AWS Website

The prerequisites include:

- An AWS account with administrative access.
- A virtual machine or local development environment to run AWS CLI and SDKs if not using AWS CloudShell.

Let us look at the steps:

1. Create or log into your AWS Management Console:

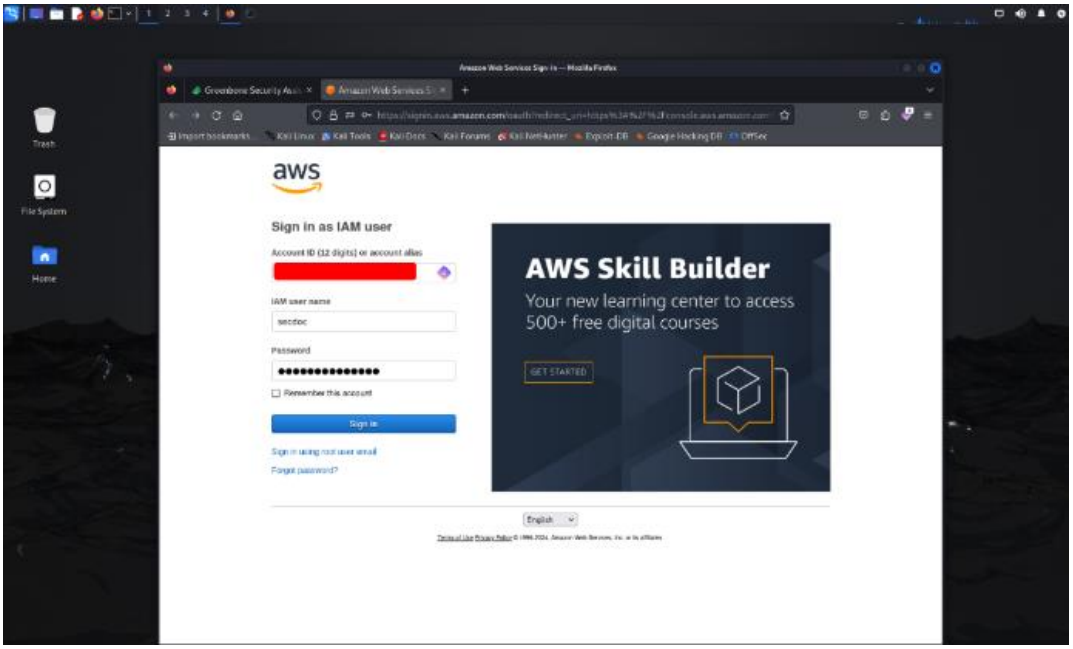
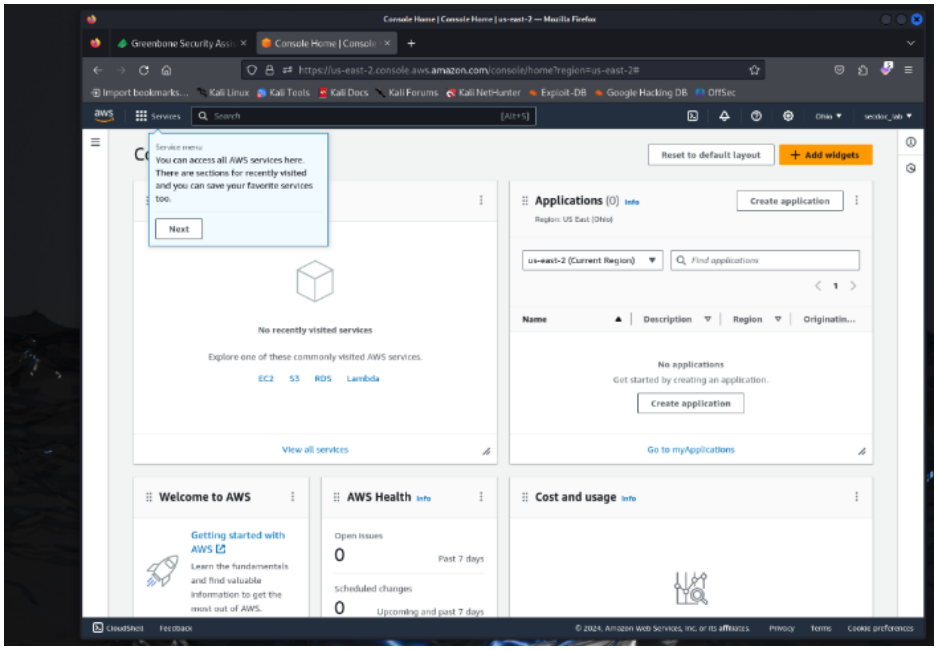


Figure 150 – AWS Login

2. Familiarize yourself with the AWS services that will be used in this lab, specifically IAM, EC2, Inspector, and CloudTrail:



(a) AWS Console

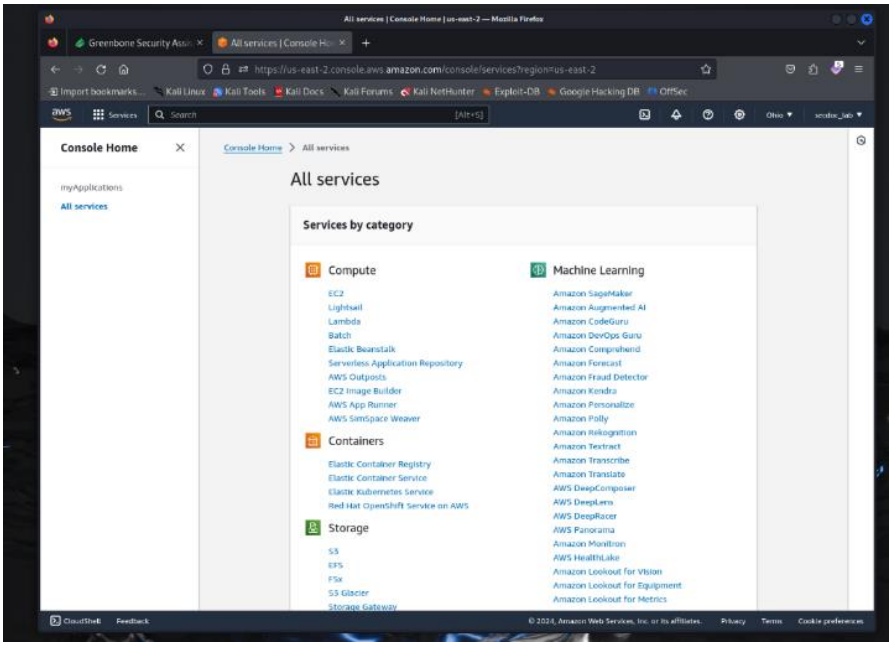


Figure 151 – AWS Service Listing

3. Create a new IAM user with programmatic access. In the left navigation pane, click **Users** and then click the **Add user** button:

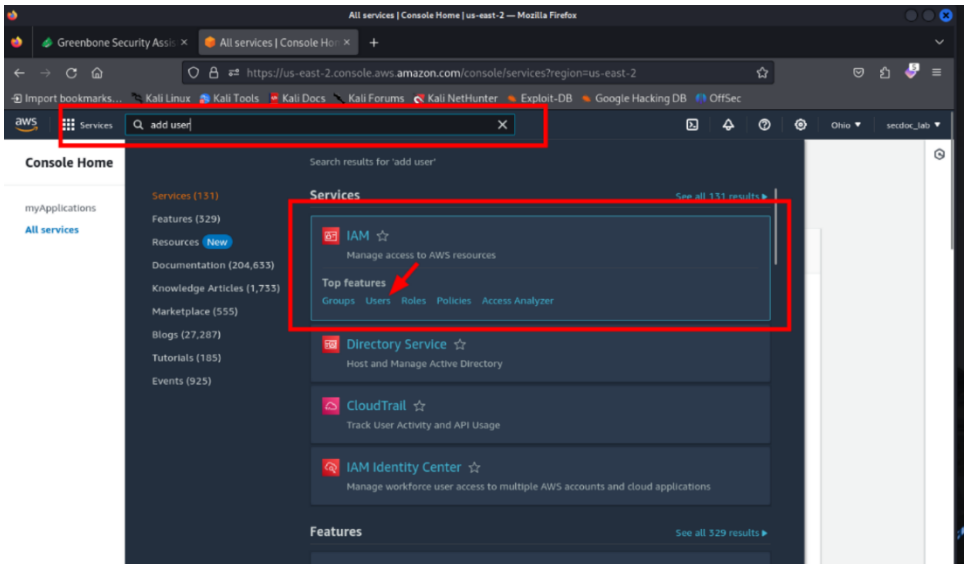


Figure 152 – Adding User

4. Enter a user name and select the type of access as **Programmatic access** for access with Access Key ID and Secret Access Key:

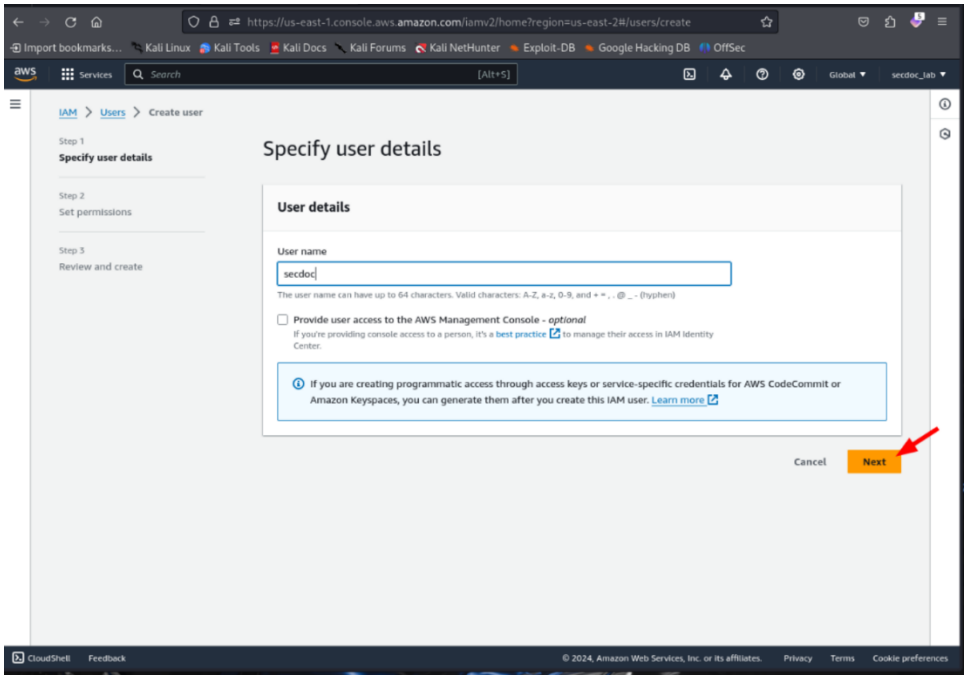


Figure 153 – User Creation dialog box

5. Click **Next: Permissions**.

6. Select **Attach existing policies directly** to assign desired managed policies to the user based on intended access level. You may need multiple policies:

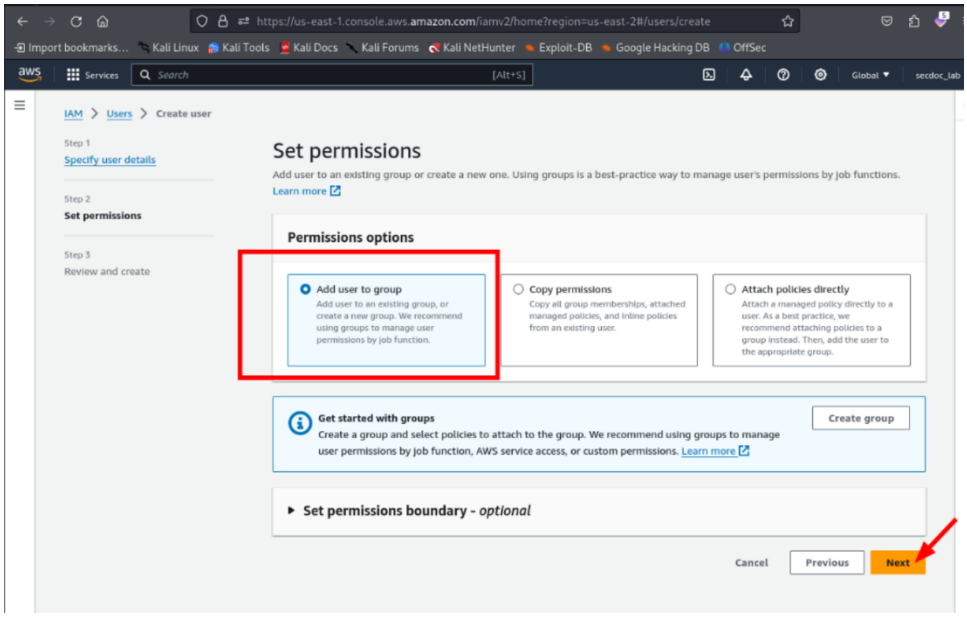


Figure 154 – Setting User Permissions

7. Click **Next: Tags**. Add any resource tags if applicable.
8. Click **Next: Review** to review all user details and permissions before continuing.
9. Click **Create user** once you have verified the configuration is correct:

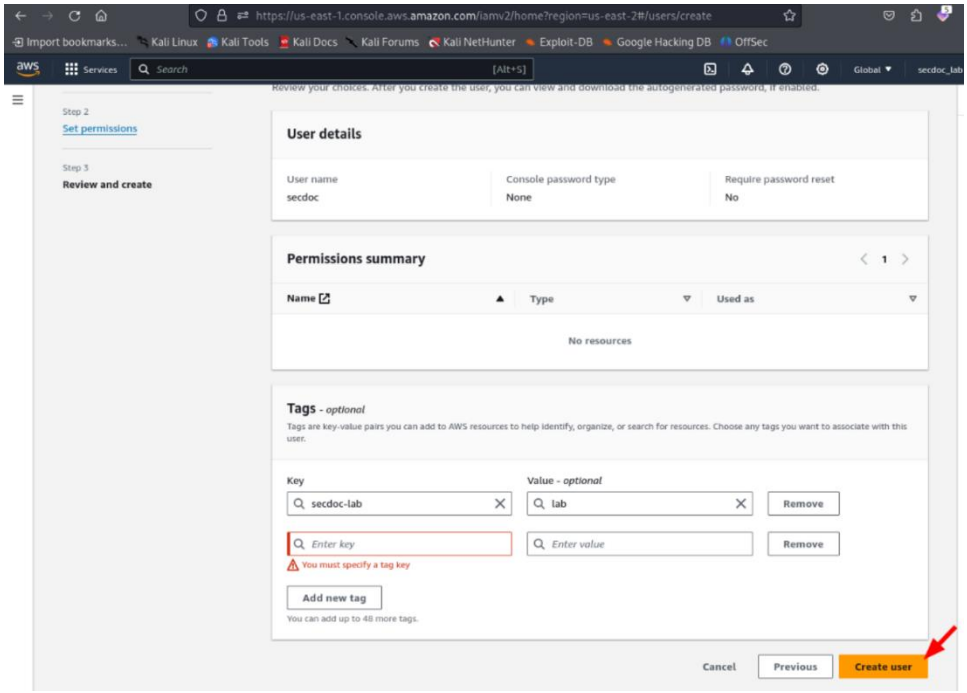


Figure 155 – Create User

10. The user Access key ID and Secret access key will display only once, be sure to copy or download the keys before continuing as they cannot be retrieved later:

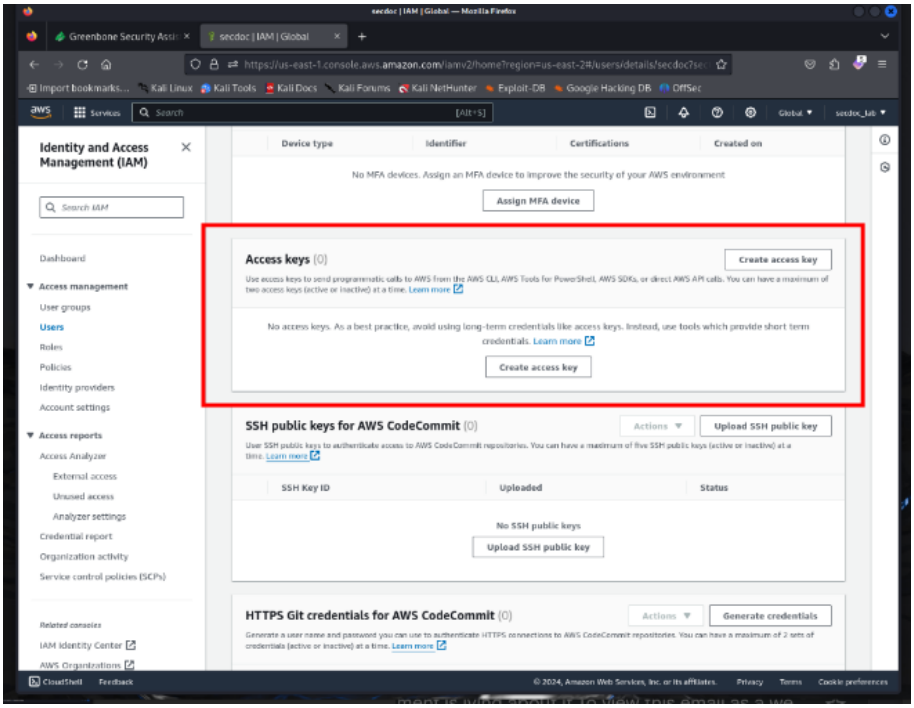


Figure 156 – Access Keys

11. Click **Close** once you have the keys saved externally.

12. The new user is now created and can access AWS programmatically using the Access Key ID and Secret Access Key.

13. Assign the IAM user to groups with policies granting the necessary permissions for EC2, Inspector, and CloudTrail. In the left navigation pane, select **Groups**:

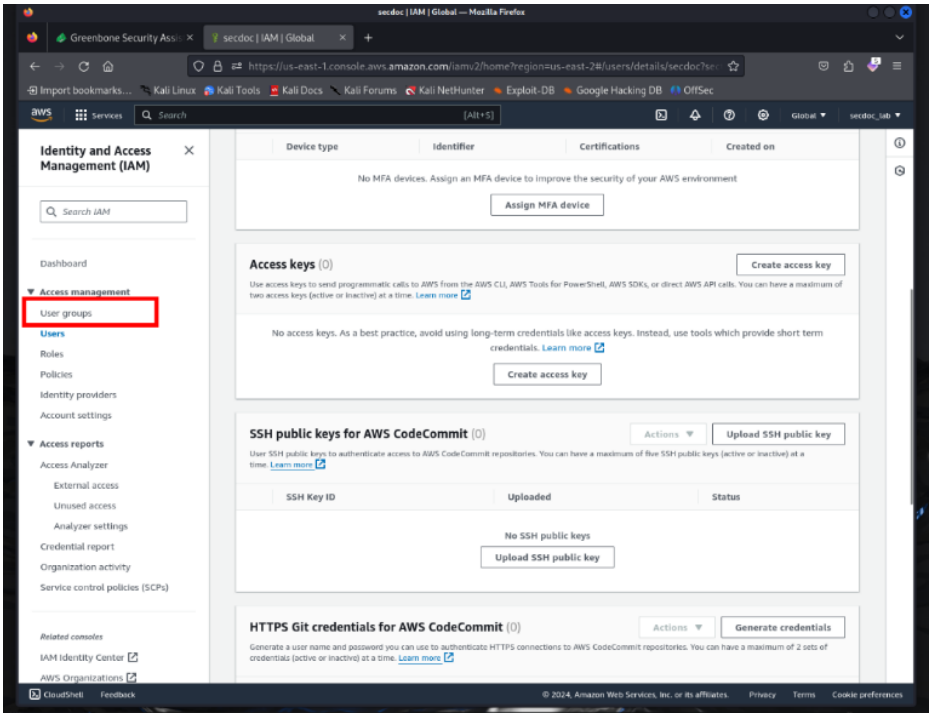


Figure 157 – User Groups

14. Click the **Create new group** button:

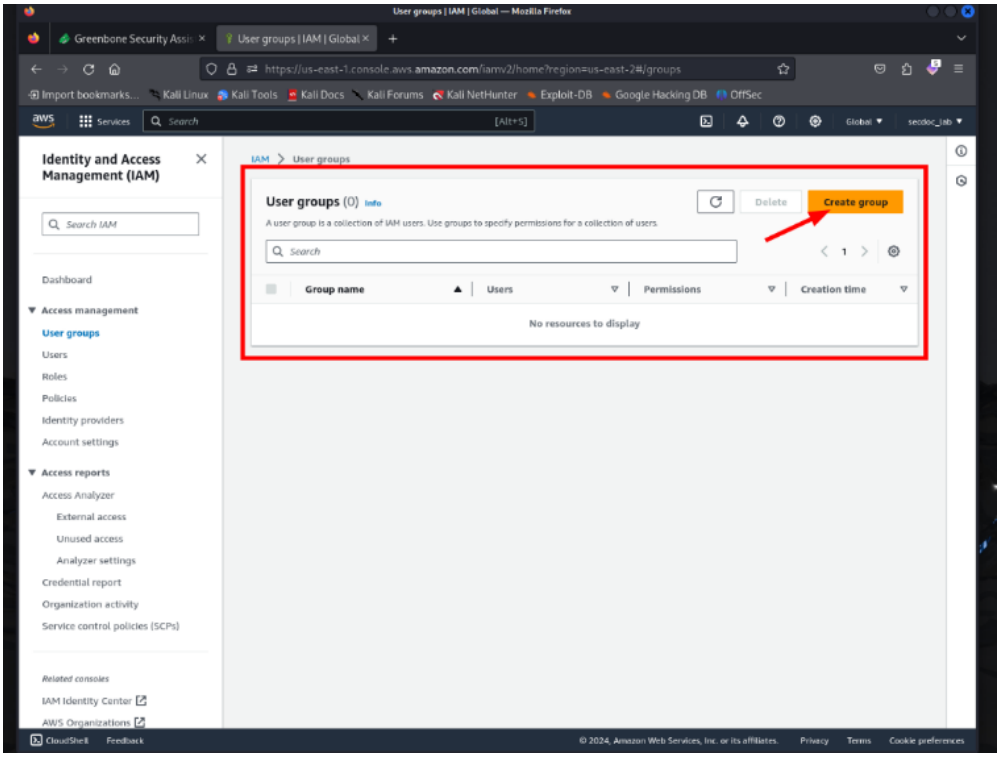


Figure 158 – Create User Group

15. Give the group a name like **EC2-Inspector-Access** and click **Create Group**:

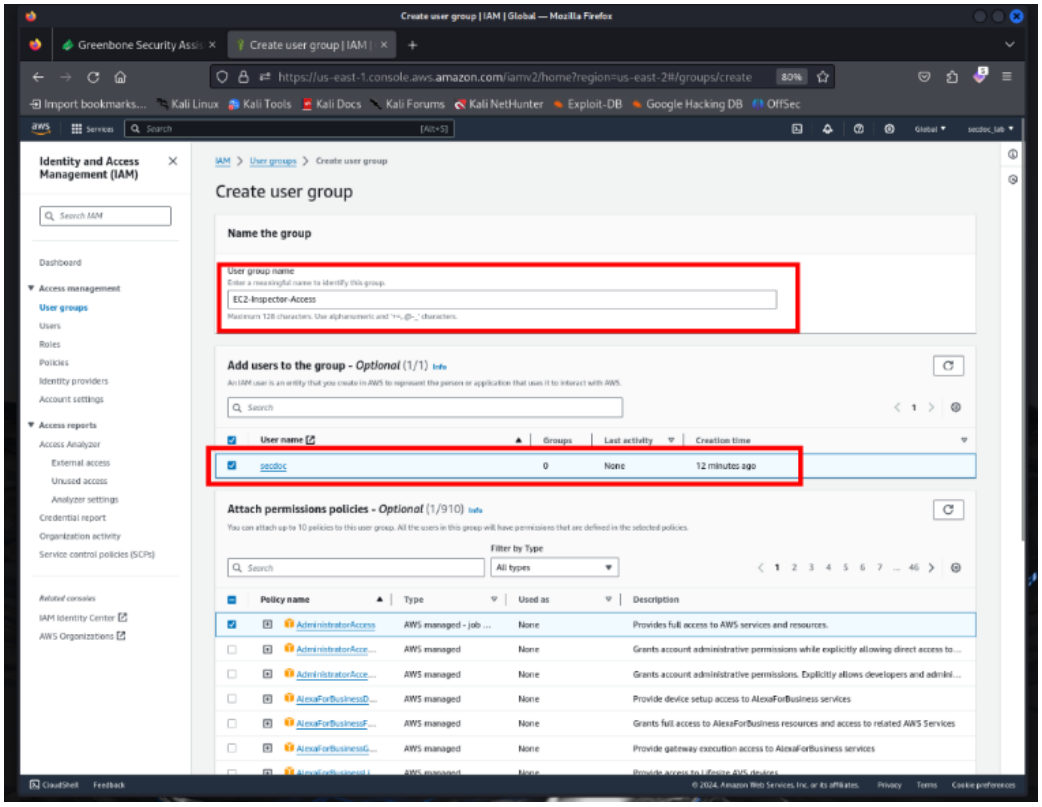


Figure 159 – Access Group Name

16. Select the newly created group in the list:

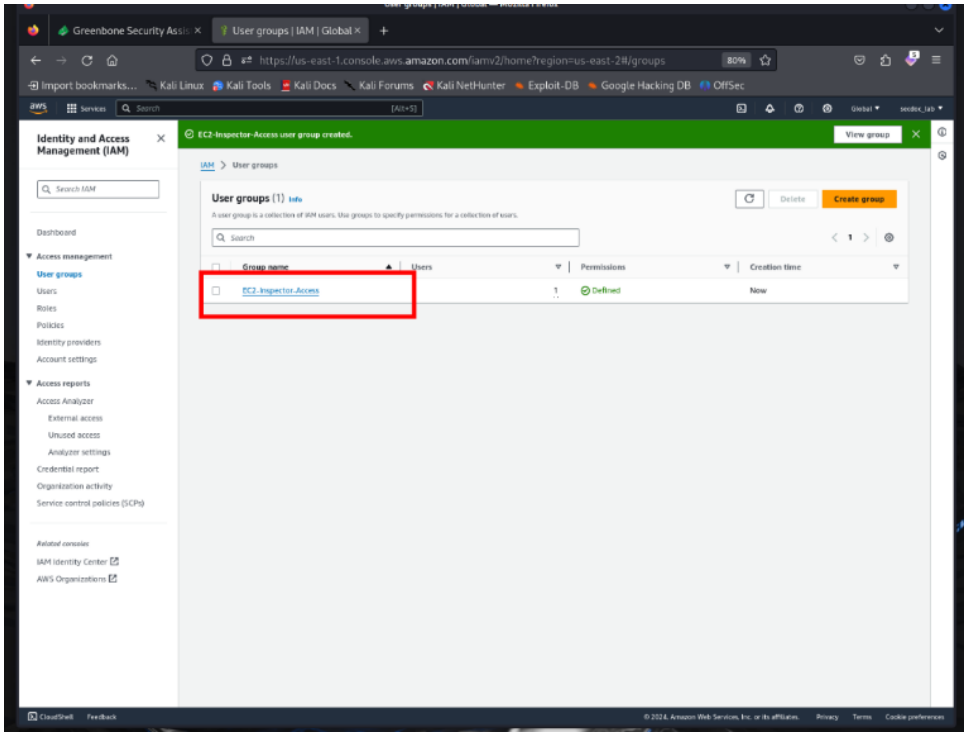


Figure 160 – Select Group

17. On the **Permissions** tab, click **Attach Policy**:

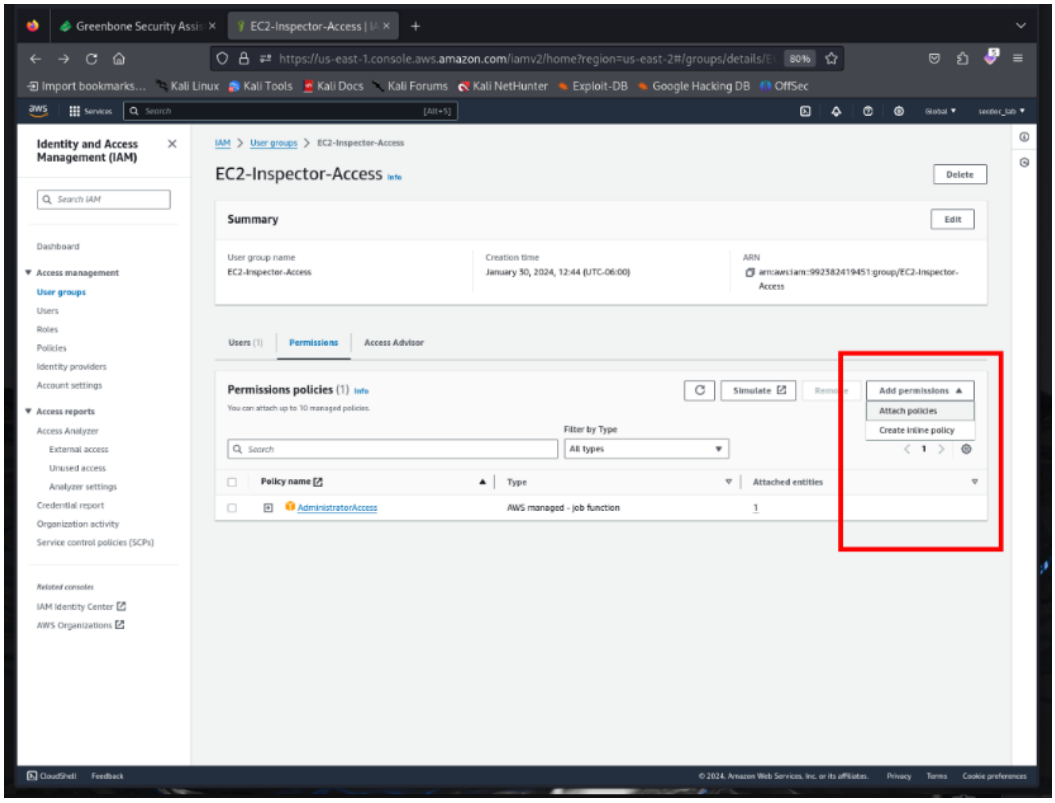


Figure 161 – Attaching Policy

18. Search for and select the following managed policies:

    A. AmazonEC2FullAccess:

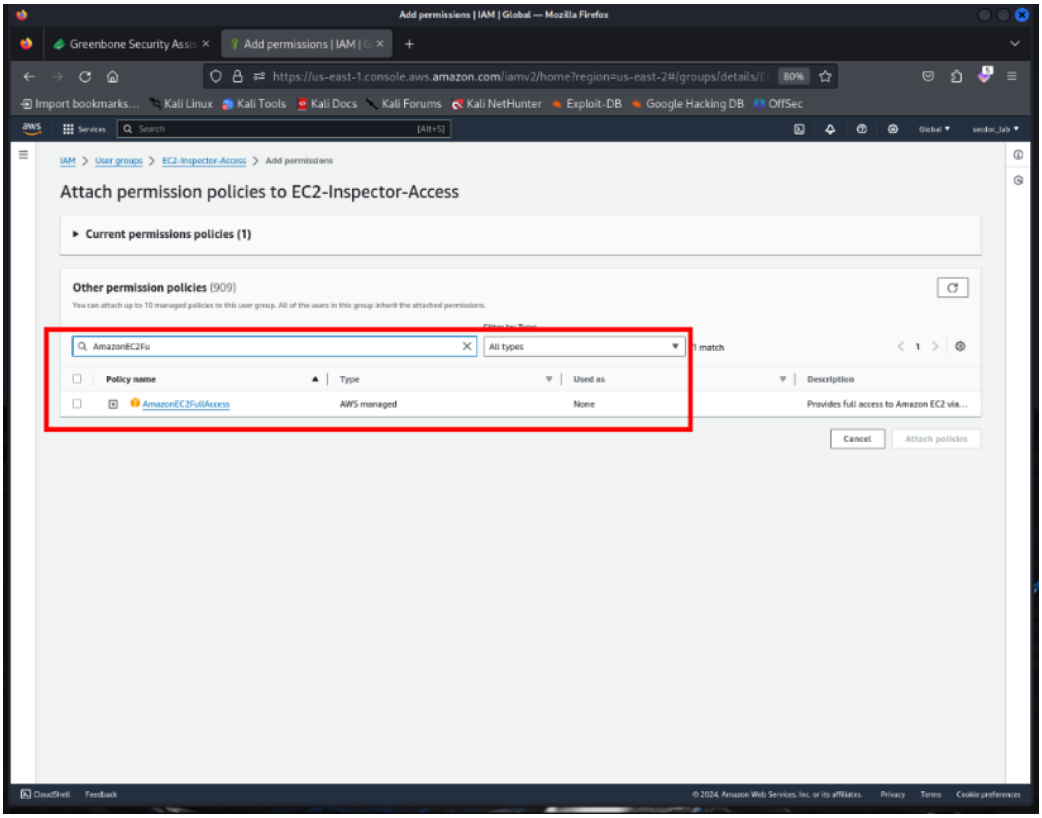


Figure 162 – EC2 Inspector Policy

B. AmazonInspectorFullAccess

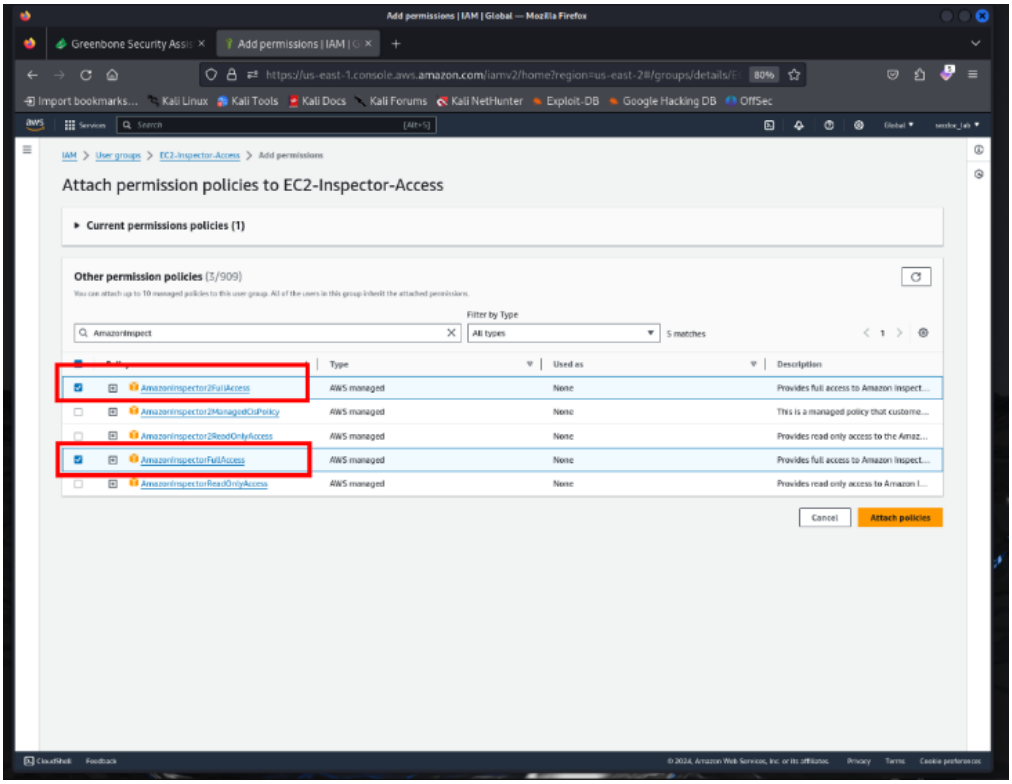


Figure 163 – Permissions

C. AWSCloudTrailFullAccess

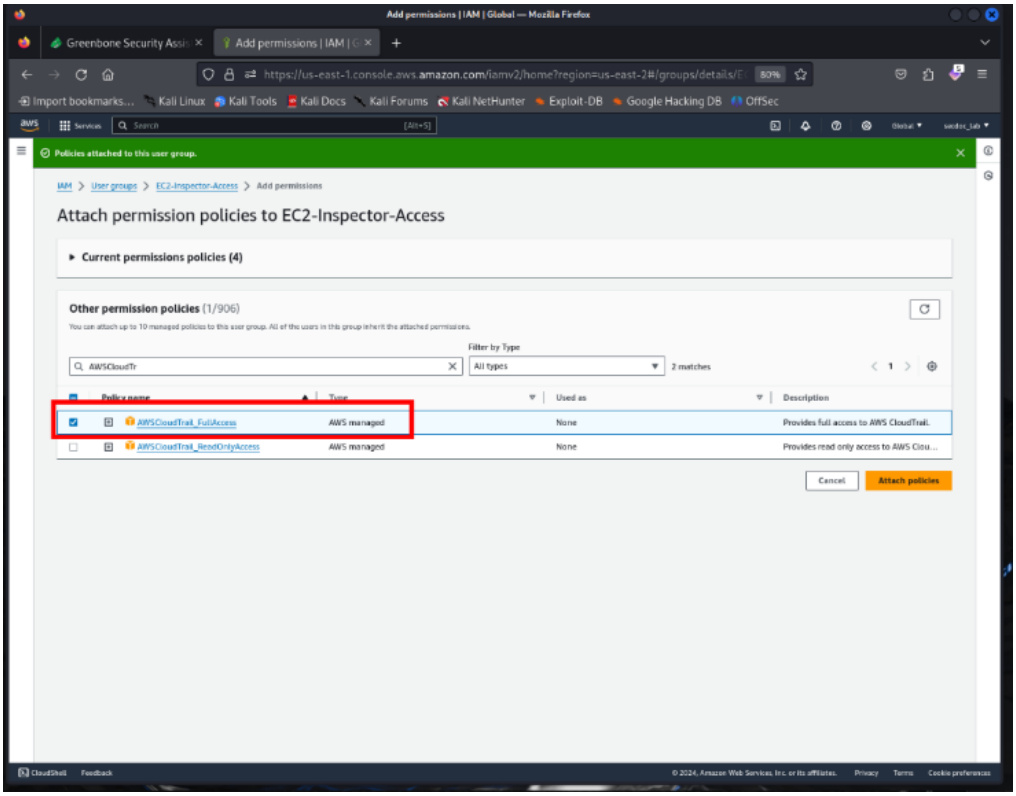


Figure 164 – CloudTrail Policy

19. Click **Attach policy** to attach all three policies to the group:

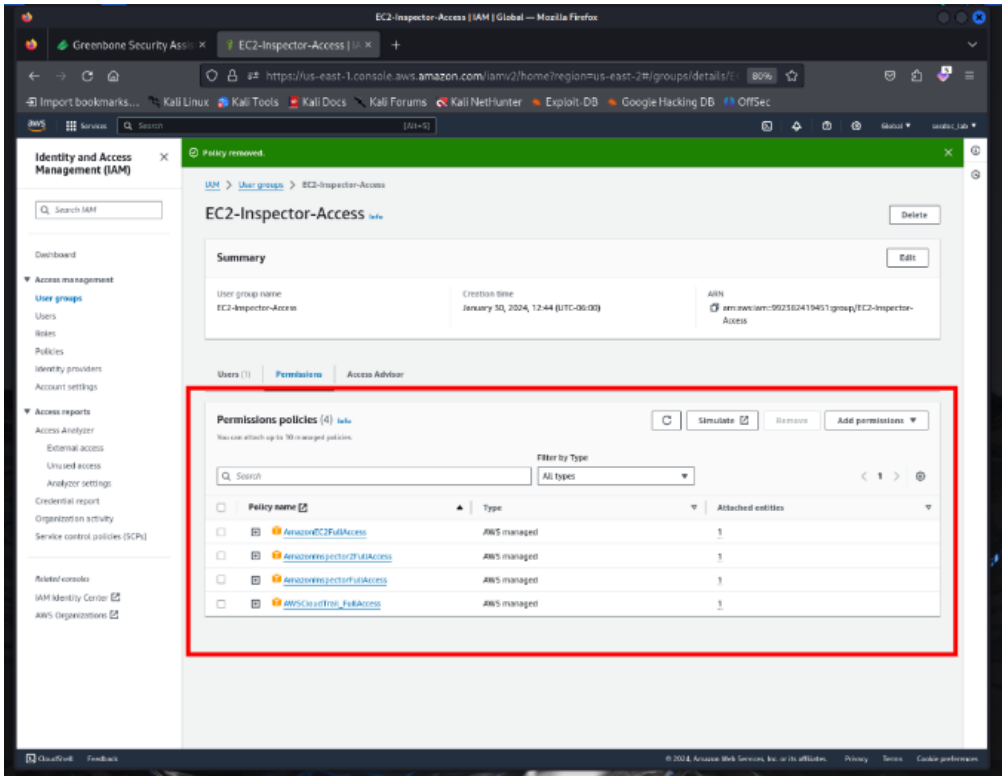


Figure 165 – Permission Policies

20. In the left navigation pane, select **Users**:

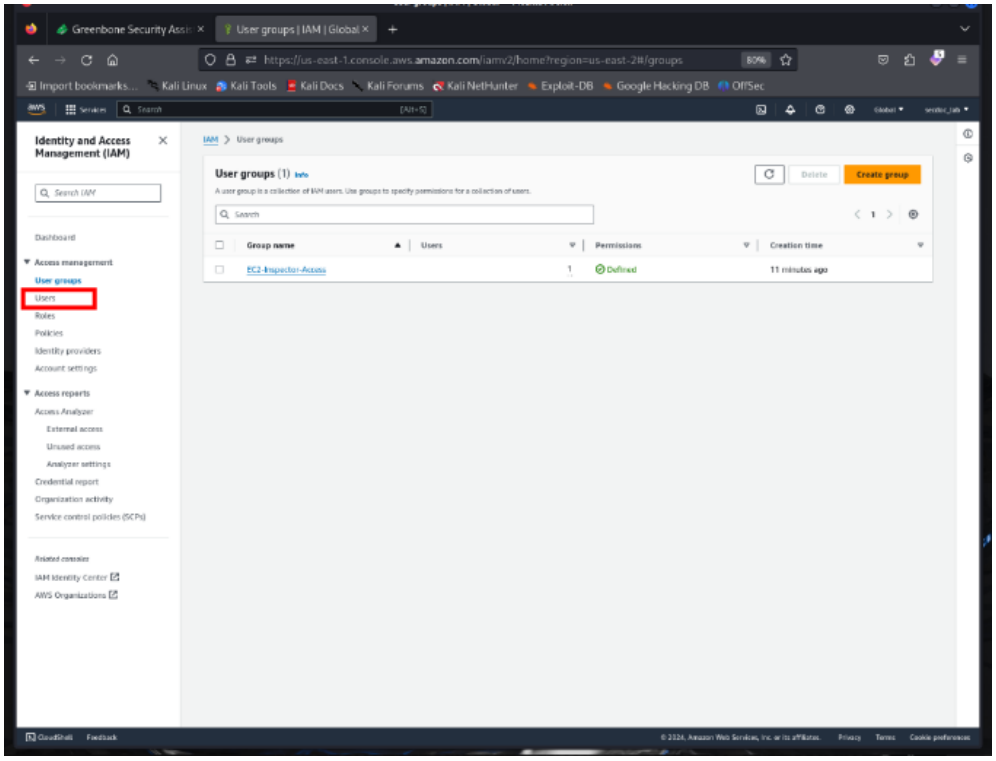


Figure 166 – Selecting Users

21. Select the target user you want to grant access permissions to:

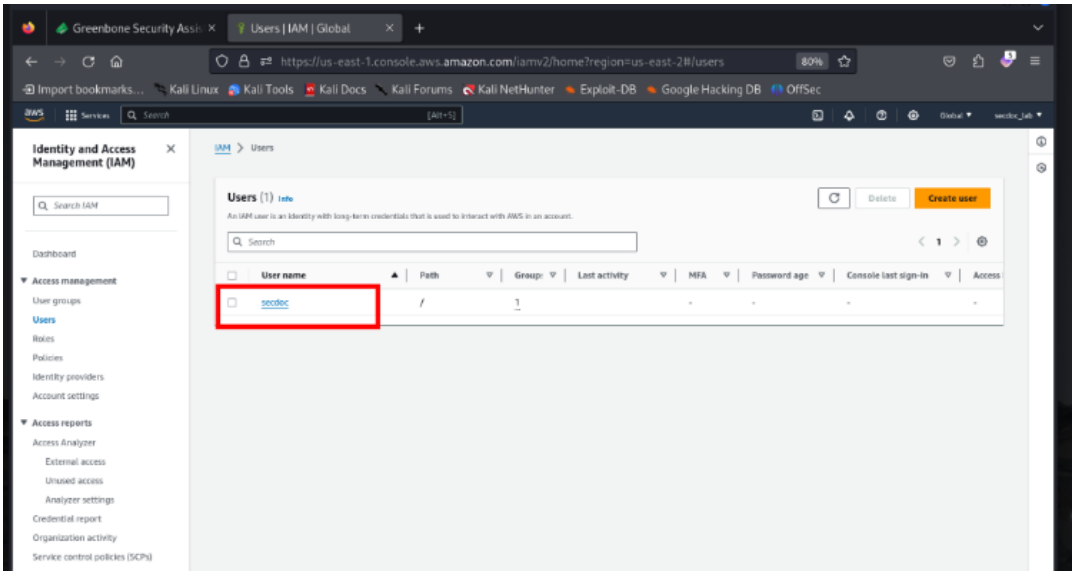


Figure 167 – User for Permissions

22. On the **Permissions** tab, click **Add user to groups**:

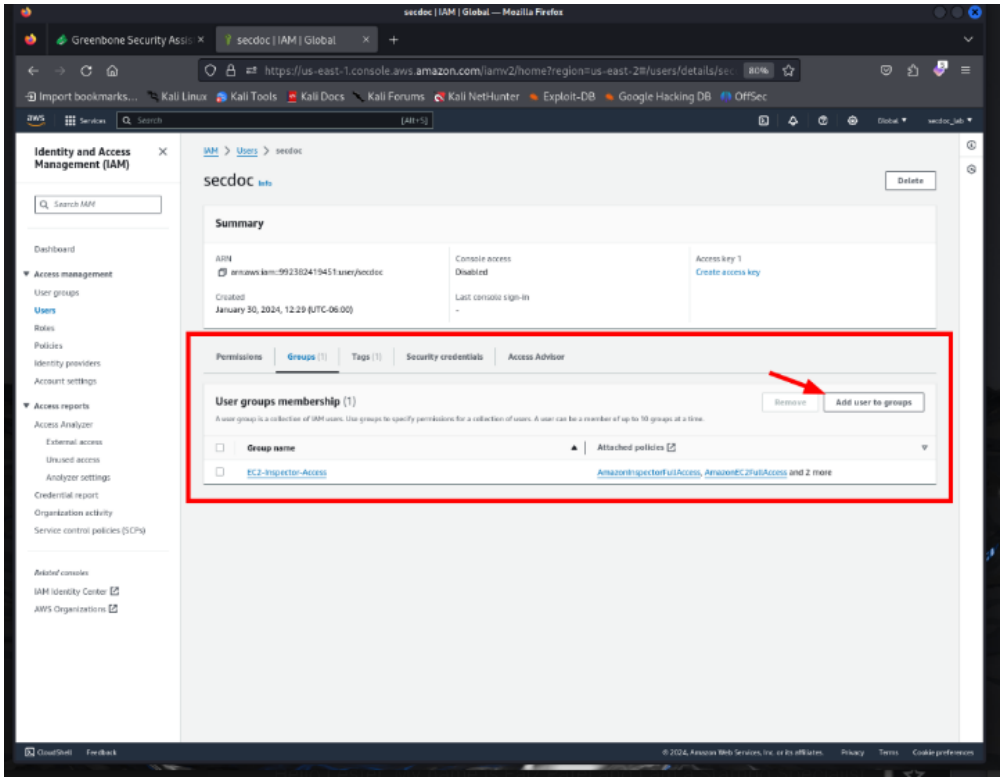


Figure 168 – Adding User to Group

23. Select the **EC2-Inspector-Access** group and click **Add to Groups**.

24. The user is now an IAM group member with the necessary EC2, Inspector, and CloudTrail permissions to perform actions allowed by those service-specific policies.

25. Create an IAM role with security auditing permissions and assign it to an EC2 instance for Inspector scanning. Launch an EC2 instance that will be the target for security assessment:

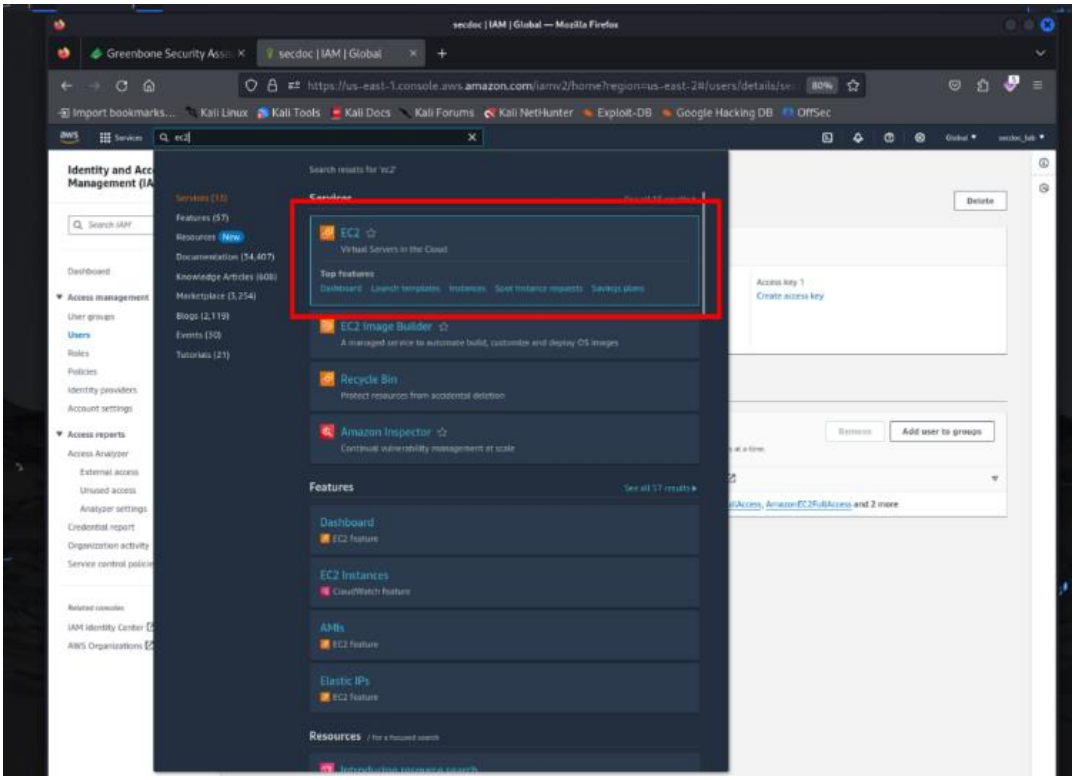


Figure 169 – Selecting EC2

26. Now, let's look at an EC2 instance setup. Launch an EC2 instance that will be the target for security assessment:

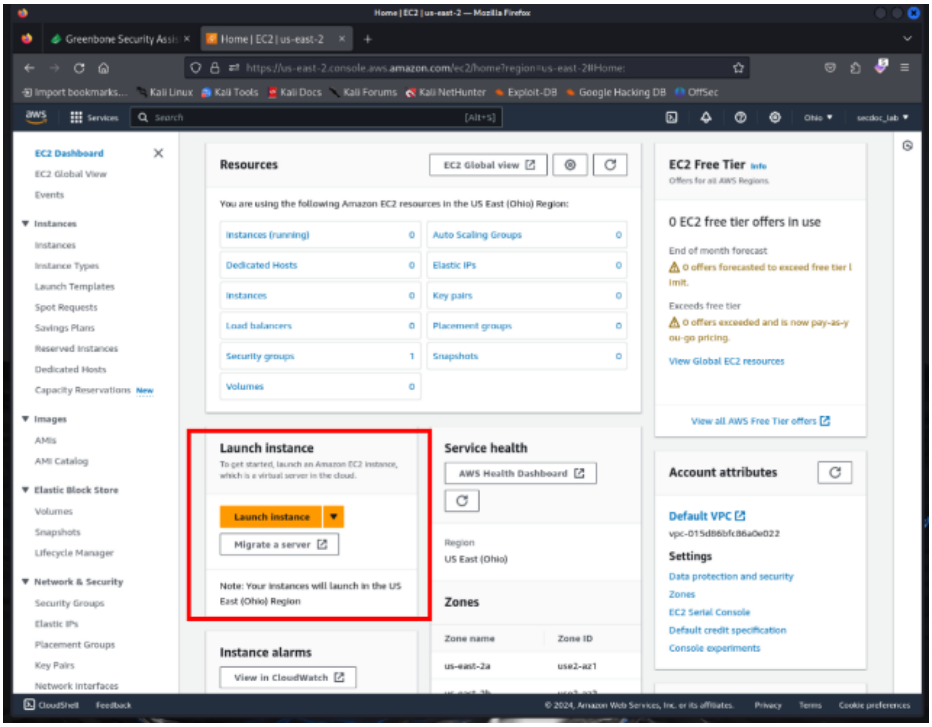


Figure 170 – Launching EC2 Instance

27. Ensure that the security group associated with the EC2 instance allows traffic only on the necessary ports (e.g., `80` for HTTP, `443` for HTTPS).

28. In the EC2 console, click **Instances** and select the newly launched instance:

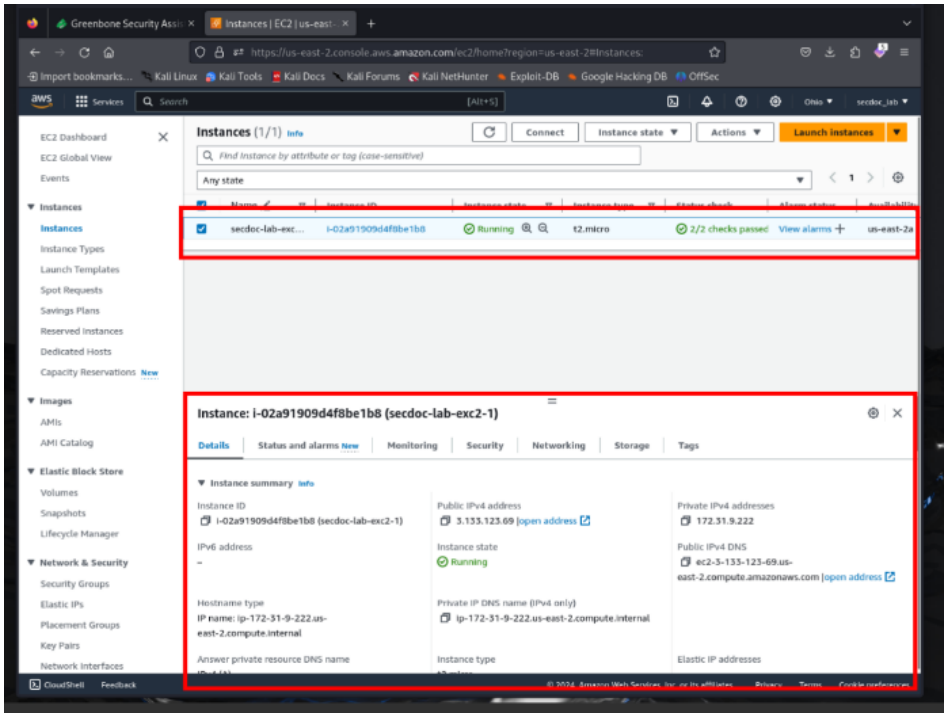


Figure 171 – EC2 Instance Details

29. Under the **Details** tab, copy or note down the instance ID:

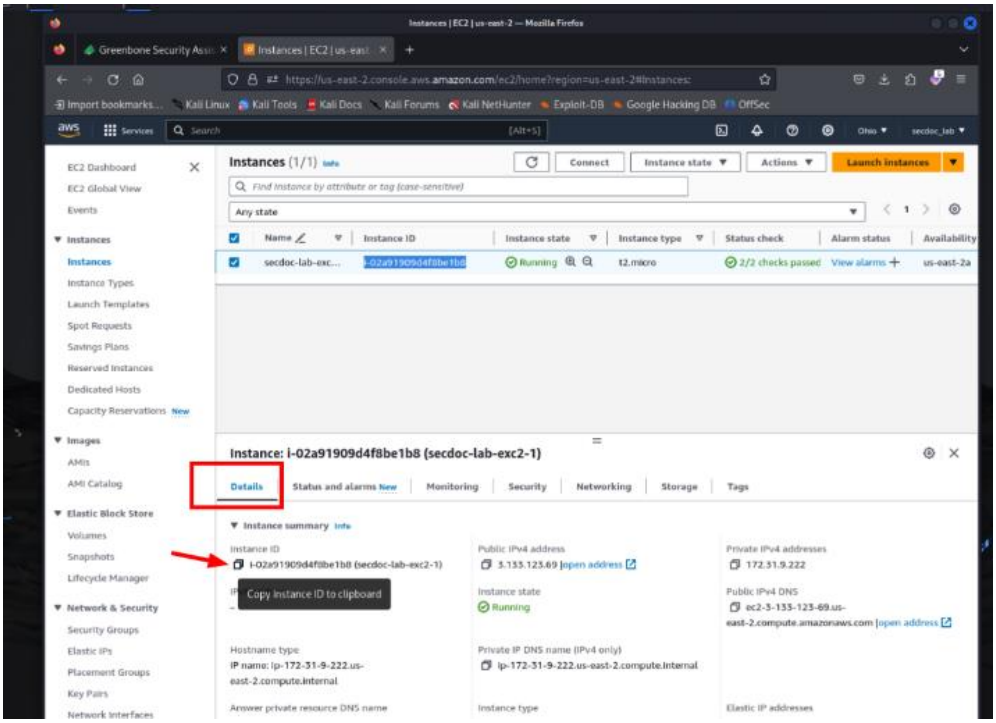


Figure 172 – EC2 Instance ID

30. Open the IAM service in the AWS console:

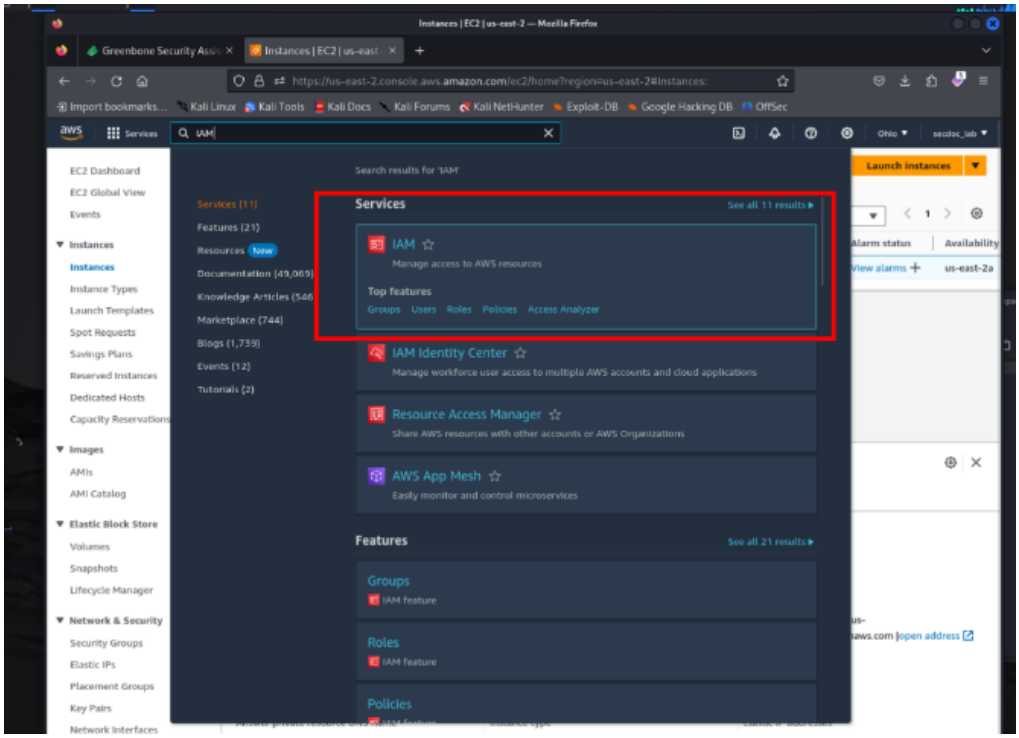


Figure 173 – IAM Console

31. Click **Roles** and then **Create role**:

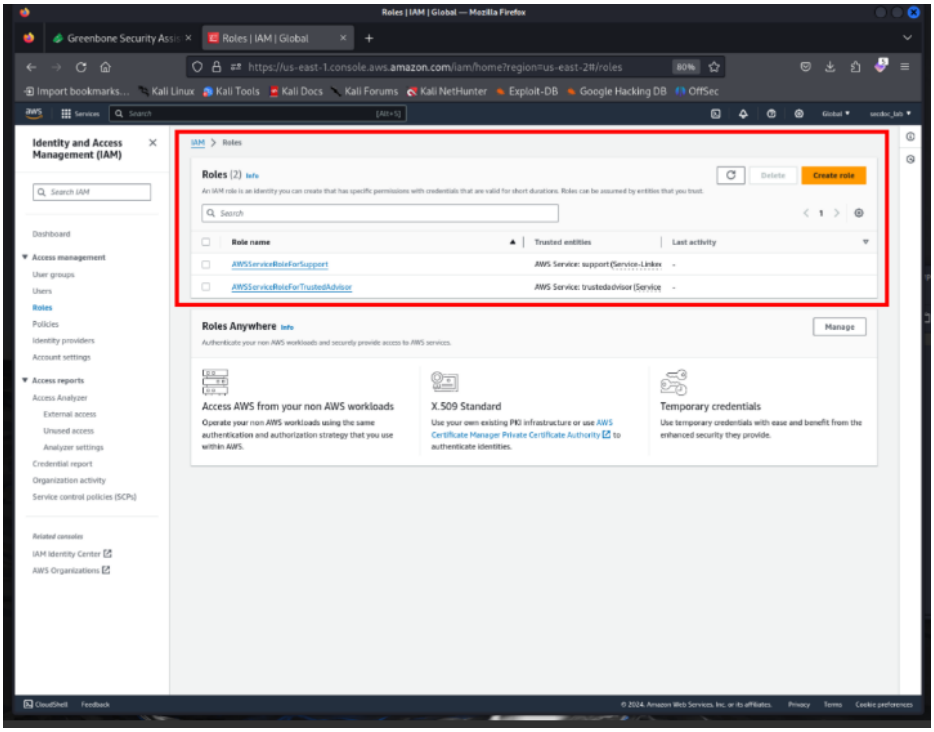


Figure 174 – IAM Roles

32. Under **Select type of trusted entity**, choose **EC2**:

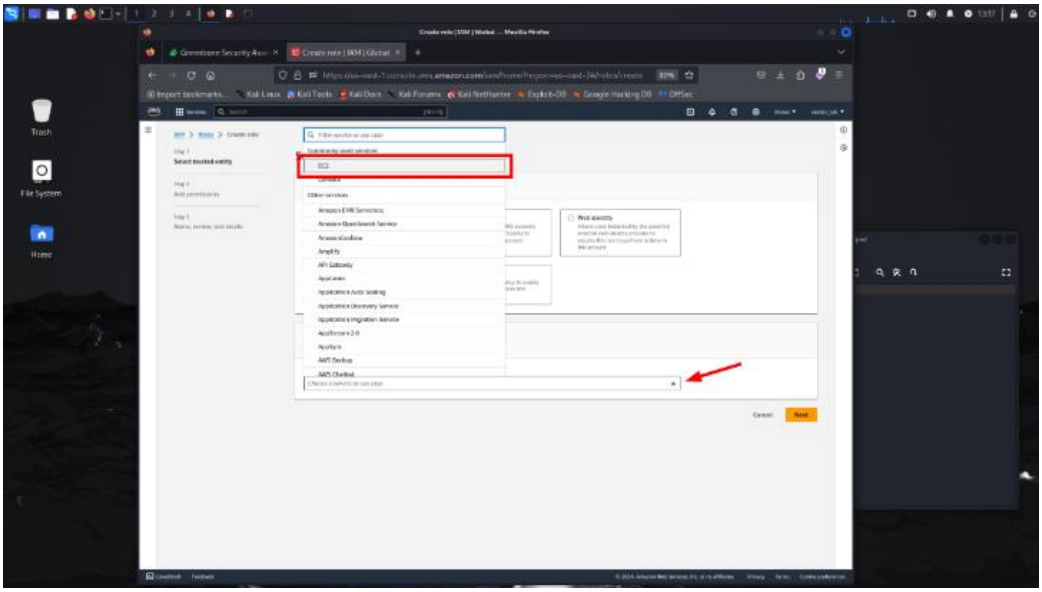


Figure 175 – EC2 Entity Selection

33. Under **Select your use case**, choose **EC2** then select **Next: Permissions**.

34. Search for and attach the AWS managed policies:

    A. AmazonInspectorFullAccess:

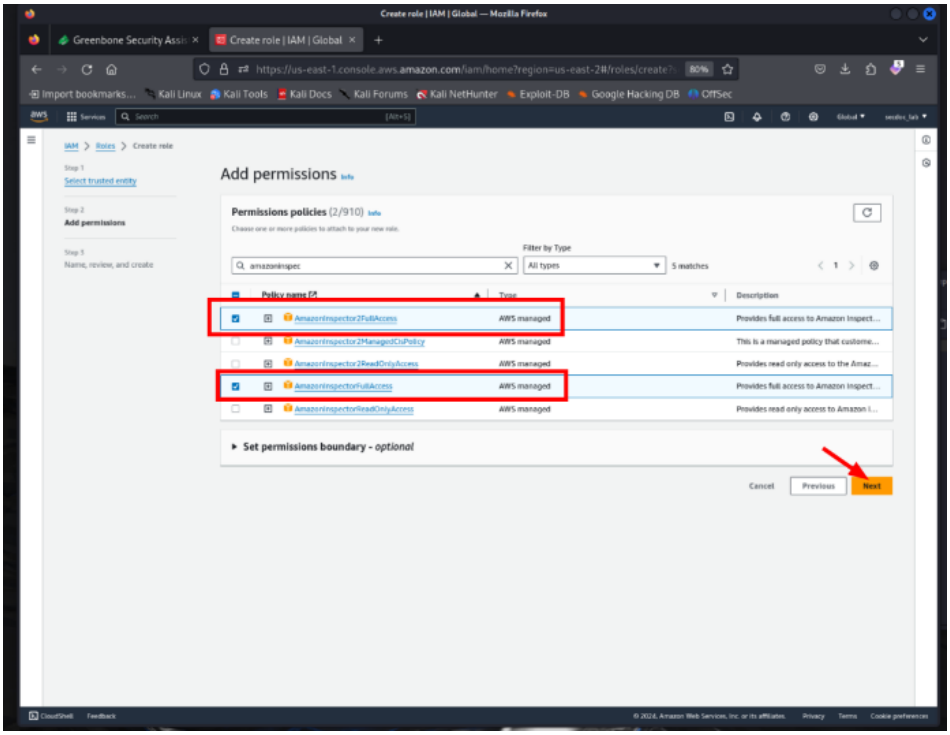


Figure 176 – IAM Permissions

B. SecurityAudit:

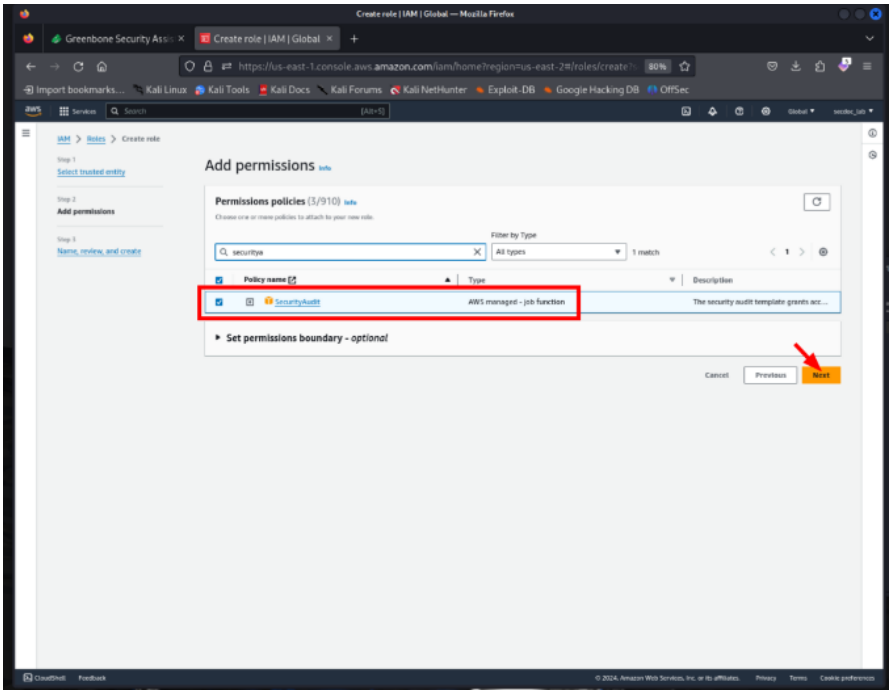


Figure 177 – Policy Selection

35. Click **Next: Tags**, optionally add tags, then click **Next: Review**:

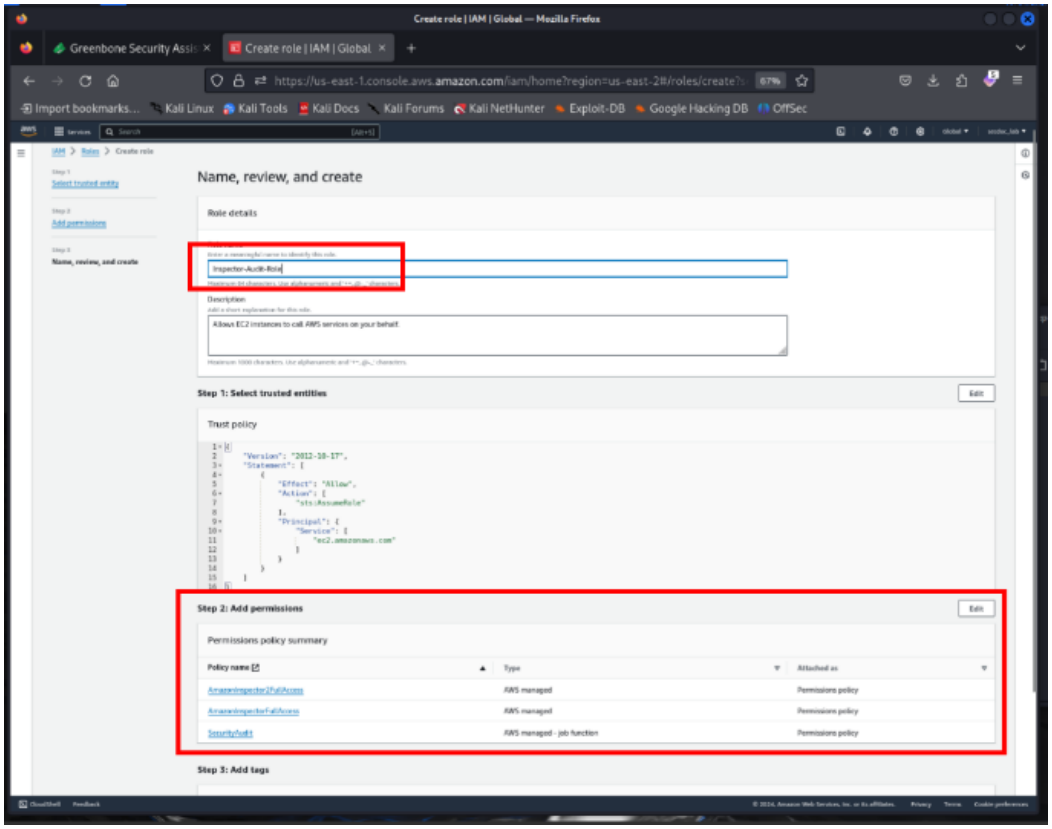


Figure 178 – Adding Permission to Role

36. Enter a role name like **Inspector-Audit-Role** and create the role.

37. Click on the newly created IAM role in the list:

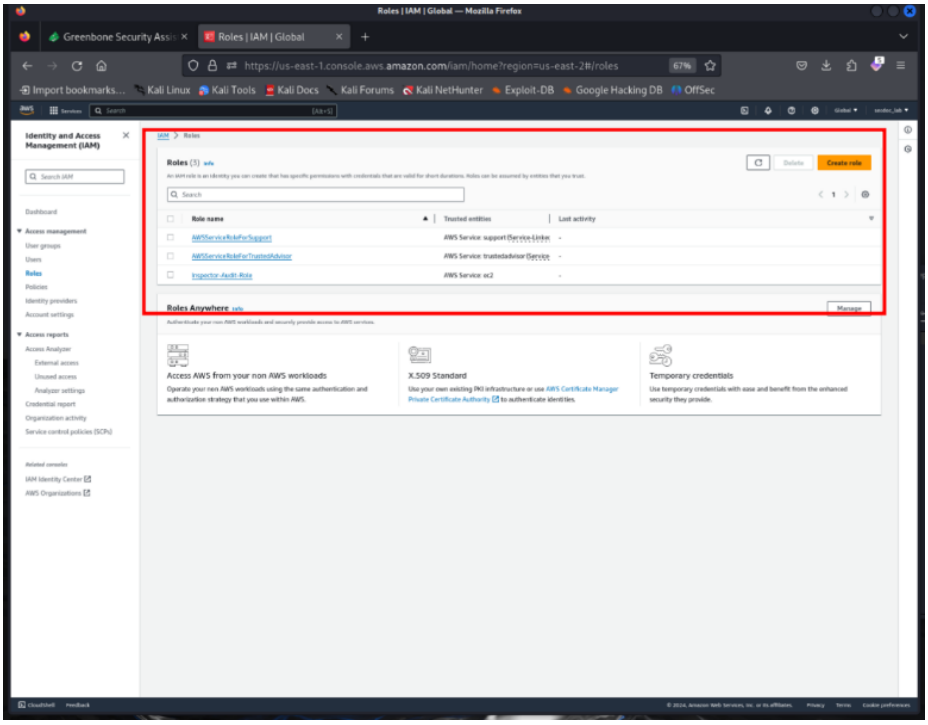


Figure 179 – Inspector Roles

38. On the **Summary** page, click **Add inline policy** and select JSON:

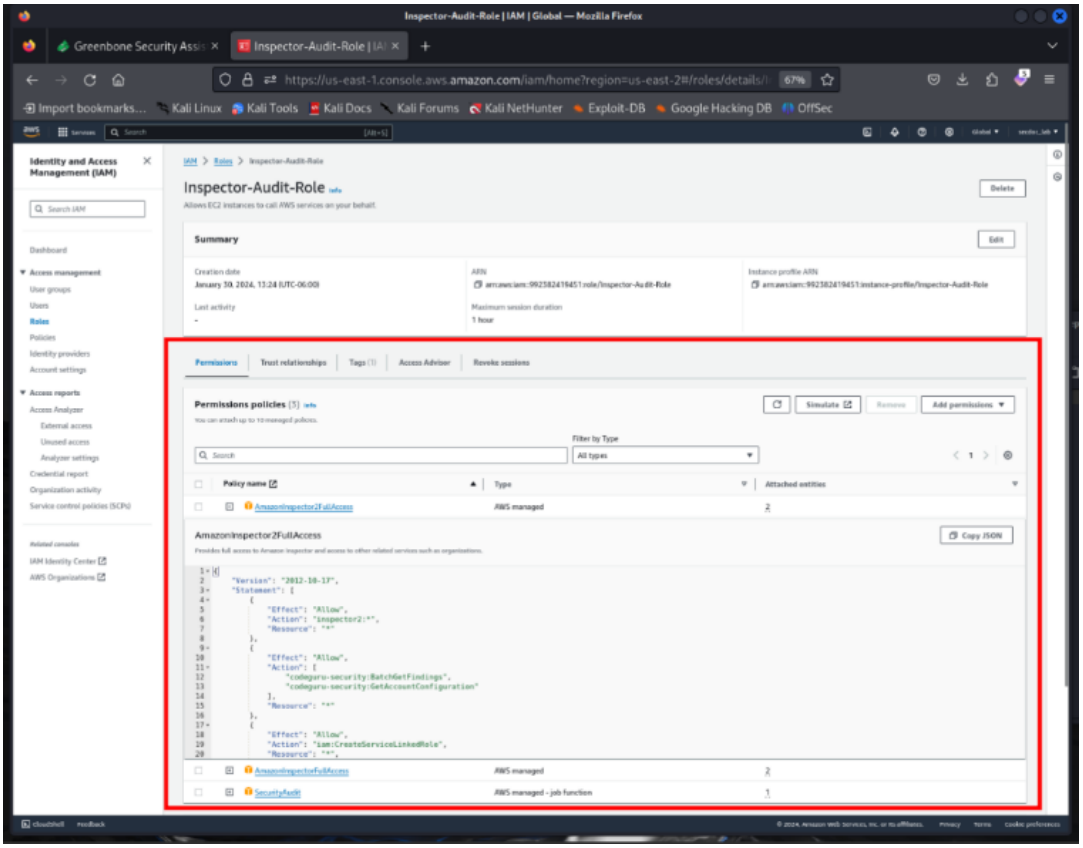


Figure 180 – Inspector Policy Permissions

39. Paste a JSON policy granting permissions to the S3 bucket where the AWS Inspector will store reports.

40. Click the **Review policy** button and then the **Save changes** button.

41. Return to the EC2 console and select the target instance again.

42. Click the **Actions** | **Instance Settings** | **Attach/Replace IAM Role** menus at the top:

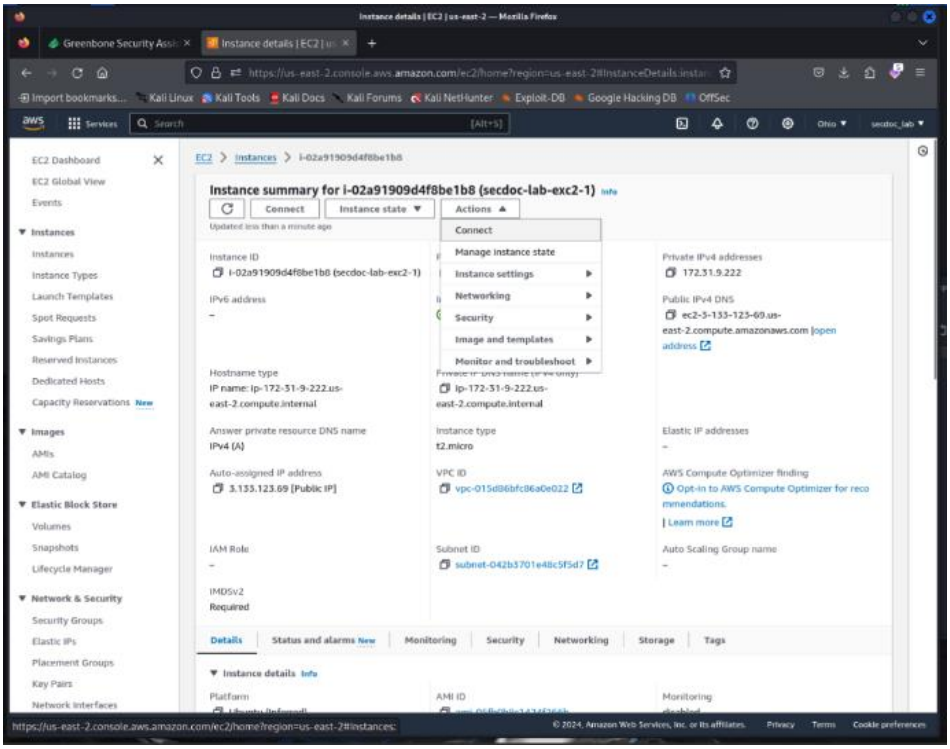


Figure 181 – Inspector Actions

43. Select the **Inspector-Audit-Role** role and click **Apply**:

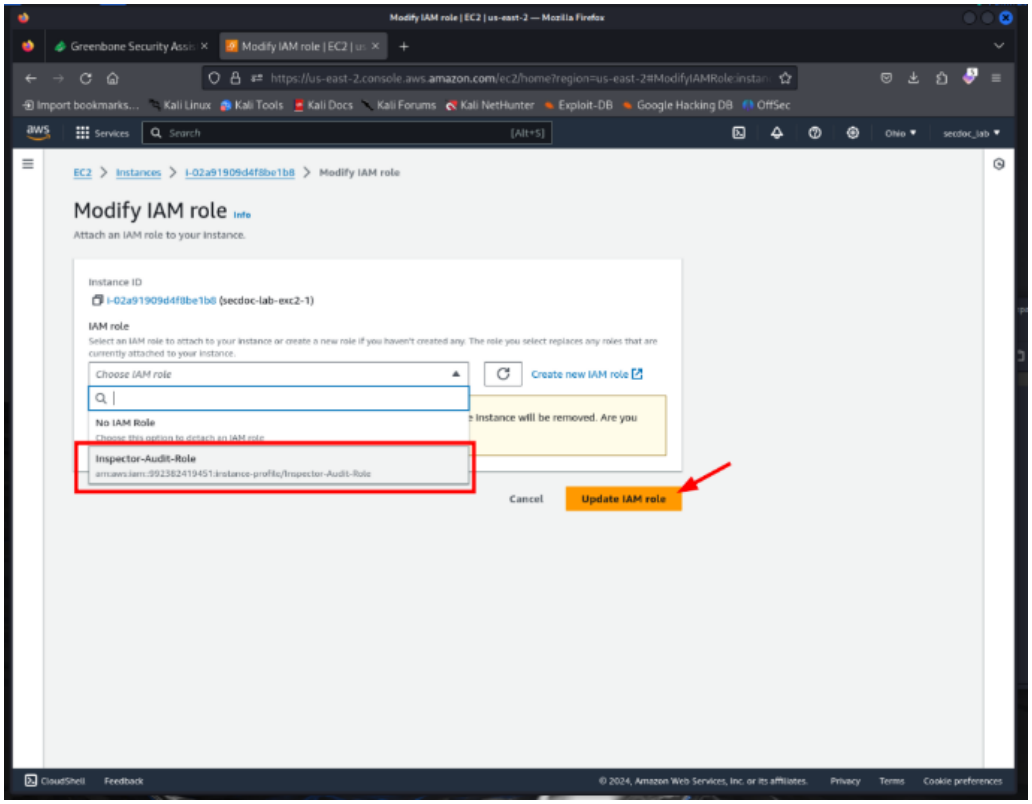
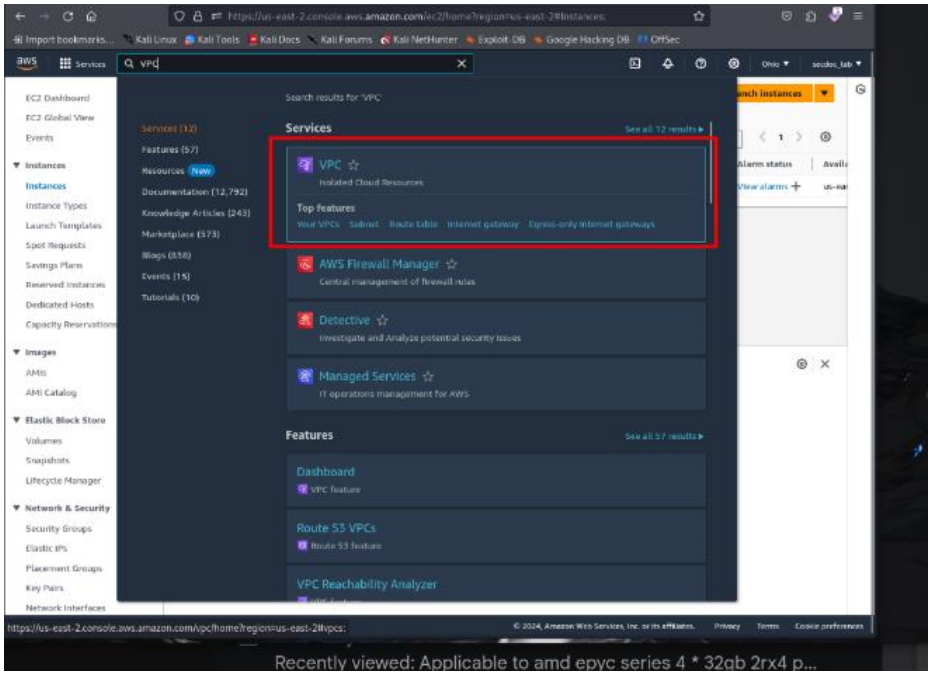


Figure 182 – Complete Audit Role Update

The EC2 instance now has the necessary permissions through the attached IAM role for Inspector assessments.

44. Let us look at an AWS Security Groups review. Click **Launch VPC Wizard** and create a new VPC with subnets according to your requirements:



(a) VPC Console

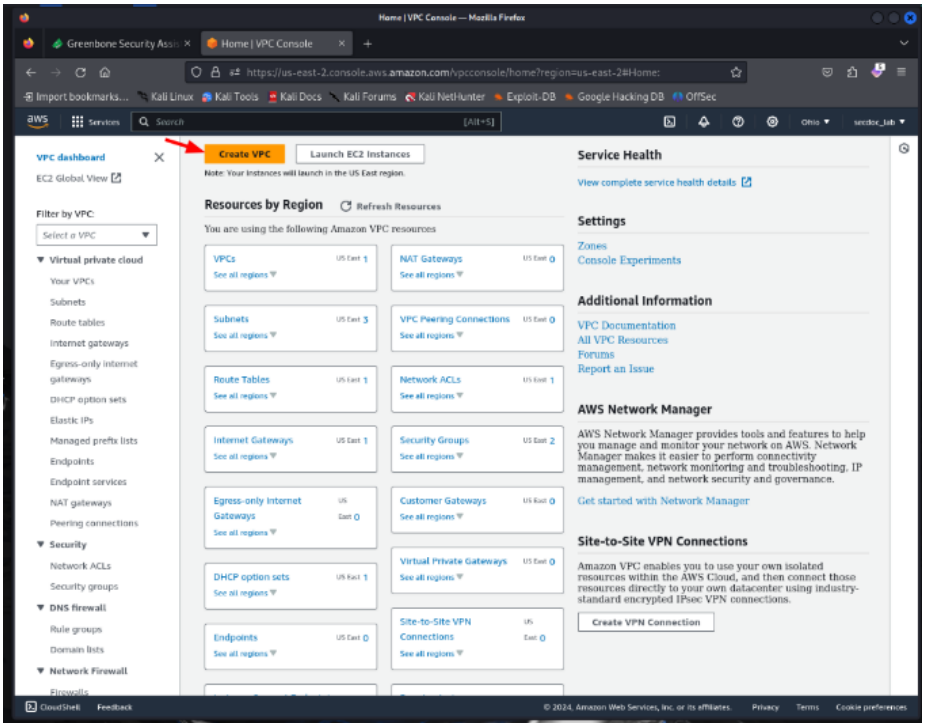


Figure 183 – Create A VPC

45. Once the VPC is created, navigate to **Security Groups** in the left sidebar:

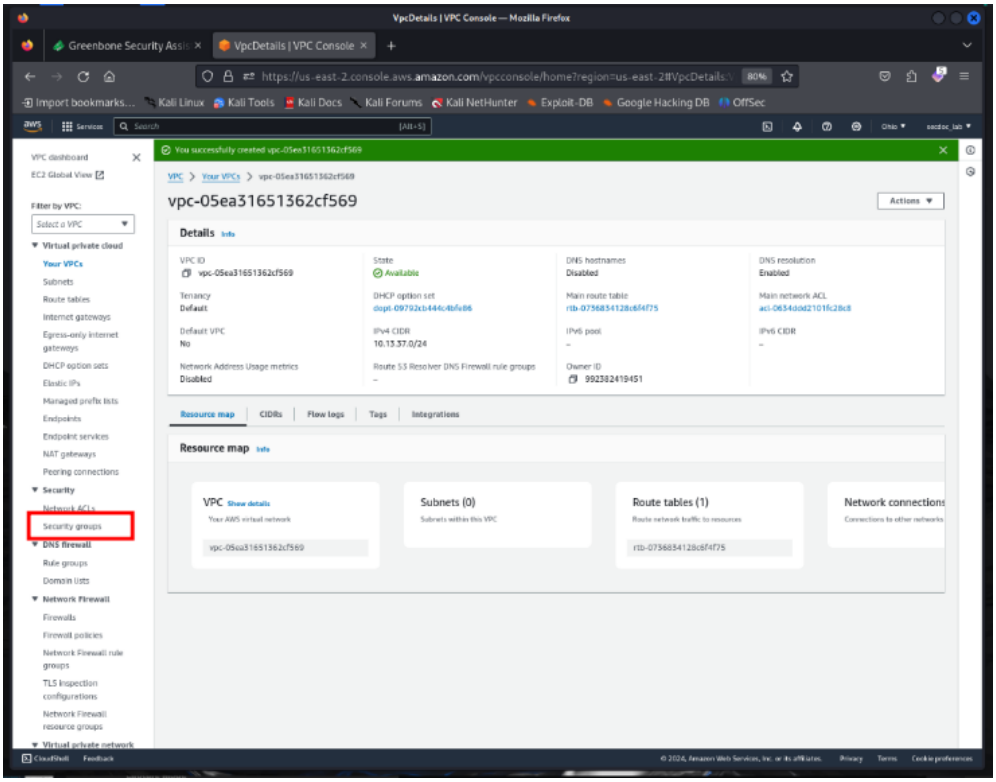


Figure 184 – Security Groups

46. Click the default security group that gets created with your VPC:

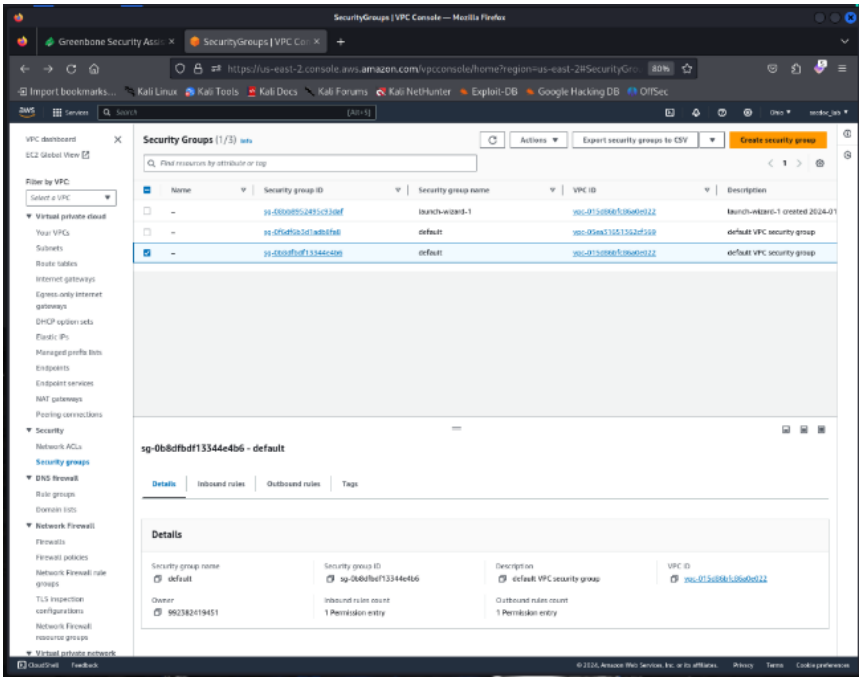


Figure 185 – Select Security Group for VPC

47. On the **Inbound rules** tab, delete any overly permissive open inbound rules like SSH open to `0.0.0.0/0`:

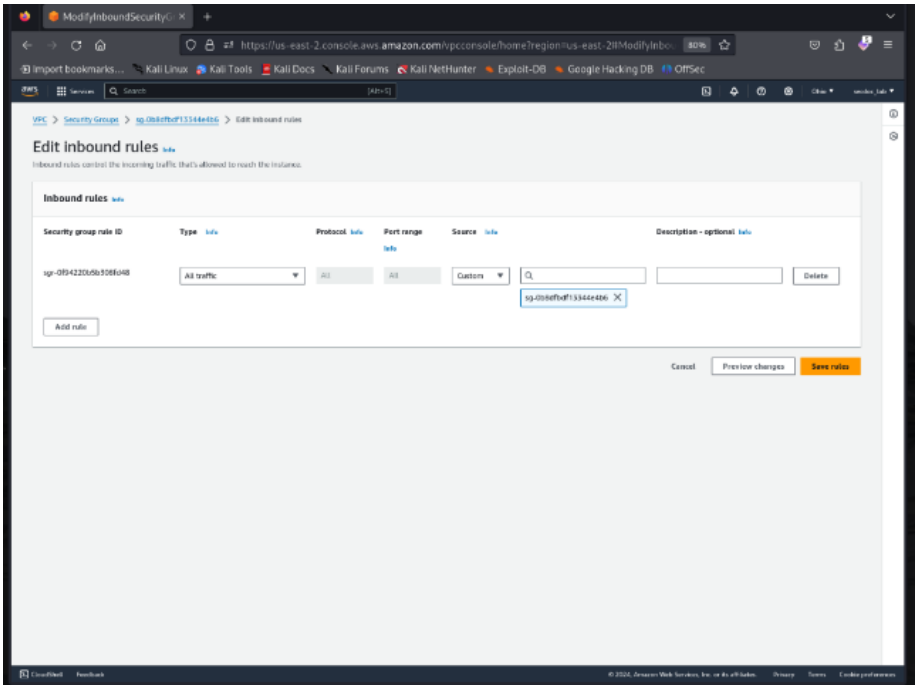


Figure 186 – Edit Inbound Rules

48. Click the outbound rules and delete any default allow all outbound rule:

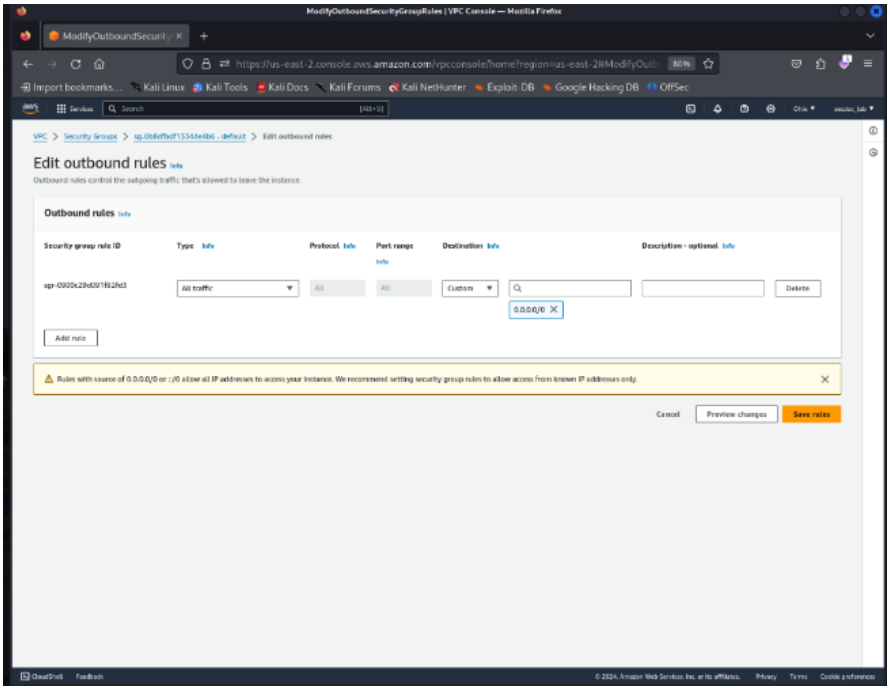


Figure 187 – Edit Outbound Rules

49. Create a new custom security group, give it a name like **web-sg**:

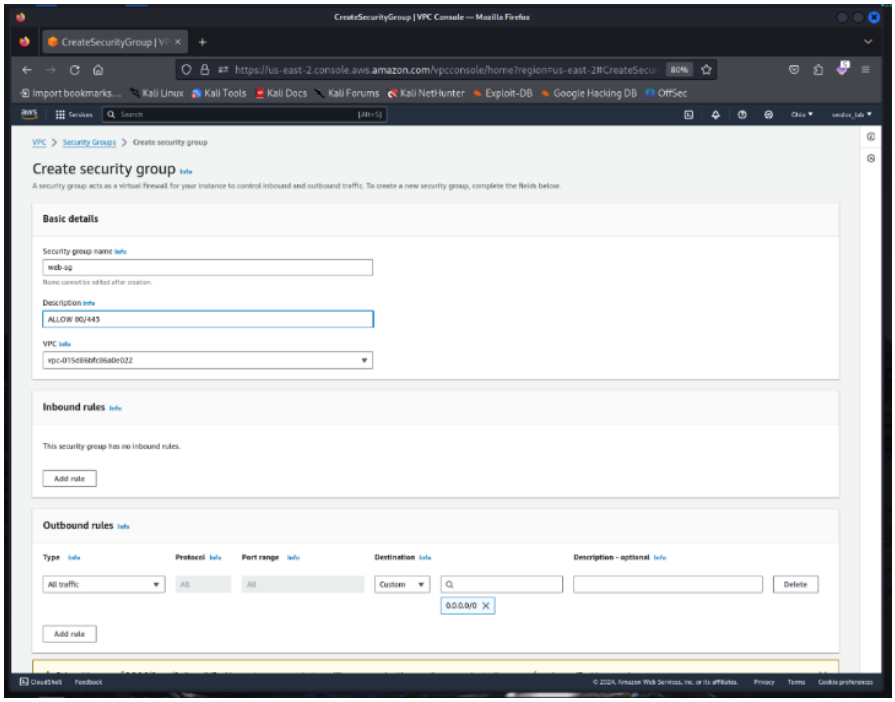


Figure 188 – Create Security Group

50. On the inbound rules tab, click **Edit rules**, **Add rule** and add the following:

A. HTTP from your local workstation IP

B. HTTPS from your local workstation IP:

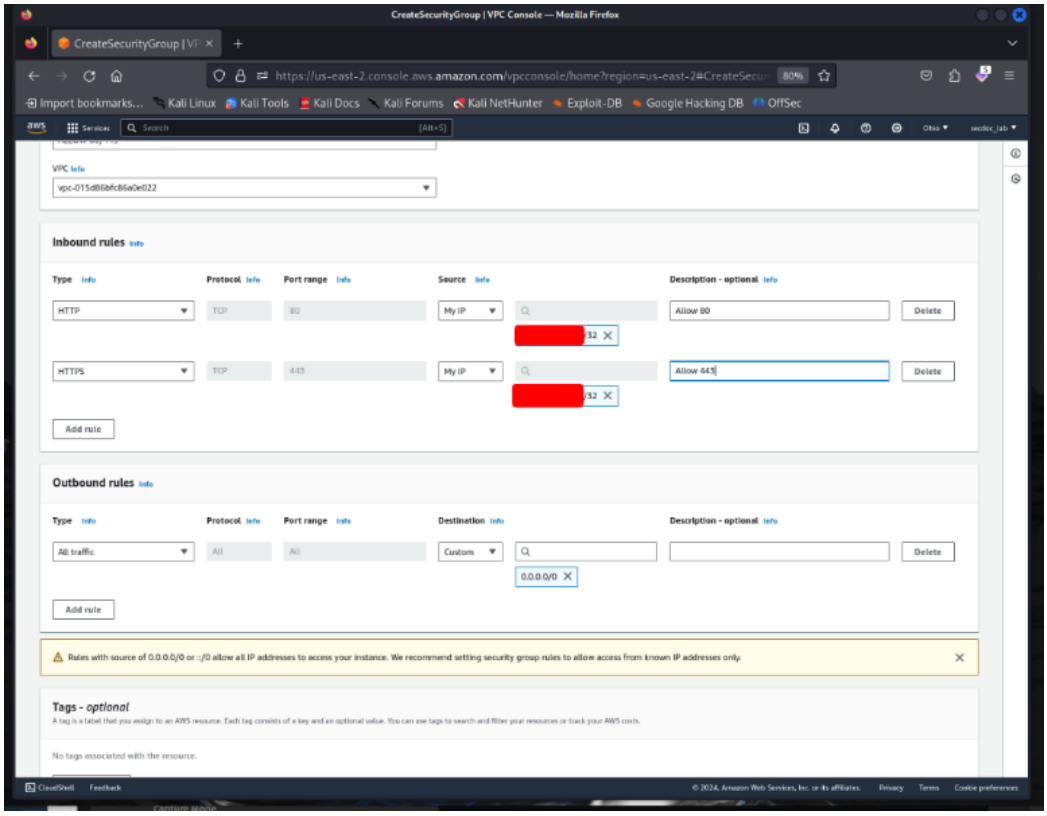


Figure 189 – Inbound/Outbound Allow Rules

51. Click **Save rules**:

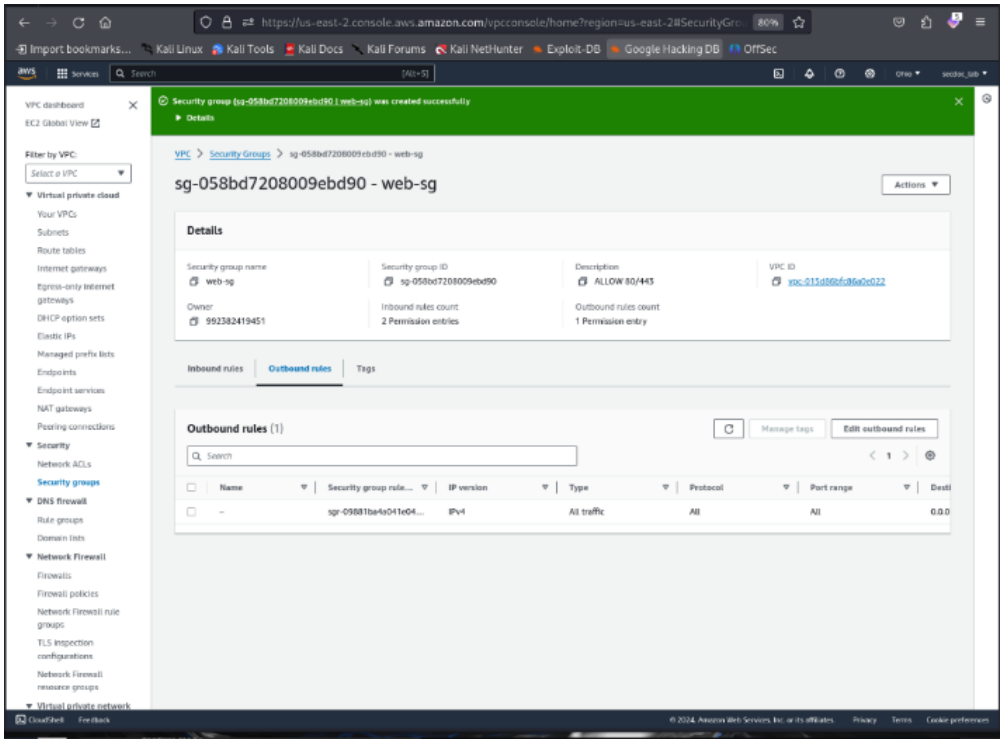


Figure 190 – Security Group Dashboard

52. Repeat to create additional locked down security groups for resources grouped logically, geographically etc.

53. Launch new EC2 instances, RDS databases etc. and assign appropriate regional, role-based security groups.

54. By leveraging VPC security groups that open just required minimum ports to authorized source IP ranges, you can drastically reduce your attack surface following least privilege access principles.

55. Now let us look at the AWS Inspector Setup. Open the Inspector service within the AWS console:

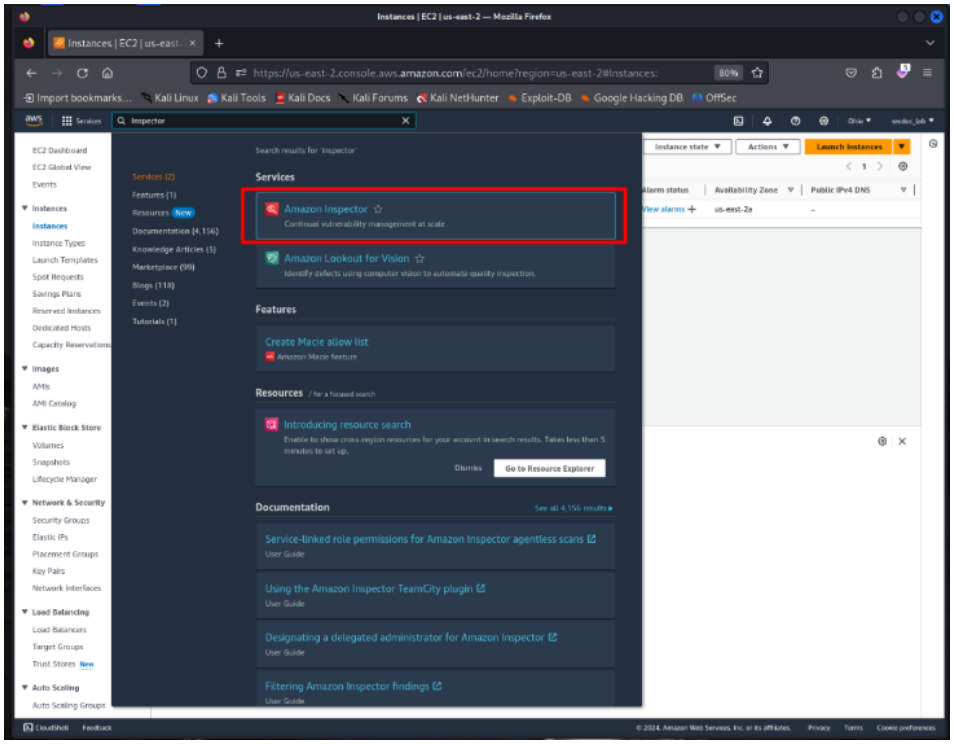


Figure 191 – Amazon Inspector Console

56. Click **Get Started** under **Inspector Assessments**:

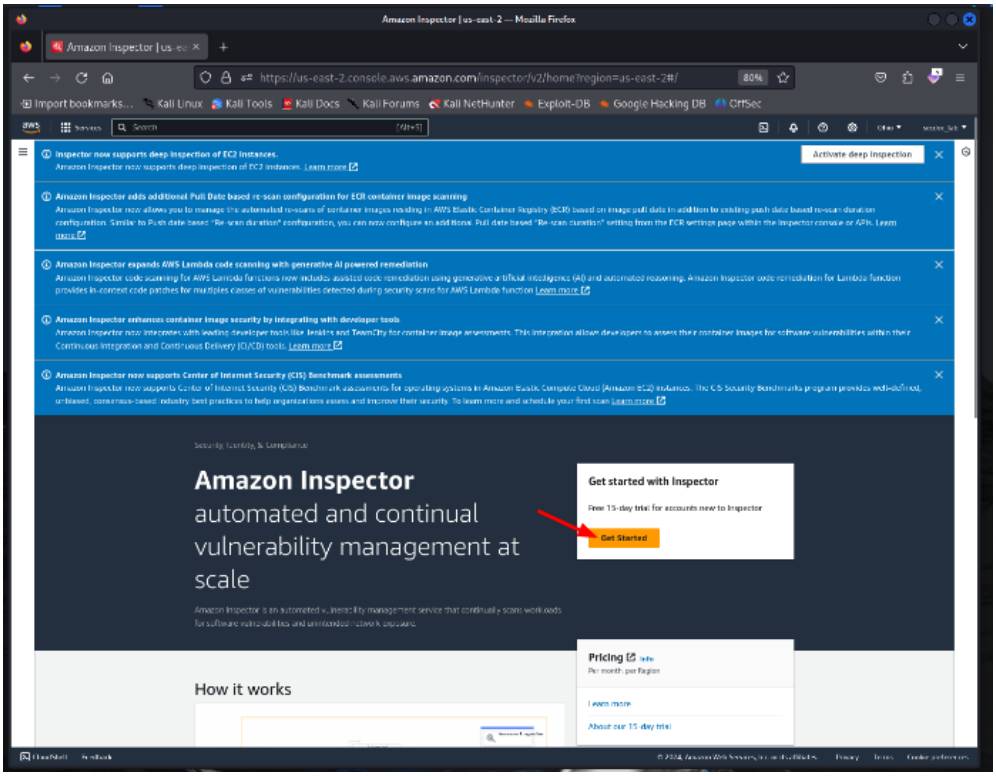


Figure 192 – Getting Started with Inspector

57. Specify an assessment target by choosing an existing resource group or creating a new resource group:

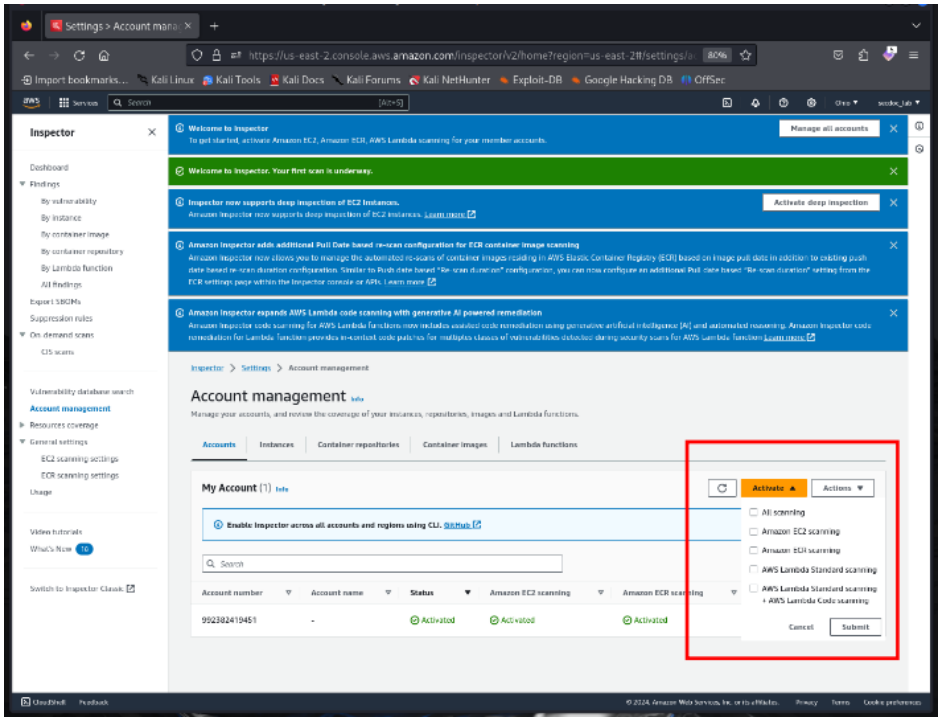


Figure 193 – Defining Inspector Targets

58. Select the EC2 instance(s) you want to scan by instance ID or tags.

59. Click **Next**:

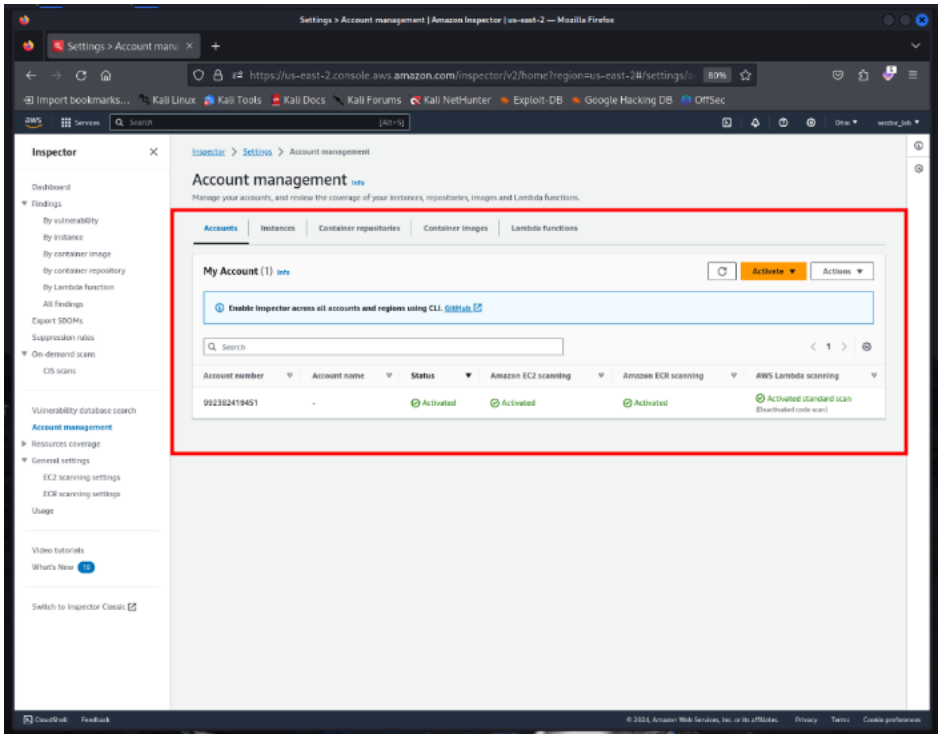


Figure 194 – Account Management

60. Select an assessment template (e.g. **Common Vulnerabilities and Exposures**) and click **Next**:

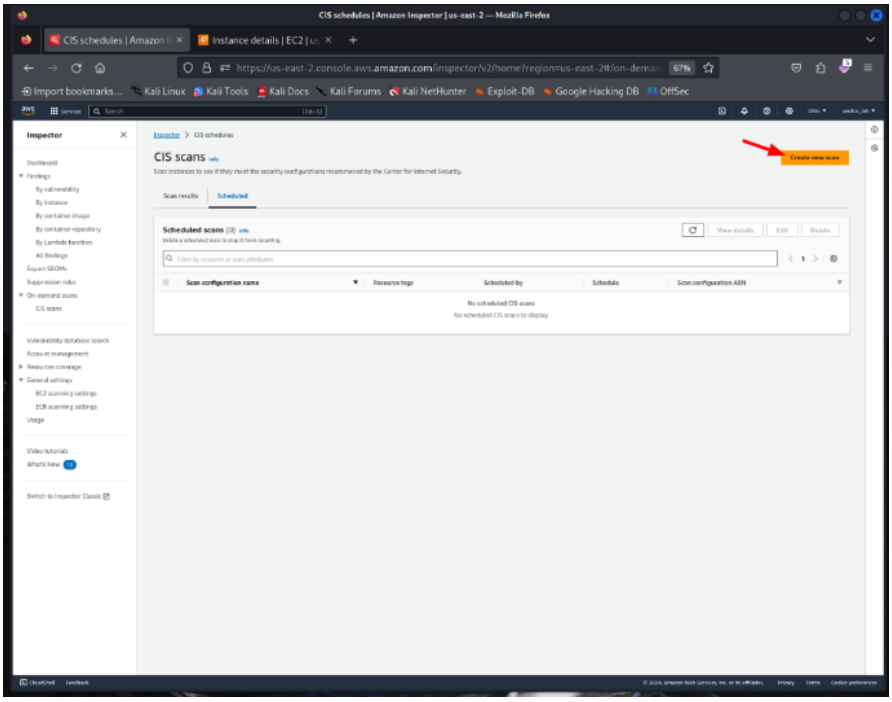


Figure 195 – Scan Type

61. Configure any optional scope downs and advanced assessment settings (or leave as default) and click **Next**:

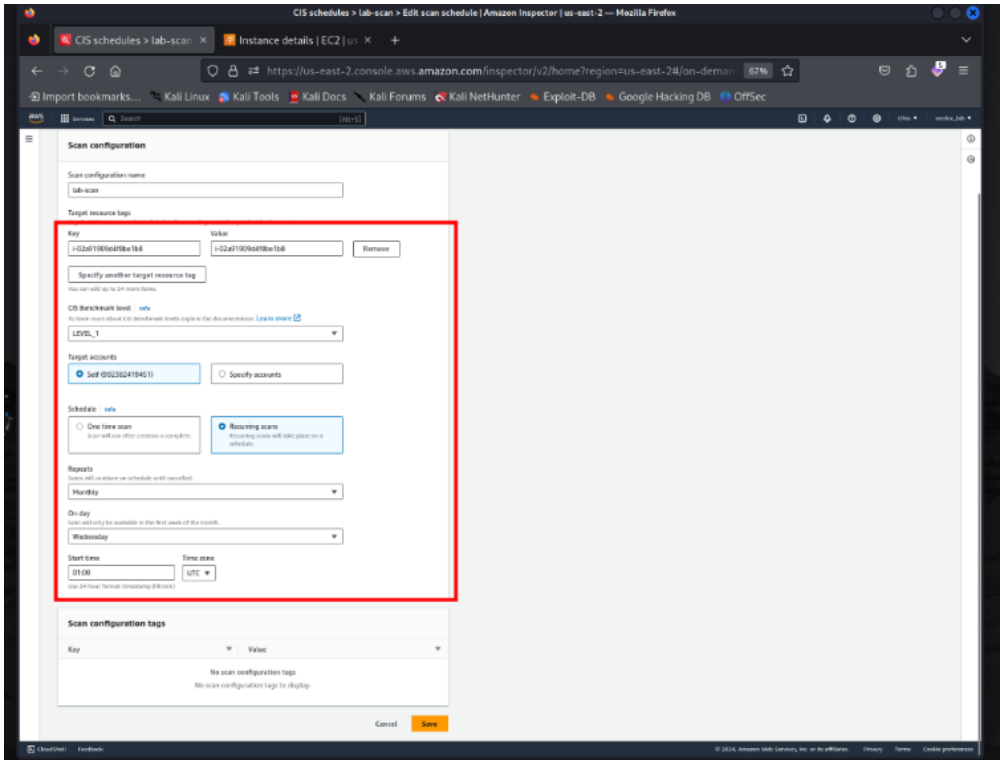


Figure 196 – Scan Configuration

62. Review the details on the final page, enter an Assessment name, and click **Start assessment**.

63. Now, let us look at the AWS CloudTrail setup. Open the CloudTrail service within the AWS console:

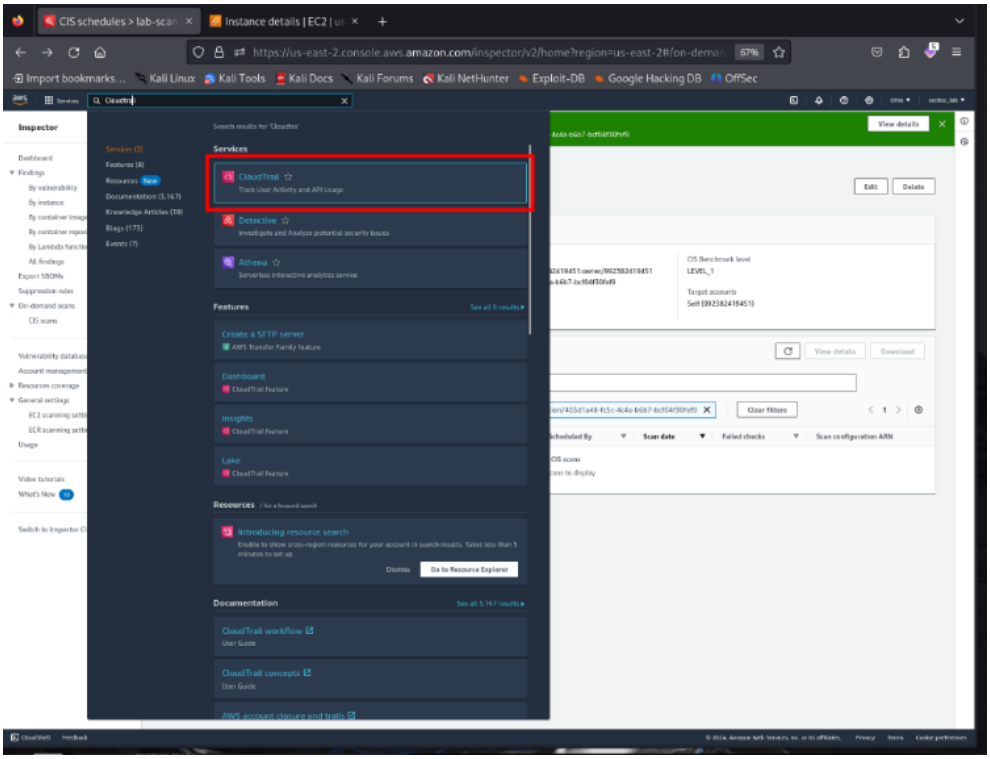


Figure 197 – CloudTrail Console

64. Click **Create trail**:

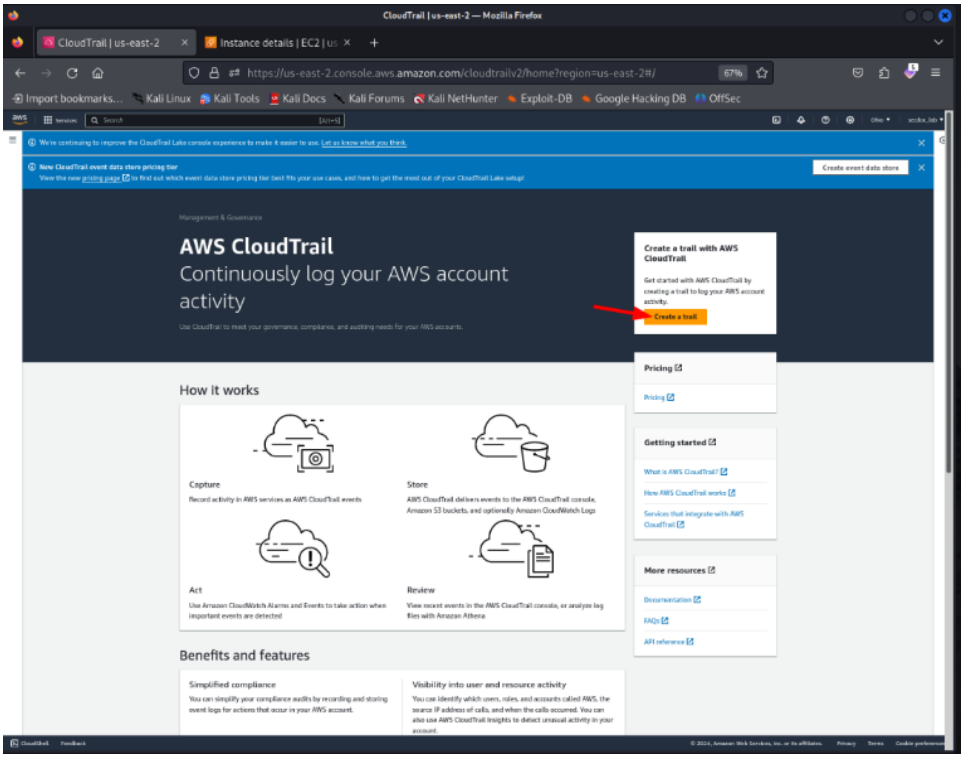


Figure 198 – Getting Started with CloudTrail

65. Enter a Trail name and optionally specify Tags.

66. For Storage location, create or choose an existing S3 bucket to store logs.

67. Click **Next**.

68. On the next page, click **Next** again to start logging using the default event selectors:

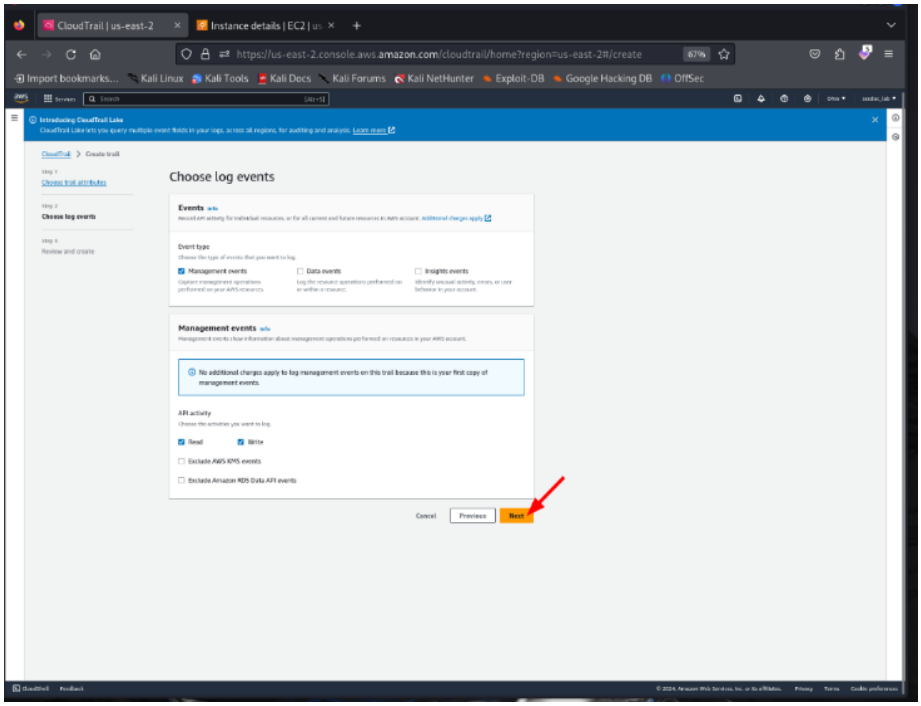


Figure 199 – Managing Events

69. Review the trail details and click **Create trail** to complete setup:

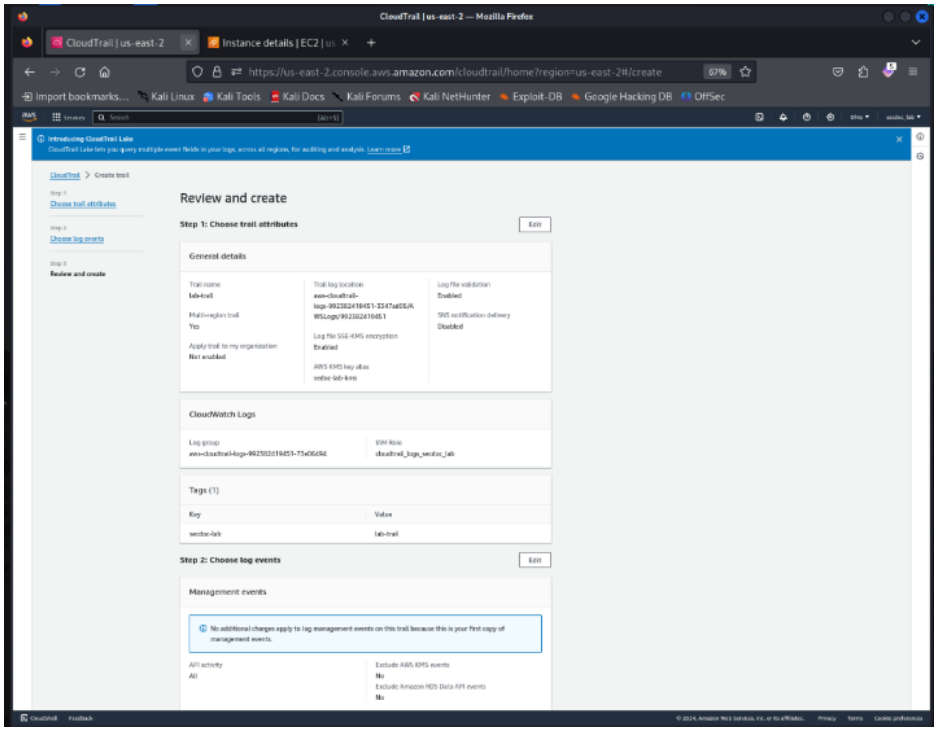


Figure 200 – Creating a Trail

70. The Inspector assessment will now run against the defined EC2 instance(s), while CloudTrail will begin recording management events for monitoring.

This lab provides a foundational understanding of how to utilize AWS tools for ensuring cloud security. It covers the deployment of resources and the configuration of native AWS security services. Adapting this lab to another cloud provider would involve similar concepts but would use the specific tools and services provided by the respective cloud platform (e.g., Azure Security Center, Google Security Command Center).

Layered cloud-native tools extend security visibility, data protection, and threat prevention to reduce risk introduced by cloud adoption.

Cybersecurity governance and compliance tools

Cybersecurity governance and compliance are critical components of an organization's overall security posture, ensuring that security practices align with business objectives

and regulatory requirements. The lab below focuses on the implementation and usage of **governance, risk management, and compliance** (**GRC**) tools.

Lab: Implementing and configuring a GRC tool (using an open-source GRC tool like Eramba)

Eramba is a comprehensive and intuitive integrated risk management platform designed to help organizations streamline and automate their GRC processes. It provides a centralized framework for managing risks, controls, audits, incidents, and compliance requirements across various domains, including IT, finance, operations, and security. With Eramba, organizations can gain a holistic view of their risk landscape, assess and prioritize risks, implement effective controls, and monitor compliance with internal policies and external regulations. The platform offers a user-friendly interface, customizable workflows, and robust reporting capabilities, enabling teams to collaborate effectively, make informed decisions, and demonstrate compliance to stakeholders and auditors. Eramba's modular architecture allows organizations to tailor the solution to their specific needs, integrating with existing tools and systems to create a seamless GRC ecosystem. Whether you are a small business or a large enterprise, Eramba empowers you to proactively manage risks, optimize controls, and drive a culture of continuous improvement in your governance and compliance efforts.

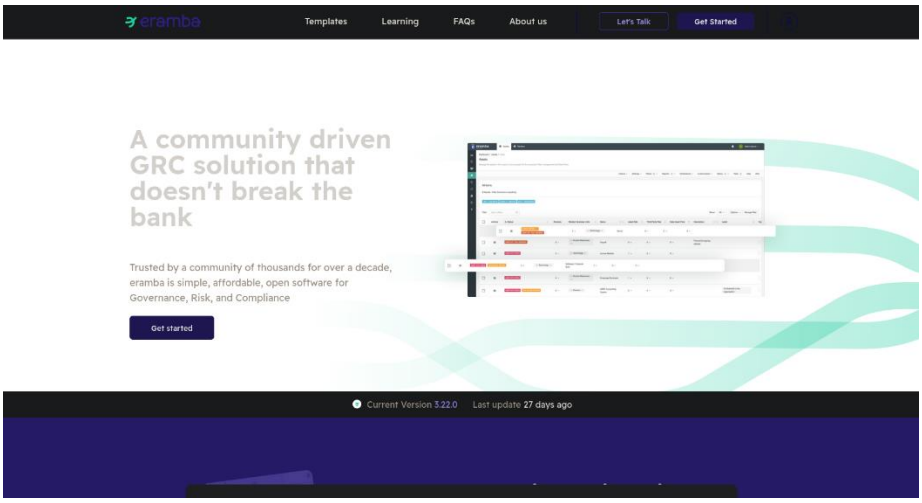


Figure 201 – Eramba

The prerequisites are:

- A virtualization platform (e.g., VirtualBox, VMware) to host the GRC tool.
- A virtual machine (VM) or container with a supported operating system (e.g., Ubuntu, CentOS). In this lab I will be installing on Debian 12.
- Installation of Docker
- Download the installation package for an open-source GRC tool like Eramba. The community edition and install instructions can be found at https://www.eramba.org/get-community.

Let us look at the steps.

1. Prepare a VM or a container within your virtualization platform.
2. Allocate sufficient resources (CPU, memory, storage) based on the requirements of the GRC tool and intended usage.
3. Configuring the Docker APT Repository - https://docs.docker.com/engine/install/debian/.
4. On a new host system, the Docker Engine apt repository must be initialized before installing Docker for the first time. This involves setting up the repository configuration to enable installing and updating Docker packages later on.
5. The process consists of adding Docker's GPG key to verify package integrity and then adding the stable apt repository definition from where Docker can be installed. With the repository configured, the Docker Engine can then be installed from the maintained repository instead of upstream sources. Subsequently, Docker can be kept up-to-date by upgrading packages from the same apt repo.
6. By setting up the optimized Docker apt repository on Debian-based systems before installation, it streamlines engine deployment and patching through a trusted and consistent package source going forward:

```
sudo apt-get update

sudo apt-get install ca-certificates curl
```

```
sudo install -m 0755 -d /etc/apt/keyrings

sudo curl -fsSL
https://download.docker.com/linux/debian/gpg -o
/etc/apt/keyrings/docker.asc

sudo chmod a+r /etc/apt/keyrings/docker.asc

 # Add the repository to Apt sources:

echo \

  "deb [arch=$(dpkg --print-architecture) signed-
by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/debian \

  $(. /etc/os-release && echo "$VERSION_CODENAME") stable"
| \

  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update
```

7. To install the latest version of the Docker Packages, run:

```
sudo apt-get install docker-ce docker-ce-cli containerd.io
docker-buildx-plugin docker-compose-plugin
```

8. Once Docker Engine installation completes, verify it is working properly by running a simple test container.

9. Launch the standard Docker hello-world image to confirm the docker daemon starts correctly and can successfully pull images, run containers, etc:

```
docker run hello-world
```

10. The output should show the test message and exit cleanly without errors.

11. Running this basic Docker image test provides a quick validation that the Engine was installed correctly and has connectivity to access images along with fundamental container runtime functionality. The ability to fetch and run this test hello-world image confirms that the Docker setup was successful.

12. Let us now look at Eramba's installation. Eramba provides a Docker deployment script that installs a container stack running the Eramba GRC platform and its dependencies:
    - MySQL (Database)
    - Redis (Caching)
    - Eramba (Web Application)
    - Cron (Batch Jobs)

    It utilizes 3 persistent volumes:
    - `data`: Holds Eramba application data
    - `app`: Stores Eramba application files
    - `db-data`: Contains MySQL database files
13. To deploy Eramba with Docker, run:

```
git clone https://github.com/eramba/docker
```

14. By leveraging Docker to containerize the application and its components, Eramba can be deployed consistently across environments without dependency hassles.
15. After cloning the Eramba Docker GitHub repository, navigate into the docker directory:

```
cd docker
```

16. Update default database credentials
17. Edit `.env` file using Nano or Vi:

```
nano .env
```

18. Change `DB_PASSWORD` and `MYSQL_ROOT_PASSWORD` variables
19. Validate external connectivity

20. Containers must reach Eramba servers for registration, updates etc.
21. Test connectivity with:

```
curl https://support-v3.eramba.org/ping.html
```

    Offline mode is unsupported.

22. By customizing credentials in `.env` and ensuring connectivity to Docker Hub and Eramba repositories, the containers can securely access private dependencies while checking for updates. Offline deployments are currently unavailable - external access is required.
23. You are now Ready to run the Docker Compose command:

```
docker compose -f docker-compose.simple-install.yml up -d
```

24. Follow the installation guide provided by Eramba, https://www.eramba.org/, to set up the GRC platform.
25. **Login screen**: You should now be able to login to eramba using `https://{IP ADDRESS}:8443`
26. The default login credentials are:
    - Username: admin
    - Password: admin

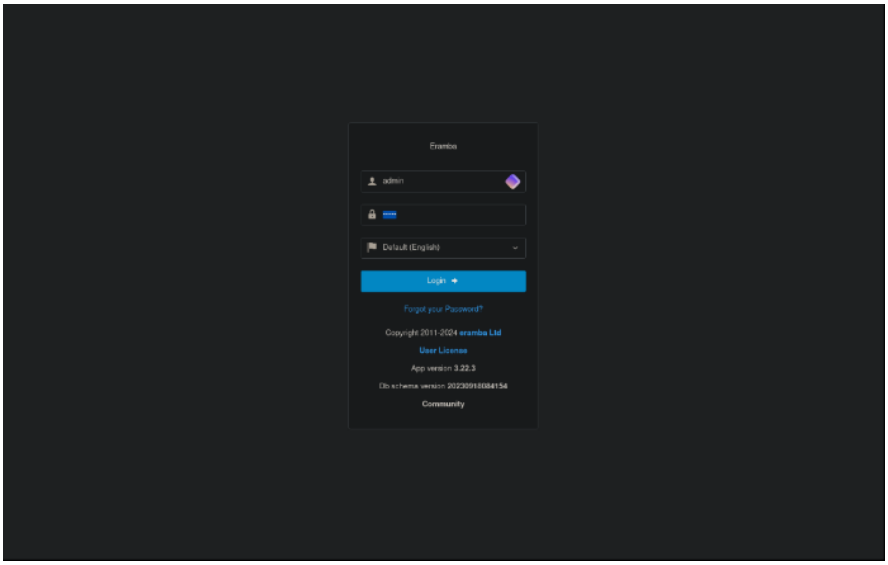


Figure 202 – Eramba Login

27. After login you will be asked to change the following:

- Default Password
- Default Admin Email:

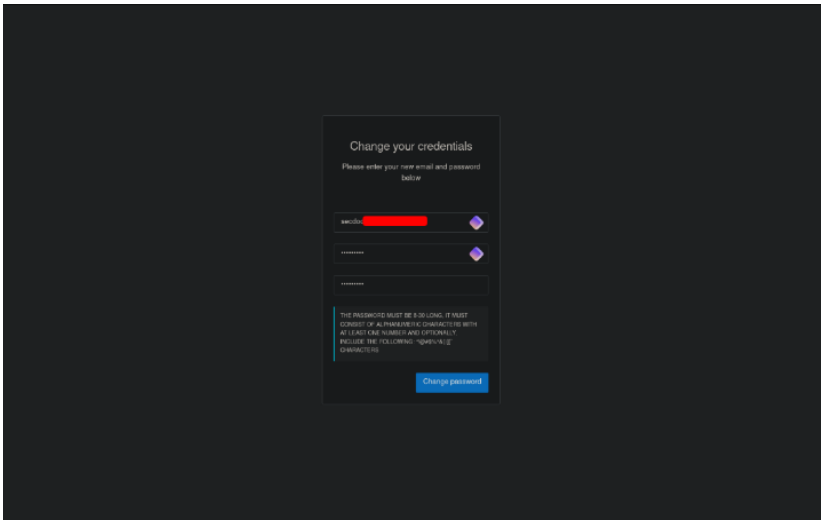


Figure 203 – Eramba Initial User

28. Community users will be asked to insert their email for verification.

29. A validation token will be sent to the email specified:

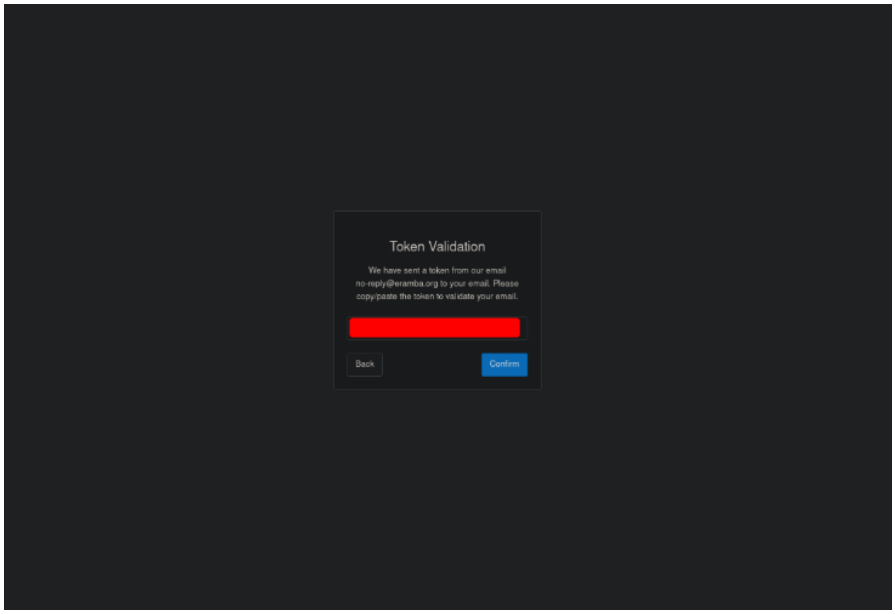


Figure 204 – Token Validation

30. With the installation of the system, you can begin building out your GRC process including report generation and more. Since the installation is the community addition, there is some limitation to what can be done but there is a lot of that can be learned and leveraged through Eramba. You can do a deep dive on the aspects of Eramba at https://www.eramba.org/learning:

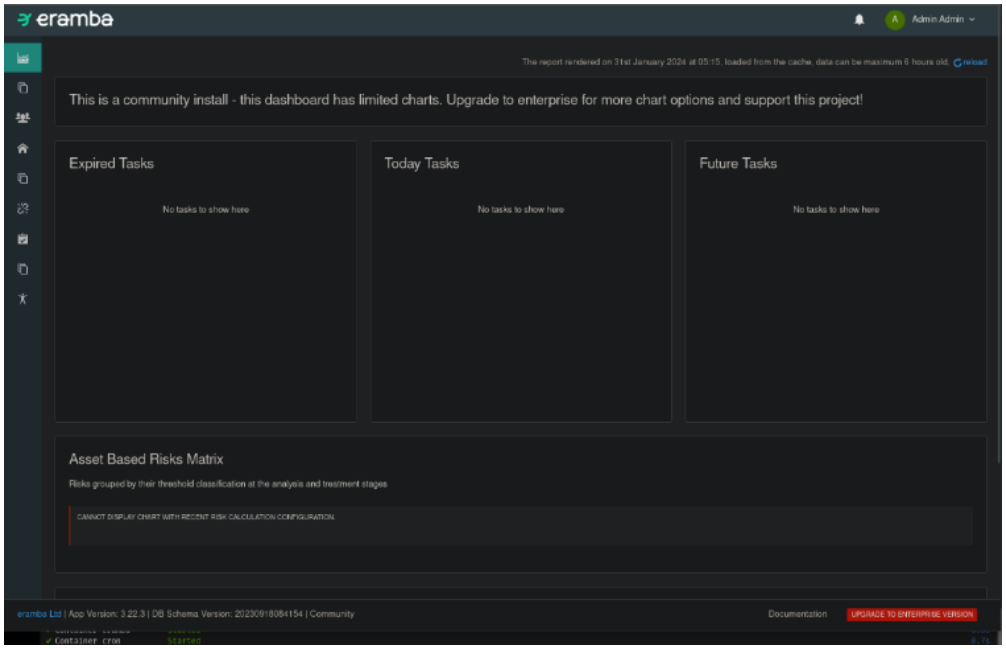


Figure 205 – Eramba Dashboard

31. Let us now see the Compliance Management setup. Import compliance packages relevant to your organization (e.g., ISO 27001, GDPR, PCI-DSS) into Eramba.
32. Map the compliance requirements to your organization's processes and controls.
33. Let us look at risk assessment frameworks. Define a risk assessment methodology in Eramba (e.g., qualitative, quantitative).
34. Create risk assessment criteria, such as likelihood and impact scales.
35. Now, policy management. Use Eramba to draft, review, and approve security policies.
36. Set up a policy review schedule and reminders for periodic updates.
37. Create a catalog of security controls within Eramba.

38. Link the controls to applicable policies and compliance requirements.
39. Configure the incident management module in Eramba.
40. Set up incident reporting channels and response plans.
41. Schedule internal audits within Eramba to assess compliance with policies and standards.
42. Monitor compliance status through dashboards and automated alerts.
43. Generate reports for different stakeholders (e.g., IT, executives, auditors).
44. Customize dashboards to highlight key compliance metrics and risk status.
45. Document and assess third-party relationships and risks within Eramba.
46. Integrate third-party compliance information into the overall risk assessment.
47. Plan and track security awareness training sessions.
48. Record employee training status and schedule reminders for recurring training.
49. Use feedback and audit results to improve security practices.
50. Update risk assessments, controls, and policies accordingly.
51. If this was a test implementation, decommission the environment properly.
52. Document the configuration process, how the tool is used, and any issues encountered or insights gained during the lab.

This lab introduces the participant to the fundamental aspects of using a GRC tool for cybersecurity governance and compliance management. By importing compliance frameworks, creating policies, managing risks, and establishing auditing and reporting mechanisms, the participant gains hands-on experience that is crucial for managing cybersecurity governance and compliance in a real-world environment.

These tools provide centralized visibility and management of critical security program artifacts for consistency, accuracy, and compliance.

Penetration testing and red team tools

A lab setup for penetration testing and Red Teaming involves configuring a controlled environment where security professionals can safely conduct attacks against systems, networks, and applications to identify vulnerabilities. This lab will demonstrate the use of Kali Linux, a popular penetration testing platform, alongside tools such as Metasploit for vulnerability exploitation and Nmap for network scanning.

Lab: Penetration testing with Kali Linux and Metasploit Framework

Penetration testing, also known as pen testing or ethical hacking, is the practice of simulating real-world cyber attacks on computer systems, networks, and web applications to identify and exploit vulnerabilities before malicious actors can take advantage of them. Kali Linux, a powerful and widely used open-source operating system, is specifically designed for digital forensics and penetration testing. It comes pre-installed with a vast array of security tools, including the renowned Metasploit Framework, which is a powerful and versatile tool used for conducting penetration testing, exploit development, and vulnerability analysis. Together, Kali Linux and Metasploit Framework provide pen testers and security professionals with a comprehensive arsenal of tools and techniques to assess and strengthen the security posture of their systems, networks, and applications.

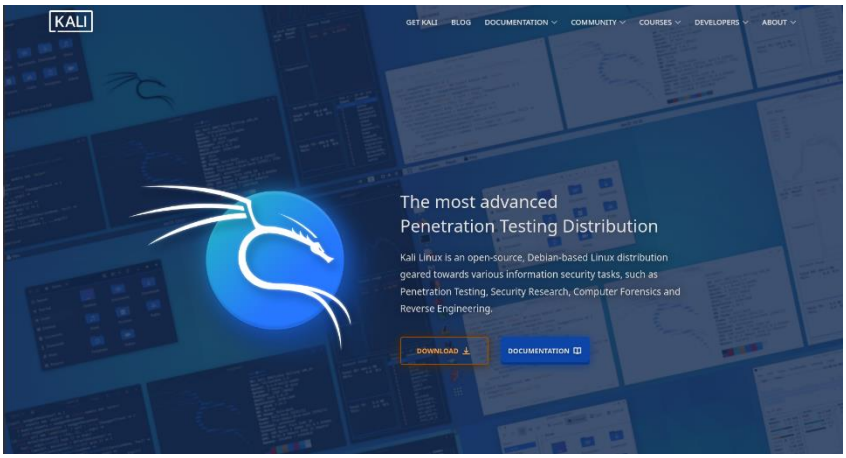


Figure 206 – Kali Website

The prerequisites are:

- A virtualization platform (e.g., VMware, VirtualBox) with a virtual network configured for the lab environment.
- Download the Kali Linux virtual machine image from the official website, https://www.kali.org/.
- Target VMs for penetration testing, like Metasploitable or OWASP Broken Web Applications, which are intentionally vulnerable.
- Permission and ethical guidelines established if not using provided vulnerable applications and in a real-world scenario.

Let us look at the steps beginning with the Kali Linux setup:

1. Import the Kali Linux VM into your virtualization platform.
2. Configure the network settings to ensure it is in the same virtual network as the target VMs but isolated from your production network.
3. Import the target VMs (like Metasploitable) into the virtualization platform:

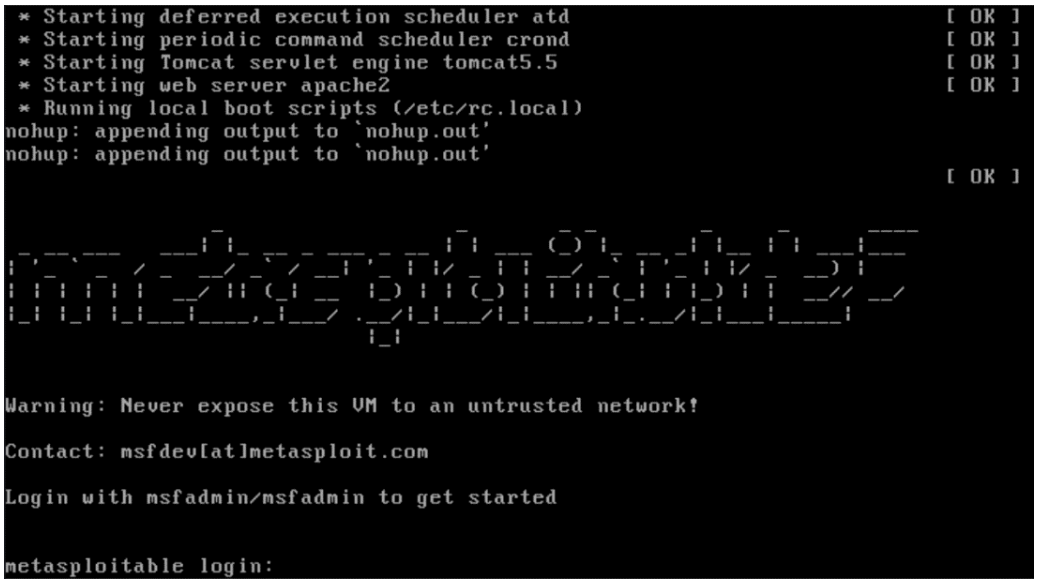


Figure 207 – Metasploitable VM

4. Verify the network configuration for connectivity with the Kali Linux VM:

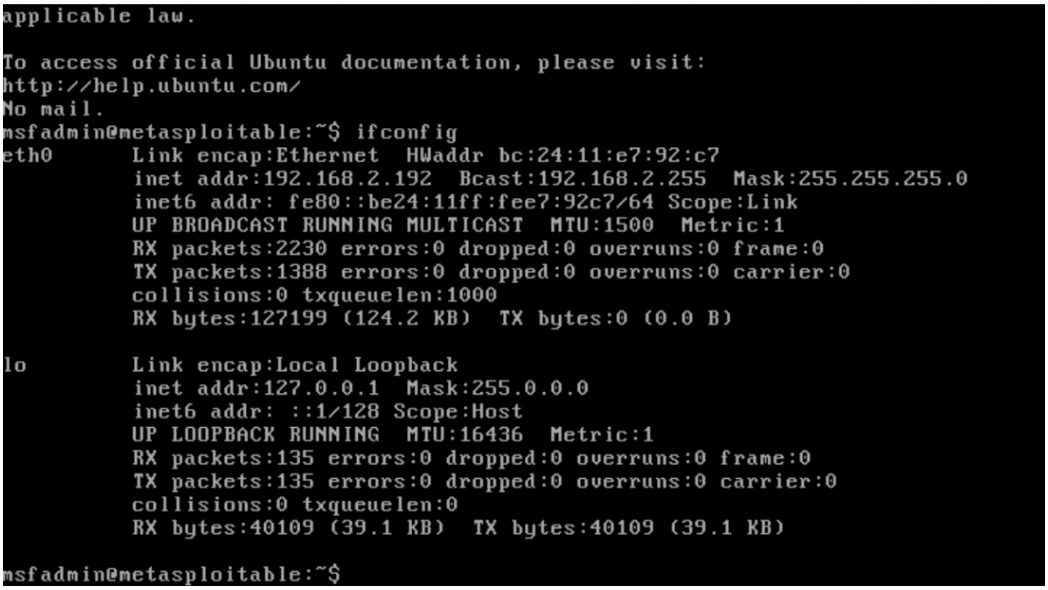


Figure 208 – ifconfig of Metasploitable VM

5. Start the Kali Linux VM and open the terminal.

6. Use Nmap to perform a network scan to discover active hosts and open ports:

```
sudo nmap -sV -T4 192.168.X.X/24
```

7. Analyze the output for potential targets:

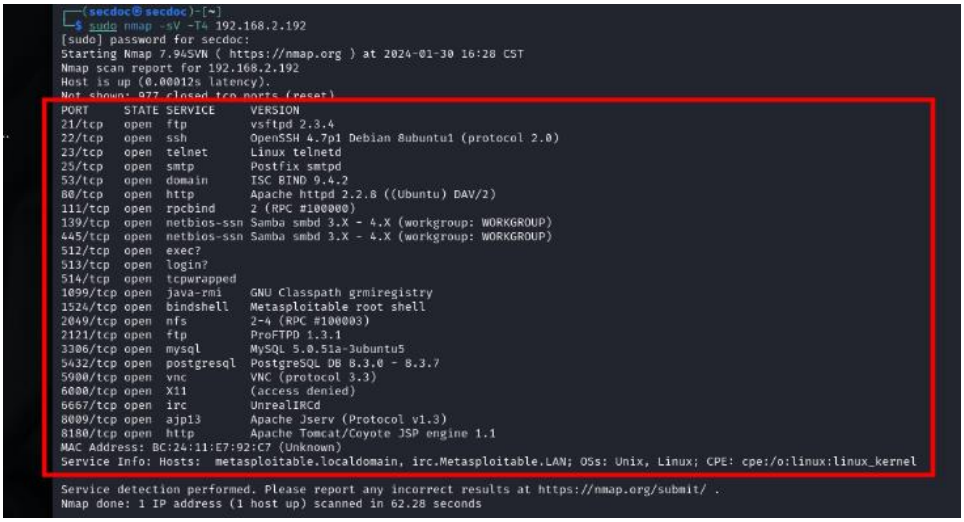


Figure 209 – NMAP scan of Metasploitable VM

8. Use a tool like OpenVAS on Kali Linux to scan the target VMs for known vulnerabilities. This can be a useful additional information gather step, but depending on the rules of engagement, may be something that could create *too much* noise for the engagement.

9. Review the vulnerability scan results, if appropriate, to prioritize targets for exploitation.

10. Open the Metasploit console on Kali Linux by typing `msfconsole` in the terminal:

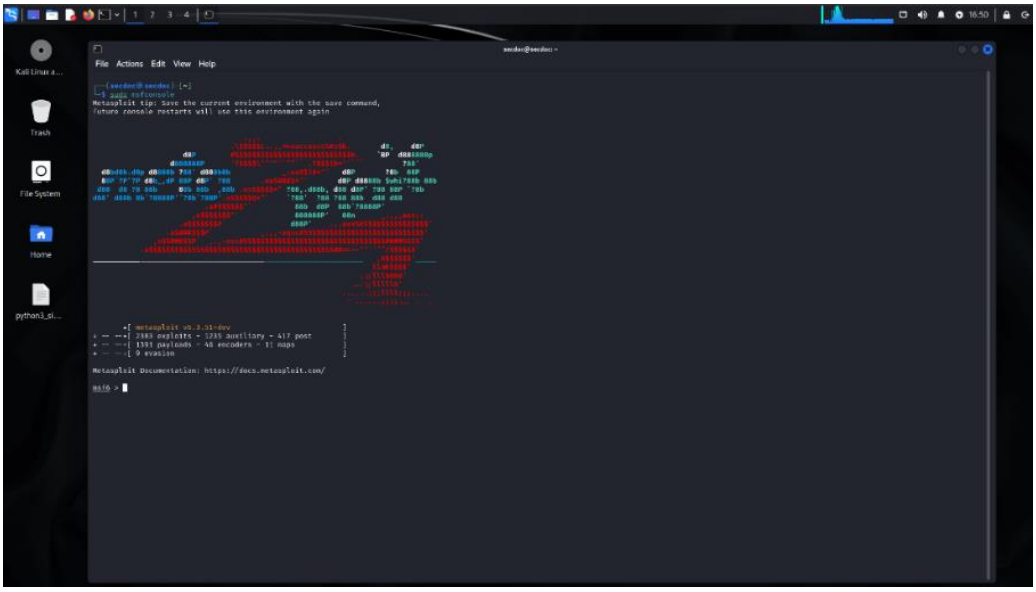


Figure 210 – Metasploit Console on Kali

11. Search for appropriate exploits for the vulnerabilities found, for example, target vsftpd version 2.3.4:

```
search type:exploit platform:[platform] [vulnerability details]
```

12. Configure and launch the exploit against a chosen target:

```
use exploit/[exploit name]

set RHOSTS [target IP]

set PAYLOAD [payload name] #This may not be necessary as with the example shown

exploit
```

13. If successful, a shell or user session should be opened on the target system:

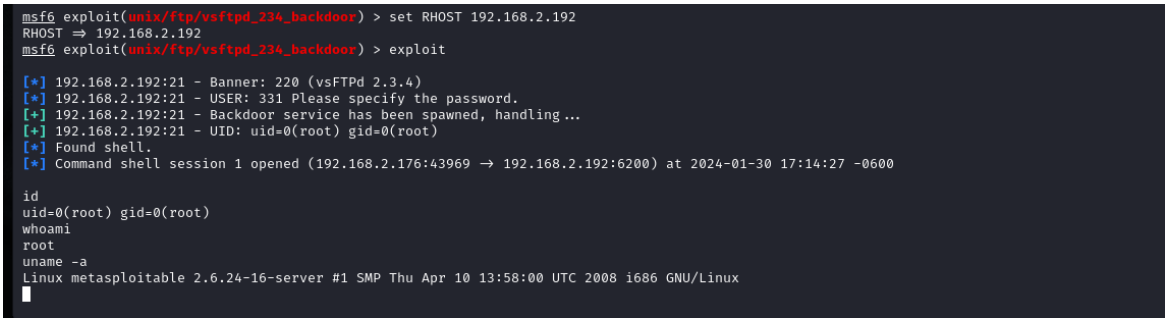


Figure 211 – Metasploitable VM Remote Reverse Shell via Metasploit

14. Document any sensitive information found or actions taken.
15. Document every step taken during the exploitation phase, including the output of each command.
16. Create a detailed report with the findings, including how each vulnerability was exploited and potential recommendations for remediation.
17. Reset the target VMs to their original state to repeat the lab or for other exercises.
18. Review and terminate any processes or sessions started on Kali Linux.
19. Always ensure that penetration testing is conducted within legal boundaries and with proper authorization.
20. Adhere to a code of ethics to respect privacy and avoid data damage.

This lab is a fundamental exercise for understanding the penetration testing process using tools commonly found in Kali Linux. It's crucial to conduct such labs in a controlled environment and to understand the potential impact of these tools and techniques in real-world scenarios. For advanced users, the lab can be expanded to include wireless network exploitation, web application attacks, and more sophisticated Red Team simulations.

Architects should leverage penetration testing toolkits judiciously based on scope and legal considerations to uncover security gaps without putting production systems at risk.

Automation and orchestration tools

Automation and orchestration tools are essential for improving efficiency, consistency, and reliability in cybersecurity operations. They allow for the coordination of complex workflows across multiple systems and processes. In this lab, we'll set up a basic security automation and orchestration platform using StackStorm, an open-source automation engine that connects your apps, services, and workflows.

Lab: Security automation with StackStorm

StackStorm is an open-source event-driven automation platform designed to simplify the automation of operational workflows and processes. It enables organizations to create and execute automated actions, known as **workflows**, in response to various events or triggers, such as monitoring alerts, system events, or user-defined conditions. StackStorm's architecture is built around three core components: sensors, rules, and actions. Sensors are responsible for monitoring and detecting events, rules define the logic and conditions for triggering actions, and actions are the automated tasks or operations performed in response to events. With its powerful rule engine, StackStorm allows for complex decision-making and conditional execution, making it a versatile solution for automating a wide range of IT operations, security operations, and business processes.

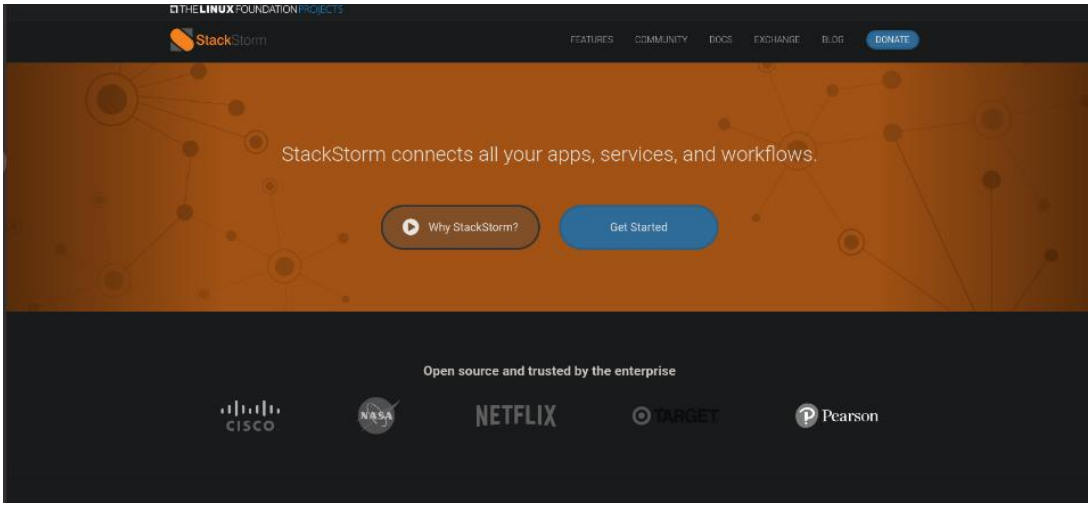


Figure 212 – StackStorm Website

You can access StackStorm here: https://stackstorm.com/.

The prerequisites include:

- A virtualization platform (e.g., VMware, VirtualBox) to host the virtual machines. StackStorm offers an OVA for a preinstalled VM but does require additional setup for use with Vagrant. You can see more at https://docs.stackstorm.com/install/vagrant.html.
- A virtual machine image with a compatible Linux distribution (e.g., Ubuntu 20.04 LTS). The screenshots will be based on Ubuntu 20.04.
- Internet access for downloading StackStorm and other necessary software components.
- Basic familiarity with Linux command-line operations and YAML syntax.

Let us look at the steps:

1. Set up a virtual machine within your virtualization platform and install a compatible Linux distribution.
2. Assign appropriate resources to the VM (e.g., at least 2 vCPUs, 4GB of RAM, and 20GB of disk space).
3. Access the VM via SSH or the console interface.
4. Follow the official documentation to install StackStorm. This typically includes running a script that installs StackStorm and its dependencies called the **Quickstart Script**:

```
 bash <(curl -sSL
https://stackstorm.com/packages/install.sh) --
user=st2admin --password=Ch@ngeMe
```

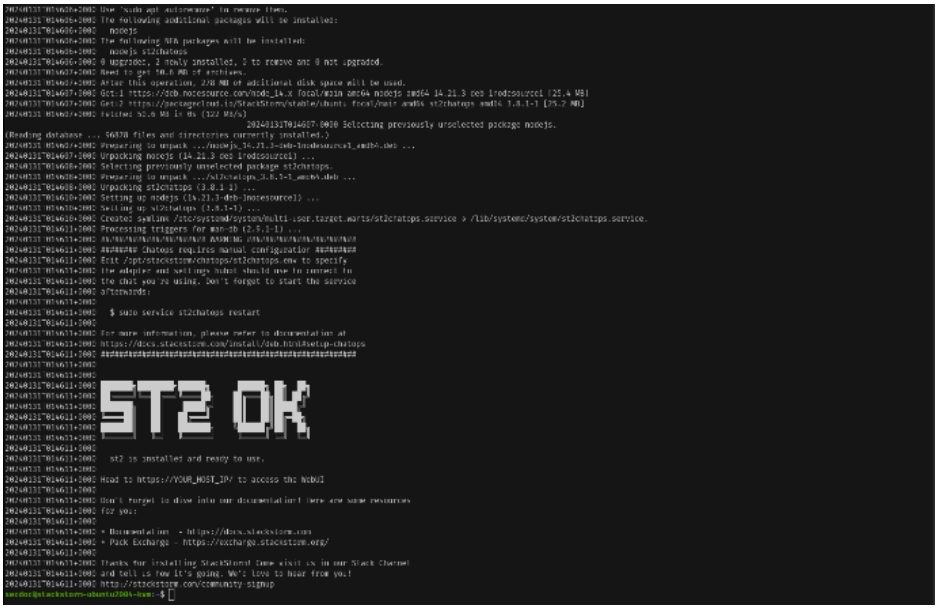


Figure 213 – Installation of StackStorm

5. Once the installation is complete, log into the StackStorm web interface using the credentials set during installation.

   Head to `https://YOUR_HOST_IP/` to access the WebUI:

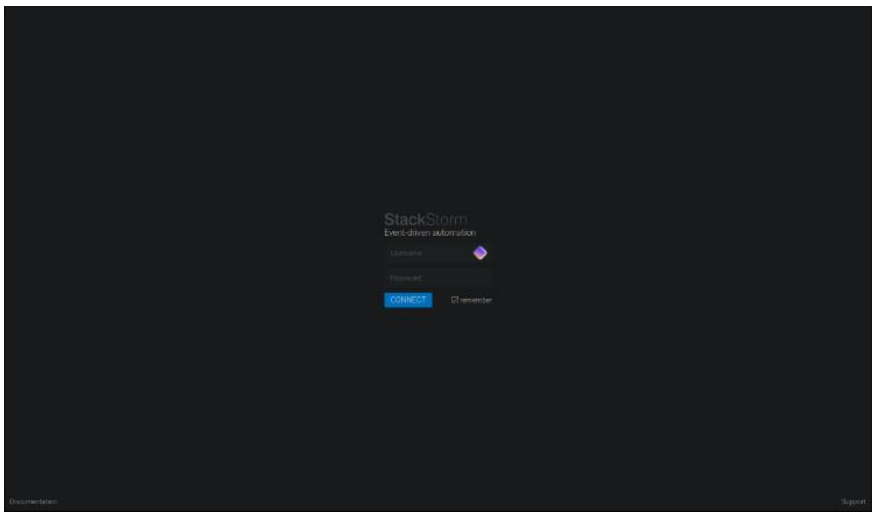


Figure 214 – StackStorm Login

6. Script sets the default password for the st2admin user to `Ch@ngeMe`.

7. Familiarize yourself with the interface and the basic concepts of StackStorm, such as packs, actions, rules, and workflows:

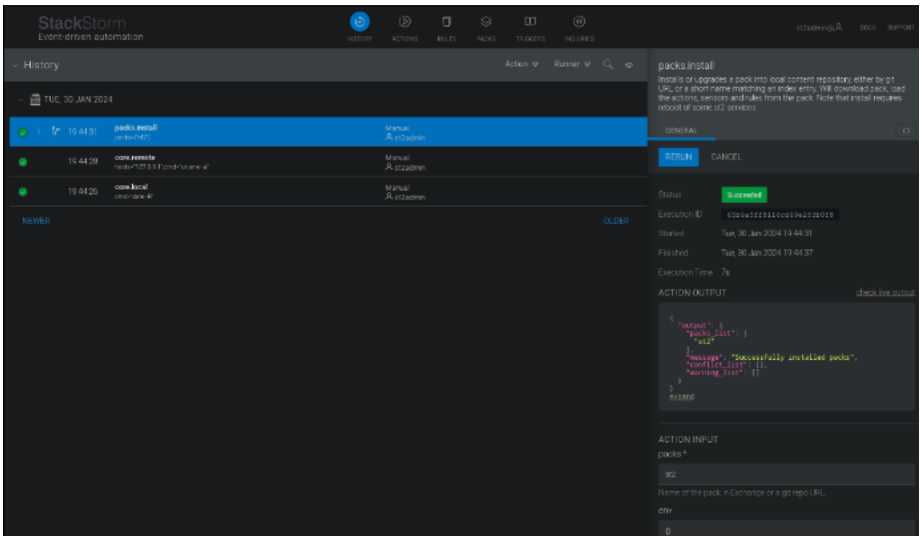
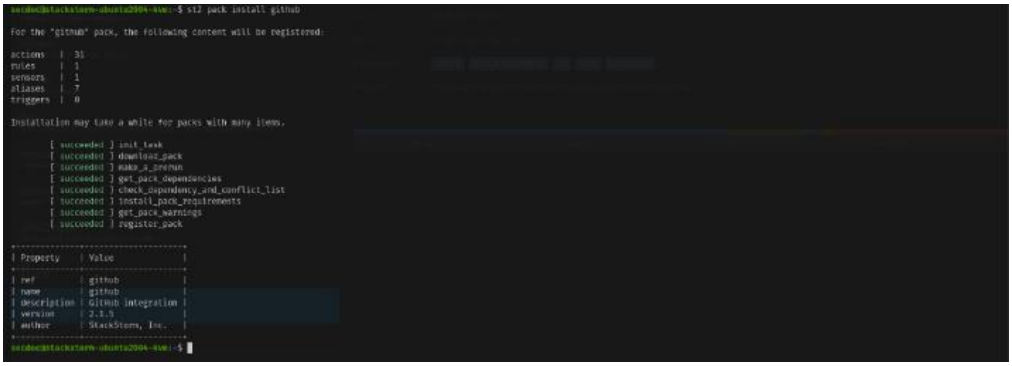


Figure 215 – StackStorm Dashboard

8. StackStorm uses packs to group actions, sensors, and rules. Install a pack for a common integration (e.g., GitHub, Slack) from the StackStorm Exchange to use as a base for automation This can be installed from the WebUI or through the cli:

```
st2 pack install github
```



(a) Install Github Module Pack

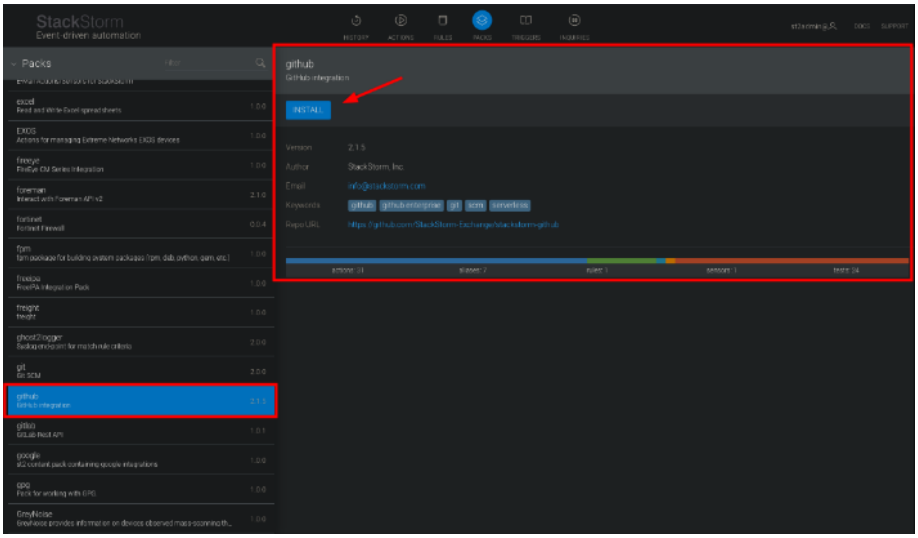


Figure 216 – Install Module

9. Test an action from the installed pack to verify StackStorm is functioning properly. For example, if you installed the GitHub pack, you can list your repositories with an action:

```
st2 action list --pack=github
```

Certainly! After installing StackStorm and integrating it with various packs (collections of pre-built actions), it's essential to test and verify that the installed

packs and their actions are functioning correctly. One way to do this is by running a specific action from the installed pack.

10. In your example, you've mentioned installing the GitHub pack, which provides actions related to GitHub operations. To test if this pack is functioning properly, you can use the `st2 action list` command to list all the available actions within the GitHub pack.

```
st2 action list --pack=github
```

This command will display a list of all the actions available in the GitHub pack. You can then select a specific action from the list to test it. For instance, if you want to list your GitHub repositories, you can run the following command:

```
st2 action run github.repos_list
```

This will execute the `github.repos_list` action, which should retrieve and display a list of your GitHub repositories.

If the action runs successfully and outputs the expected result (e.g., a list of your GitHub repositories), it confirms that StackStorm is functioning correctly, and the GitHub pack is properly installed and integrated.

You can follow a similar approach for testing actions from other installed packs. Simply list the available actions using `st2 action list --pack=<pack_name>`, select an action you want to test, and run it using `st2 action run <action_ref>`.

By testing different actions from various packs, you can ensure that StackStorm is functioning as expected and can automate various tasks and workflows across different systems and services:

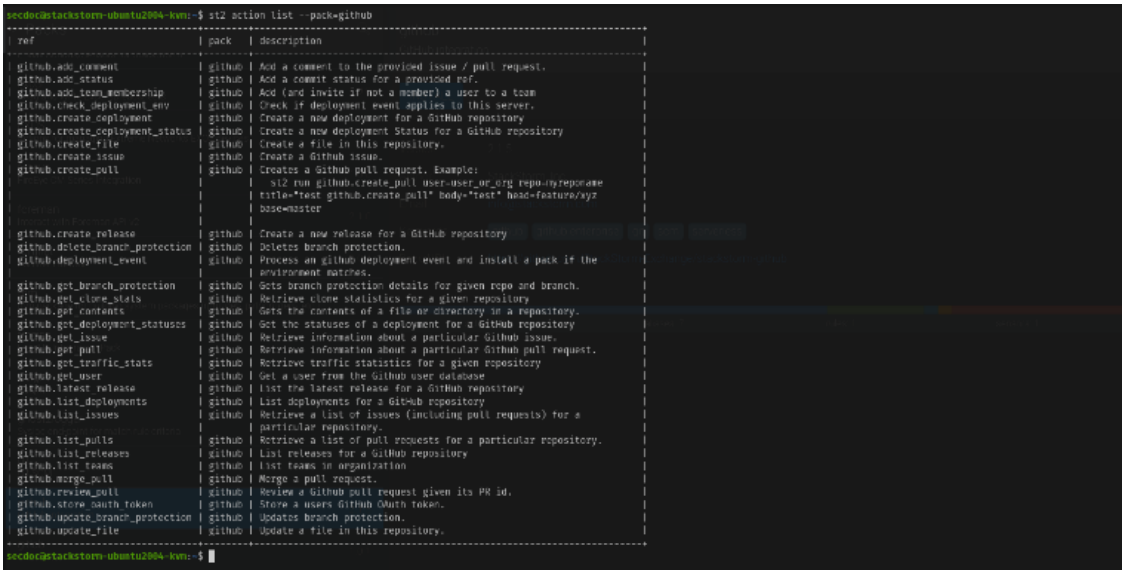


Figure 217 – Action List for Github Pack

More information can be found at https://docs.stackstorm.com/start.html.

11. Now, let us look at workflow design and sensors/triggers. Create a simple workflow using the Orquesta workflow engine built into StackStorm. Write a YAML file that defines the workflow. There are great examples provided by StackStorm to help you get started and learn the workflow process at https://docs.stackstorm.com/orquesta/start.html and sensors/triggers at https://docs.stackstorm.com/sensors.html.

12. Monitor the StackStorm execution history in the web interface to see the results of your workflow.

13. Check the logs for any errors or issues:

```
tail -f /var/log/st2/st2*.log
```

14. Upon completion of the lab, remove any sensitive information from the StackStorm system if necessary.

15. Document the workflow creation process, action execution steps, rule definitions, and any troubleshooting steps taken during the lab.

This lab introduces the basic concepts of security automation using StackStorm. It covers the installation and initial configuration of the automation tool, the use of packs for integration, the design and execution of a simple workflow, and the creation of automation rules. Upon completion, participants will have a foundational understanding of how to leverage automation for cybersecurity tasks.

SOAR platforms enable architects to connect disparate tools into a unified security ecosystem while leveraging automation for improved efficiency, consistency, and response times.

The taxonomy and associated examples are not exhaustive, as the domain of cybersecurity is dynamic and continuously evolving. New categories or tools may emerge as technologies and threats progress. The integration and interoperability of these tools are also critical, as cybersecurity architects must ensure that disparate security systems can work together seamlessly to provide a comprehensive defense-in-depth strategy.

Summary

Based on the provided list of labs and exercises, it appears to cover a comprehensive range of topics and practical implementations related to cybersecurity and information assurance. The labs and exercises cover various aspects of security, including threat modeling, intrusion detection and prevention, firewalls, SIEM solutions, antivirus software, endpoint detection and response, identity and access management, data encryption, vulnerability scanning, security configuration management, patch management, digital forensics, incident response, application security testing, cloud security, GRC tools, penetration testing, and security automation.

These hands-on labs and exercises provide valuable opportunities to gain practical experience and develop skills in various security tools and technologies. Participants can learn about threat modeling using tools like Microsoft Threat Modeling Tool and OWASP Threat Dragon, configure and deploy intrusion detection/prevention systems with Snort, set up and configure firewalls using OPNsense, implement SIEM solutions with Graylog, and deploy antivirus software like ClamAV.

Additionally, participants can explore endpoint detection and response with Wazuh, configure IAM solutions like Keycloak, implement data encryption using VeraCrypt, perform vulnerability scanning with OpenVAS, and practice security configuration management with Ansible. The labs also cover patch management with WSUS, digital forensics using The Sleuth Kit and Autopsy, incident response with Security Onion, static and dynamic application security testing with SonarQube and OWASP ZAP, respectively.

Furthermore, participants can gain experience in securing cloud environments with AWS, implementing GRC tools, conducting penetration testing with Kali Linux and Metasploit, and automating security tasks with StackStorm.

Overall, these labs and exercises provide a comprehensive and practical learning experience, equipping participants with the knowledge and skills necessary to address various cybersecurity challenges and implement effective security measures across different domains.