Network Security II project documentation

Kenneth Nnadi.
University of Oregon

CSC 534

June 1, 2024

Comprehensive Documentation for OpenVAS Vulnerability Management Using GVM API

Introduction

Vulnerability management is a crucial aspect of security that not only improves the lives of security professionals but also serves as an efficient way of managing threats and preventing them from turning into exploitable vulnerabilities. This project was born out of a desire to enhance security management and provide a streamlined method for controlling security using automation.

In this project, I focused on automating vulnerability scanning with OpenVAS Greenbone Vulnerability Management (GVM) API, using the GVM CLI and Unix socket connection. . Unix Socket Connection in Greenbone Vulnerability Management (GVM) leverages Unix domain sockets for local inter-process communication between the GVM components, such as the Greenbone Vulnerability Manager (gvmd) daemon and the Greenbone Security Assistant (GSA). This method enhances security by limiting communication to the local machine, avoiding exposure to network-based threats. Also, Unix domain sockets offer superior performance due to reduced overhead compared to network protocols. The socket path, typically specified as /var/run/gvmd.sock, is the key configuration parameter for establishing this connection. Unix Socket Connection is ideal for setups where all GVM components reside on the same host, ensuring efficient, secure, and fast communication. The project functions by automatically initiating a scan of a specific network at scheduled times, logging the scan status, parsing the results upon completion, retrieving the findings, and sending information about detected vulnerabilities based on their CVSS severity scores. This allows administrators to simply check their email for scan results and promptly address any high-severity vulnerabilities, thus preventing potential security breaches.

Scope of Project

Initially, the project aimed to cover a well-known vulnerability management SIEM tool called Tenable Nessus. However, the plan changed due to the high cost of the enterprise version, which led me to opt for the open-source SIEM tool, OpenVAS. While working with OpenVAS, I encountered challenges with the HTTPS API due to certificate issues that I couldn't resolve immediately because of time constraints. Consequently, I switched to using the Unix socket connection, which also employs the same API but in a different format, implemented via command line scripts. This project comprehensively covered vulnerability management within the constraints of the Unix socket connection, which only works on the system where OpenVAS is installed. Therefore, the project did not cover scanning external networks but was limited to the network or system on which it was deployed.

Installation

This documentation provides a comprehensive guide to installing and configuring OpenVAS (Open Vulnerability Assessment System) and automating vulnerability scans using a custom Bash script. The script leverages the Greenbone Vulnerability Management (GVM) CLI to perform scans, monitor progress, and send email reports.

OpenVAS Installation

Prerequisites

Ensure your system meets the following requirements:

- Operating System: A Linux distribution (e.g., Ubuntu, Debian, CentOS). I used kali for this experiment.
- System Resources: Sufficient CPU, memory, and disk space. It is pertinent to note that GVM has a very huge database which it uses to store the necessary signature used in the framework. It is expected of the user to have a very big RAM assigned to the machine used in conducting the vulnerability for resilience and reliability. I used 8GB RAM for this experiment. It also contribute to the fastness of the framework.
- Privileges: Root or sudo access.

Step-by-Step Installation

Update System Packages: Updating the system packages ensures that you have the latest security patches and software updates. This is a crucial step to avoid any compatibility issues during the installation process.

Commands:

sudo apt update

sudo apt upgrade

Install Required Dependencies: Dependencies are additional software packages that OpenVAS relies on to function correctly. These include tools for network scanning, XML parsing, and other utilities.

Commands:

sudo apt install sqlite3 python3-pip python3-paramiko python3-lxml python3-defusedxml python3-pyfiglet texlive-latex-extra xsltproc rpm alien nsis fakeroot wapiti

Install OpenVAS Packages: Installing the OpenVAS packages involves downloading and setting up the core components of the OpenVAS system, including the scanner, the manager, and the graphical interface.

Commands:

sudo apt install openvas && gvm

Configure OpenVAS: The configuration step involves initializing the OpenVAS database and setting up the environment. This step ensures that all components can communicate and function as intended. Run the setup command and follow the prompts to complete the configuration:

sudo gvm-setup

Check if the installation gvm dependencies was done/installed correctly:
Using the check-setup checks if the downloaded gvm is correctly installed. When this is run, it could be used as a debugging tool to download the extra dependencies that is needed to effectively utilize the gvm infrastructure. Commands:
sudo gvm-check-setup
Start OpenVAS Services: Starting the services makes OpenVAS operational. The services include the Greenbone Security Assistant (GSAD), the Greenbone Vulnerability Manager (GVMD), and the OpenVAS Scanner.
sudo systemctl start gvmd
sudo systemctl start gsad
sudo systemctl start ospd-openvas

Access OpenVAS Web Interface: The web interface allows you to interact with OpenVAS using a browser. It provides a user-friendly way to configure targets, scan tasks, and view reports.
Open a browser and navigate to:
https://localhost:9392
Log in with the default credentials (Username: admin, Password: admin) and change the password when prompted. (write something about it, especially changing the password.

Post-Installation Steps (pertinent)

1. Update Vulnerability Feeds: Regular updates to the vulnerability feeds ensure that OpenVAS has the latest information on known vulnerabilities. This step is crucial for accurate and comprehensive scanning.

   sudo greenbone-feed-sync --type NVT

   sudo greenbone-feed-sync --type SCAP

   sudo greenbone-feed-sync --type CERT

1. Harden Configuration: Hardening involves applying security best practices to reduce the risk of vulnerabilities within the OpenVAS setup. This can include enabling SSL/TLS, setting strong passwords, and configuring firewalls.

Configuring OpenVAS Manually

Creating a Target

To create a target in the OpenVAS web interface:

1. Go to Configuration → Targets.
2. Click New Target.
3. Fill in the Name, Hosts, and other necessary details.
4. Save the target.

Creating a Scan Config

1. Go to Configuration → Scan Configs.
2. Click New Scan Config.
3. Configure the scan settings according to your requirements.
4. Save the configuration.
5. Go to scan and hit the saved task to start scanning.

GVM Scan Automation Script

The provided Bash script automates several tasks, significantly streamlining the process of setting up and running vulnerability scans with OpenVAS. The automation ensures consistency and saves time, especially for regular scans.

Script Configuration

The script is configured through a series of variables that define the scan parameters, target details, and email settings. These variables need to be set according to the specific environment and requirements.

Script Execution

To execute the script, it must be run in a Bash-compatible shell. The script will perform a series of automated steps to authenticate with the GVM server, check for existing targets, create new scan tasks, monitor the scan's progress, and send an email with the scan results.

Detailed Feature Explanation

1. Authentication with the GVM Server

Authentication is the first step in the script. It involves providing the GVM server with valid credentials (username and password) to gain access to its functionalities. This step ensures that only authorized users can initiate scans and access the scan results. The script sets environment variables with the GVM user credentials, which are then used in subsequent GVM CLI commands to maintain a secure session.

export GVMD_PASSWORD=$GVM_PASSWORD

export GVMD_USER=$GVM_USER

2. Checking for Existing Targets and Creating New Ones if Necessary

Before starting a scan, the script checks if the specified target already exists in the GVM database. This involves querying the GVM server for targets and matching them against the desired target name. If the target exists, the script retrieves its ID for use in creating the scan task. If the target does not exist, the script creates a new target with the provided IP address and port list, ensuring that the scan will run against the correct system or network.

existing_target_id=$($GVM_CLI --gmp-username $GVM_USER --gmp-password $GVM_PASSWORD socket –xml "<get_targets/>" | xmllint --xpath "string(//target[name='$TARGET_NAME']/@id)" -)

3. Listing and Selecting Scan Configurations

The script lists all available scan configurations on the GVM server. Scan configurations define the types of vulnerability checks that will be performed. The script searches for the specified scan configuration by name. If the configuration is found, the script retrieves its ID. If the specified configuration is not available, the script can fall back to using a local scan configuration file. This ensures that the scan uses the correct set of vulnerability checks.

create_target_response=$($GVM_CLI --gmp-username $GVM_USER --gmp-password $GVM_PASSWORD socket –xml "<create_target><name>$TARGET_NAME</name><hosts>$TARGET_IP</hosts><port_list id=\"$PORT_LIST_ID\"/></create_target>")

4. Creating and Starting Scan Tasks

With the target and scan configuration IDs available, the script creates a new scan task. A scan task associates a target with a scan configuration and defines when and how the scan will run. Once the task is created, the script starts the scan. This involves sending a command to the GVM server to initiate the scan, which will then begin running according to the specified parameters.

create_task_response=$($GVM_CLI --gmp-username $GVM_USER --gmp-password $GVM_PASSWORD socket --xml "<create_task><name>$SCAN_NAME</name><config id=\"$policy_id\"/><target id=\"$target_id\"/></create_task>")

5. Monitoring Scan Progress

After starting the scan, the script enters a monitoring loop. It periodically checks the status of the scan task by querying the GVM server. The script reports the scan's progress and current status to the user. This monitoring continues until the scan is complete. This feature ensures that the user is informed about the scan's progress and can take action if there are any issues or delays.

6. Retrieving and Processing Scan Reports

Once the scan is complete, the script retrieves the scan report from the GVM server. The report contains detailed information about any vulnerabilities detected during the scan. The script processes the report, extracting relevant information such as vulnerability names, severity levels, and affected systems. This processed data is then saved to an output file for further analysis and record-keeping.

vulnerabilities=$(echo "$report" | xmllint --xpath '//result' -)

echo "$vulnerabilities" > "$OUTPUT_FILE"

7. Sending Email Notifications with Scan Summaries

To provide immediate feedback on the scan results, the script sends an email notification. The email includes a summary of the detected vulnerabilities, highlighting the most critical issues. The email body is constructed with a list of vulnerability names and their severities, and the full report is attached for detailed review. This feature ensures that stakeholders are promptly informed about potential security issues and can take action to mitigate them.

vulnerability_summary=$(echo "$report" | xmllint --xpath "//result" - | grep -oP '(?<=<name>).*?(?=</name>)|(?<=<severity>).*?(?=</severity>)' | paste -d ' ' - - | sed 's/^/Name: /;s/ /, Severity: /')

email_body="From: $EMAIL_FROM\nTo: $EMAIL_TO\nSubject: $EMAIL_SUBJECT\n\nThe scan is complete. Here are the severity levels and names of the found vulnerabilities:\n\n$vulnerability_summary\n\nThe full report is attached."

echo -e "$email_body" | msmtp -a default -t

Error Handling

The script includes error handling mechanisms to manage common issues such as failed target creation, missing scan configurations, and failed task starts. Error messages are printed to the console, and the script exits with an appropriate status code. This helps in identifying and resolving issues promptly.

Troubleshooting

Failed Installation: Verify that all dependencies are installed, network connectivity is available, and there are no conflicting packages.

Authentication Issues: Double-check GVM credentials and ensure they are correctly set in the script.

Script Errors: Review the script output for detailed error messages and ensure all required files and configurations are in place.

Database Issue with PostgreSQL Version: When installing OpenVAS, if the default PostgreSQL version (e.g., 15) is used instead of the required version (e.g., 16), you might encounter errors related to database compatibility. These errors can prevent OpenVAS from functioning correctly. To resolve this issue, you need to install the correct version of PostgreSQL and configure OpenVAS to use it. Here are the steps to do this:

Remove Existing PostgreSQL Version: First, ensure that any existing PostgreSQL installation is removed to avoid conflicts.

sudo apt-get remove --purge postgresql*

sudo apt-get autoremove

sudo apt-get autoclean

Add PostgreSQL Repository: Add the PostgreSQL repository that contains the desired version.

sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt/ $(lsb_release -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'

wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -

Install PostgreSQL Version 16: Install the required version of PostgreSQL.

sudo apt-get update

sudo apt-get install postgresql-16

Initialize the PostgreSQL Database: Initialize the database cluster for the new PostgreSQL version.

sudo pg_ctlcluster 16 main start

Verify PostgreSQL Installation: Ensure that PostgreSQL 16 is correctly installed and running by checking the version of the PostgreSQL.

psql --version

Configuring OpenVAS to Use PostgreSQL 16

Create OpenVAS Database and User: Create the necessary database and user for OpenVAS.

sudo -u postgres psql

CREATE USER gvm WITH PASSWORD 'your_password';

CREATE DATABASE gvmd OWNER gvm;

\q

Configure OpenVAS to Use the New Database: Update the OpenVAS configuration to point to the new PostgreSQL database.

sudo gvm-setup

sudo gvm-check-setup

For security reasons, it is important to change the default username and password used to access OpenVAS. This section explains how to change these credentials to user-chosen values. Also, at the installation of gvm, it automatically generates the password or the default password which is very large number. You need to modify it for simplicity.

Changing the default password and username

Access the GVM Command Line or any terminal: Use the gvmd command to change the password.

sudo gvmd --create-user=new_username --new-password=new_password

Verify Password Change: Log in to the OpenVAS web interface using the new password to ensure the change was successful.

By following this documentation, you can successfully install and configure OpenVAS and automate vulnerability scans using the provided Bash script. This setup helps simplify the process of identifying and addressing security vulnerabilities in your network.

Privacy concern in vulnerability management

Vulnerability management is a pertinent process for maintaining the security and integrity of an organization's IT infrastructure. It involves the continuous identification, assessment, and remediation of security vulnerabilities. However, this process can introduce significant privacy concerns that must be carefully managed. Ensuring the protection of sensitive data while conducting vulnerability assessments is vital for maintaining trust and compliance with legal requirements.

Key Privacy Concerns

Data Exposure: Vulnerability scans collect detailed information about system configurations, installed software, and potential vulnerabilities. This data can include sensitive information that, if not properly secured, could be accessed by unauthorized parties, leading to data breaches. It's essential to safeguard this information to prevent exploitation by malicious actors.

Data Retention: The storage of vulnerability scan reports and logs poses privacy risks. These documents often contain comprehensive details about an organization's network and systems. Retaining this data for extended periods increases the likelihood of exposure through accidental leaks or targeted attacks. Proper data retention policies are necessary to mitigate this risk.

Compliance with Privacy Regulations: Organizations must ensure that their vulnerability management practices comply with relevant privacy regulations, such as the General Data Protection Regulation (GDPR) in the European Union, the California Consumer Privacy Act (CCPA) in the United States, and others. Non-compliance can result in severe legal penalties and damage to the organization's reputation. Ensuring adherence to these regulations is critical.

Access Controls: Inadequate access controls to vulnerability management tools and data can lead to unauthorized access. It's crucial to implement strict access control measures to ensure that only authorized personnel can access sensitive vulnerability information. Regular audits of access logs help detect and respond to any unauthorized access attempts.

Impact on Personal Data: Vulnerability scans on systems that handle personal data, such as databases containing customer information, can potentially compromise the integrity and confidentiality of that data. Conducting scans in a manner that minimizes the risk of exposing personal data is essential to maintaining privacy.

Best Practices to Address Privacy Concerns

Data Encryption: Encrypt data both at rest and in transit to protect the sensitive information collected during vulnerability scans. Encryption ensures that even if data is intercepted or accessed without authorization, it cannot be read or used maliciously. Implementing strong encryption protocols is a fundamental step in safeguarding data.

Access Control and Auditing: Implement strict access control policies to restrict access to vulnerability management tools and data to authorized personnel only. Regularly audit access logs to detect and respond to any unauthorized access attempts. This helps maintain the integrity and confidentiality of the data.

Data Minimization and Anonymization: Collect only the data necessary for effective vulnerability management. Where possible, anonymize sensitive data to reduce the risk of exposure. For example, use pseudonyms or aggregate data to limit the amount of personally identifiable information (PII) collected. This practice helps minimize the potential impact of data breaches.

Regular Data Purging: Establish policies for the regular deletion of old vulnerability scan data and reports. Retain data only for as long as necessary to meet security and compliance requirements, and ensure that deletion processes are secure and irreversible. Regular data purging reduces the risk of data exposure over time.

Compliance with Legal Requirements: Stay informed about applicable privacy laws and regulations. Ensure that your vulnerability management practices align with these requirements, including obtaining necessary consents for data processing and implementing adequate data protection measures. Compliance helps avoid legal repercussions and maintains organizational integrity.

Training and Awareness: Educate employees and stakeholders about the importance of privacy in vulnerability management. Conduct regular training sessions to ensure that everyone involved understands the privacy risks and best practices for mitigating them. Building a culture of privacy awareness is crucial for effective data protection.

Incident Response Planning: Develop and maintain an incident response plan that includes procedures for addressing privacy breaches related to vulnerability management. This plan should outline steps for containing and mitigating the breach, notifying affected parties, and reporting to

regulatory authorities if required. A well-defined incident response plan enhances the organization's ability to handle privacy incidents effectively.

Future Work

In future work, I plan to extensively implement the HTTPS API vulnerability automation using GVM tools. Additionally, I will revisit and work on integrating Nessus until it functions seamlessly.

Appendix

Updating Feeds: Regularly update vulnerability feeds to ensure the latest security information is available. This is crucial for accurate scanning and reporting.

Starting Services: Ensure all OpenVAS services are running correctly to maintain system functionality and accessibility.

Script Configuration Variables

- GVM_CLI: Path to the GVM CLI tool.
- GVM_USER: Username for GVM authentication.
- GVM_PASSWORD: Password for GVM authentication.
- TARGET_NAME: Name of the target to scan.
- TARGET_IP: IP address of the target.
- SCAN_NAME: Name of the scan task.
- POLICY_NAME: Name of the scan policy.
- PORT_LIST_ID: ID of the port list.
- OUTPUT_FILE: File to save the scan results.
- SCAN_CONFIG_FILE: Path to a local scan configuration file.
- EMAIL_TO: Email address to send the report.
- EMAIL_FROM: Email address sending the report.
- EMAIL_SUBJECT: Subject of the email.

Script Execution

1. Prepare the Environment: Ensure all necessary dependencies are installed and configured.
2. Execute the Script: Run the script using a Bash-compatible shell:

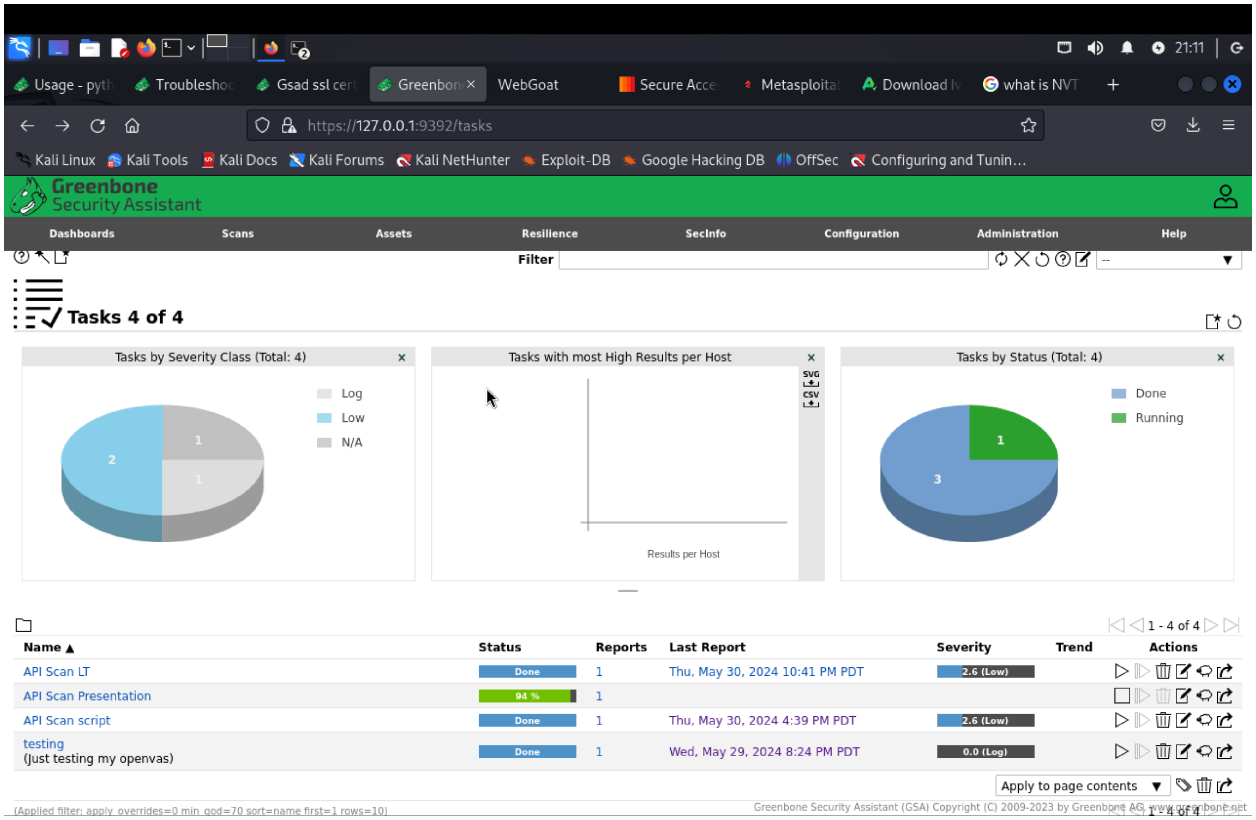
   ./scan.sh

1. Monitor the Script: The script will provide status updates as it progresses through each step.

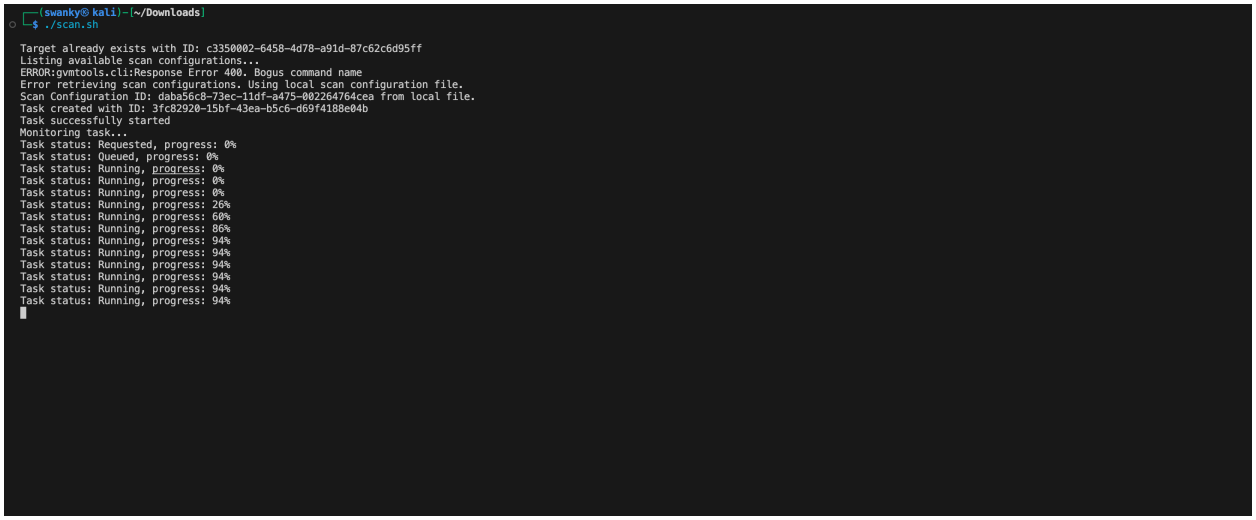
2. After the script is done running, it will extract the vulnerability and the severity of the vulnerability and send it to the recipient.

Pictures from implementation

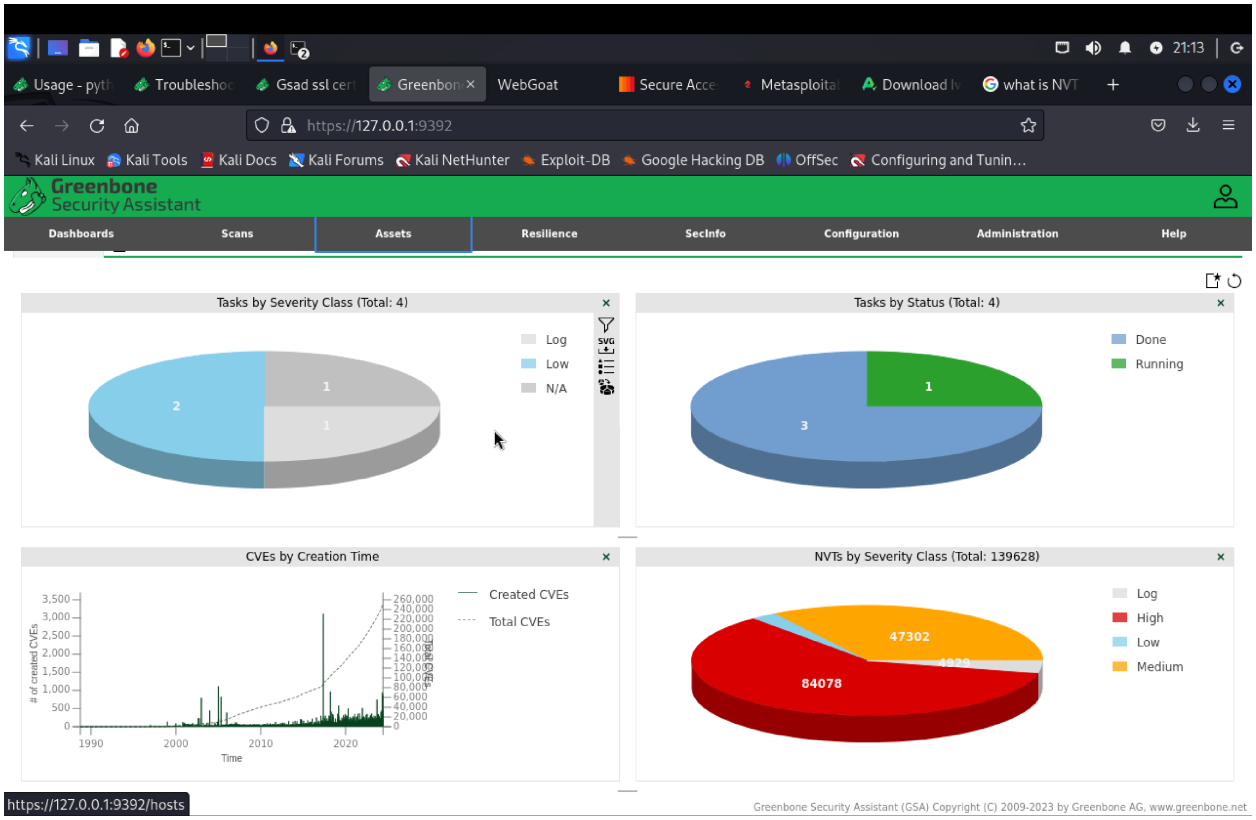
This is the Task scanned and statistics showing each of the task that have been scanned and the status, severity etc.



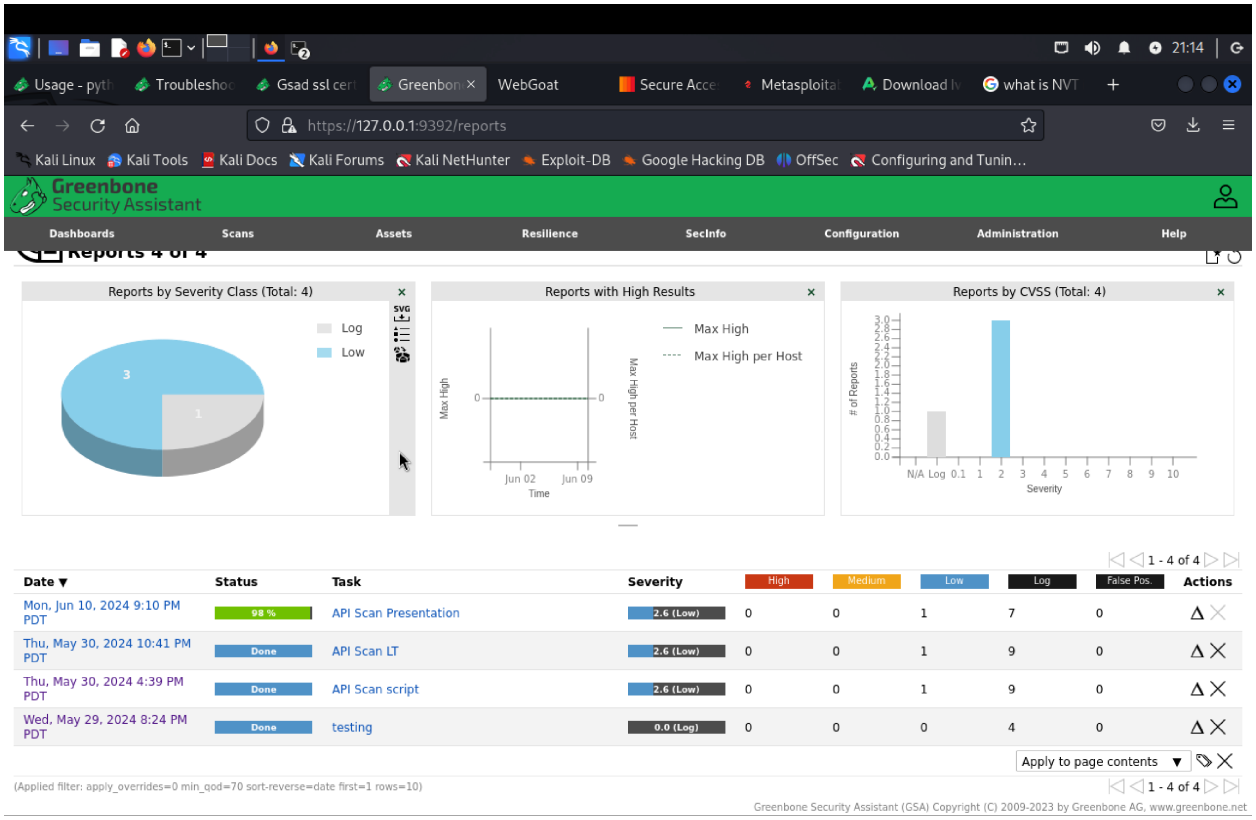
The script run on the termina. As you can see, the script update the status of the scan as it progresses. This could also be logged and used for troubleshooting incase anything happen during the scan process.



Dashboard showing the dashboard of the gvm and different statistics



Reports dashboard showing the reports of all the scan that have been done with time and the status.



This is when the scan is done, it then send an email notification to email address added in the script with the severity level and the vulnerability found in the scan.

The email showing the vulnerability and the severity level is below.