



**KOÇ UNIVERSITY**  
*College of Engineering*

**DEPARTMENT OF COMPUTER ENGINEERING**

**COMP 291/391 SUMMER PRACTICES I/II**

**Mehmet Murat Budak**

Internship Company and Department:  
Koç Üniversitesi Bilgi Teknolojileri Direktörlüğü / Bilgi Güvenliği  
19.06.2023 / 18.08.2023

**Supervisor (Name, Department, Phone, Fax, E-mail, etc.):**

Ertuğrul Doğan, Information Security, Work Phone: +90 212 338 1432,  
Mobile: +90 5438851861, E-mail: edogan@ku.edu.tr

**KOÇ UNIVERSITY**  
**COLLEGE OF ENGINEERING**  
**COMPUTER ENGINEERING DEPARTMENT**  
**COMP 291/391 SUMMER PRACTICES I/II**

1. Your write-up should adhere to those guidelines. Most importantly, all the write-up and figures and tables in your report (except the appendix) must belong to you, and must be in your own sentences. Any material that is copied from another source must be put in the appendix only, and properly referenced.

The report should be 20 to 30 pages including table of contents, main text (typed 1.5 line-spaced with 12pt fonts and 1” margins all around), figures with figure numbers and captions, and references. Appendixes are additional and there is no page limit for them. All figures, tables etc. must be also numbered appropriately.

2. Do not forget to have your supervisor sign all necessary documents, including your **Internship Evaluation form** (or Staj Degerlendirme Formu in Turkish) and **bottom of each page of the Work/Project Summary**, which is completed during (not after) your internship, as well as **the first page of your report** (where it is indicated that your supervisor should sign).

By submitting this report, you agree to all rules stated above, and all Koç University policies including the Koç University Student Code of Conduct as seen on:

<http://vpaa.ku.edu.tr/academic/student-code-of-conduct>

Failure to comply will result in severe penalty, and possibly forwarding to the disciplinary action committee.

## TABLE OF CONTENTS

### 1. INTRODUCTION

### 2. COMPANY DESCRIPTION

### 3. AUTOMATED VULNERABILITY ASSESSMENT AND REPORT GENERATION FOR KOÇ UNIVERSITY

#### 3.1. Problem Statement

#### 3.2. Team

#### 3.3. Tools and Techniques Used

#### 3.4. Detailed Explanation

#### 3.5. Results

### 4. CONCLUSIONS

### APPENDIX

### REFERENCES

# 1. Introduction

During my internship at Koç University's Information Security department, which spanned from June 19, 2023, to August 18, 2023, I was actively involved in cybersecurity operations, mainly focusing on ways to enhance the network security within school's local network, and ensuring a secure digital environment for both faculty and students. Network security these days is a very complex and an actively evolving concept, and it is absolutely essential for a big organization like Koç University to ensure that the network it uses has the necessary robustness, security, and capabilities to support the technological demands of its operations. My role focused on one of the vital aspects of Network Security, vulnerability detection and report generation.

## **Problem**

Considering the vast size of the network infrastructure of Koç University, manually analyzing each network node for vulnerabilities is not only impractical, but also very resource demanding. The Information Security department, being aware of the limited resources they have, has been using an automated vulnerability scanner called Nessus, which is a product of a proprietary software provider: Tenable Security. As of today, the yearly license for Nessus Expert costs around 200,000 TRY (or 5,300 USD) and since each computer system varies vastly in terms of the software, protocols, or Technologies used, accurately finding the vulnerabilities on all of the systems on the Koç Universities network is very difficult. On top of these disadvantages of Nessus, one of the main reasons there was a need for a different tool was the encouragement of open-source solutions by the Presidency of the Republic of Türkiye's Digital Transformation Office.

## 2. Company Description

Koç University, established in 1993 by the Vehbi Koç Foundation, is a distinguished and prominent educational institution located in Istanbul, Turkey. It is revered for its rigorous academic programs, research initiatives, and an environment fostering intellectual growth and development. The university operates across various sectors of education, including Engineering, Sciences, Social Sciences, and Humanities.

The Information Technology (IT) department at Koç University is a critical unit responsible for ensuring seamless technological operations across the campus. It aims to provide an extensive infrastructure, reliable services, and innovative solutions to support the academic and administrative functions of the university. The efficiency of the IT department is directly correlated with the success of the Koç University. The specific branch of the IT I worked at is Information Security, which manages one of the most crucial aspects of the entire IT infrastructure, without proper security measures in place, the IT department would be risking the digital security of more than 8,000 current students, and the operation of faculty labs and websites would be compromised.

The IT Department provides a wide range of services encompassing network administration and management, server administration, software development, and technological support for both academic and administrative purposes. Moreover, it facilitates a secure digital environment for research and data management, aiding the university in maintaining its high standards of academic integrity and excellence.

The department is not directly involved with research; however, it provides consulting and infrastructure for the University's faculty with their research activities and projects. For example, if the faculty is not sure as to how to collect and store data for their research activities, the IT provides them with necessary resources and consulting as to how to store and collect data adequately, making sure that any legal guidelines are followed, this is crucial because there are complex and comprehensive laws which must be followed in Turkey regarding data collection and processing (KVKK). This consultation service for faculty has become even more of a necessity considering how much research is being done in the field of Machine Learning and AI in which sensitive and vast amounts of data is collected.

The opportunity to intern at my own university's IT department was particularly appealing due to two main reasons. The first one being that I was always interested in Cyber Security, even before starting the university I would try to complete pentest boxes on sites such as tryhackme.com and hackthebox.com, and I already had experience with tools and frameworks such as nmap and Metasploit.

### **3. Automated Vulnerability Assessment and Report Generation for Koç University's Local Network**

#### *3.1. Problem Statement*

##### **Specific Problem**

Considering the vast size of the network infrastructure of Koç University, actively assessing the Network risks and finding effective solutions for mitigating security risks is a very important and challenging task. Only in 2022, over 25,000 new vulnerabilities were identified and documented, marking the highest annual figure up to that point, according to the National Security Agency [1]. It's anticipated that the trend of rising vulnerability counts will continue in 2023, with an average of more than 1,900 vulnerabilities expected to be published per month according to a report by insurance provider Coalition [2]. The sheer volume of devices connected to the network, along with the ever-increasing number of vulnerabilities being identified globally, requires an automated approach to vulnerability assessment.

##### **Purpose Of Solution**

The objective of implementing an automated vulnerability assessment system through OpenVAS is to facilitate up to date, continuous monitoring of the network's security profile. This agile system enables the Information Security team to identify, evaluate, and act upon emerging vulnerabilities, thereby gaining a strategic advantage over potential security threats. Additionally, the reporting capabilities of the system offer an immediate overview of network security, allowing the IT department to dynamically prioritize resource allocation and security countermeasures based on live data and up to date data. This approach not only ensures that the organization is aware of its current security profile but also enables active management of vulnerabilities in alignment with the latest threat intelligence. The system also does this without requiring a high level of technical skill from the person operating it, which is required in manual vulnerability assessment and penetration testing practices, thereby saving the IT resources.

### **Contribution to the Project**

As an intern, I have extensively researched how OpenVAS could enhance Koç University's IT department's capabilities in identifying and managing network vulnerabilities. After completing my research, I was tasked with installing OpenVAS on a special Virtual Machine which was hosted on the Schools Local network. The Virtual Machine was also configured to have special privileges and was whitelisted from the schools' firewalls, so that the automated tests could commence with maximal verbosity. The installation part was very troublesome and led me to leverage containerization techniques such as Docker. After the installation and integration process was complete, I ran multiple tests on actively vulnerable targets which were analyzed both manually and with other automated tools. By comparing the results gained from OpenVAS with manual testing results and other automated tools results I was able to gather practical and numerical data which helped me create reports and presentations to showcase how precise and accurate OpenVAS was at finding security issues.

### **Criteria for Success**

The criteria for success for this internship project were well defined and used a variety of metrics. Because I had the opportunity to gather lots of practical data from previous vulnerability assessments done by the Information Security team and other 3<sup>rd</sup> party penetration testers, I could directly compare the results OpenVAS provided and numerically assess how well OpenVAS could detect security issues. Specifically, the evaluation focused on several key performance indicators, the most emphasized one being the detection rate of various types of vulnerabilities -ranging from simple configuration issues and version vulnerabilities to complex SQL injections and information disclosure- across different operating environments and network conditions.

I analyzed how well OpenVAS could detect different kinds of vulnerabilities under different conditions and circumstances compared to other methods of security assessment. Additionally, the comparison extended beyond detection capabilities; it also incorporated qualitative measures like cost-efficiency, ease of deployment, user-friendliness, and scalability for future security requirements. Another layer of the evaluation was aimed at understanding the automation capabilities of OpenVAS. I analyzed how well the system could be integrated into existing workflows, and the extent to which it could streamline the vulnerability assessment process by automating repetitive tasks.

To ensure the integrity and reliability of my findings, I created reports and presentations which highlighted the OpenVAS results against the baseline data. This allowed for an objective analysis that was further enriched by subjective feedback from the Information Security team. Overall, the success of the project was measured through a comprehensive, multi-dimensional set of criteria that enabled an in-depth understanding of OpenVAS's capabilities, limitations, and potential for future utilization.

### **Previous Work**

Prior to this project, Koç University primarily relied on two methods for conducting vulnerability assessments. The first was the usage of Nessus, a proprietary tool by Tenable Security that is widely recognized and used in the industry. However, Nessus presented several challenges that were inconsistent with the needs of Koç University's IT team. First and foremost, Nessus exhibited inconsistent reliability in its results. This was most evident when the results of Nessus were compared to manual testing results reported by third parties. Nessus consistently fell short in detecting some critical vulnerabilities, thereby undermining its effectiveness in securing the network against evolving threats.



Secondly, the closed-source nature of Nessus stands in contrast to Koç University's focus on open-source solutions, a direction increasingly incentivized by the Presidency of the Republic of Türkiye's Digital Transformation Office. The closed-source nature of Nessus limited the customization and adaptability of the tool, adding another layer of complexity and hindrance to its integration within the existing IT framework. Thirdly, Nessus did not provide actionable insights for mitigating identified vulnerabilities, leaving the team to seek supplemental resources to bridge this informational gap, considering how many network nodes must be secured in the vast University network, this posed to be a very big problem for the Information Security team, consuming a lot of work hours.

Financially, Nessus also proved to be quite expensive, the base cost for an annual license is approximately 5,300 USD, which becomes exponentially more expensive when adding advanced features like active monitoring and alarm systems. This high-cost structure was yet another factor that led the Information Security team to consider alternative solutions, such as OpenVAS, that could provide more value for the investment.

The second method utilized by Koç University for vulnerability assessment is broad scope penetration tests by 3rd party cybersecurity firms, which I had the chance to observe and participate in. While this method provided the most comprehensive analysis of the security profile of the network, it was also very expensive and inflexible. Usually, these tests were conducted at most once per year, and in a world in which there are approximately 1900 vulnerabilities discovered each month waiting one year to fortify a network as crucial as Koç University's is a very risky approach. The second main problem with these tests is that they are very expensive to conduct and therefore it is difficult to communicate if security issues are checked or not, because the information security team has to wait one year to see if the security issues are actually fixed or not.

### 3.2. *Team*

In the context of this project, I carried out the vulnerability assessment and OpenVAS implementation largely independently, while operating under the general oversight of two other members from Koç University's Information Security Department. Specifically, I operated under the mentorship of Oğuz Özkan, the leading Security Operations Specialist, and Ertuğrul Doğan, the Staff IT directorate. While I undertook the primary tasks of implementation and troubleshooting, I engaged in regular consultations with these individuals to incorporate their feedback into the project's execution. In addition, they provided recommendations on which evaluation metrics I should prioritize and guided me in choosing appropriate network segments for data collection.

Additionally, on issues of system compatibility and software installation challenges, I coordinated with Hümayun Haşimler, the Senior System and Cloud Administrator. Although the responsibility for implementing the OpenVAS configuration and troubleshooting remained solely with me, Haşimler's expertise was very helpful in resolving issues related to Virtual Machines, thereby ensuring the smooth progression of the project.

The decision-making process was consultative in nature, with meetings primarily focused on discussing weekly activities, planning future initiatives, and evaluating the state of network security at the university. These consultative interactions aided in refining the project's focus and ensuring its alignment with the broader objectives of the Information Security Department.

### 3.3. *Tools and Techniques Used*

The primary software package employed was OpenVAS, installed on a Kali Linux Virtual Machine (VM). OpenVAS was chosen for its robust vulnerability assessment capabilities and its integration with Greenbone's suite of security tools. The software was essential for scanning and identifying network vulnerabilities in real-time. Given that the VM resides on the local network, OpenVAS was a fitting choice for an integrated approach to vulnerability assessment within Koç University's infrastructure. The platform's advantages are its rich vulnerability database and high customization potential. However, OpenVAS does demand a fair amount of manual configuration, and its setup proved to be very complicated.

Docker played a critical role in overcoming installation issues with OpenVAS, which I initially encountered and was unable to resolve. This containerization platform simplified the deployment of OpenVAS, mitigating various installation challenges. The primary advantage of using Docker was its capability to package an application and its dependencies into a single container, thereby making the software more portable and less dependent on the underlying operating system. Most importantly, this containerization technique eliminates the variability in the host operating system, ensuring that if the program runs on a single person's Docker container, it will run in all the other Docker containers across the world, this vast compatibility feature of Docker, enabled me to overcome the issues I have faced with `build from source` technique I was initially trying to implement. A downside to using Docker is the additional system resources it consumes, which can occasionally contribute to performance overhead.

For initial testing and troubleshooting of OpenVAS, I selected Arch Linux and Debian Linux due to their unique package management capabilities. Arch Linux, with its rich AUR (Arch User Repository), provided access to a vast array of packages, facilitating the exploration and implementation of additional tools as required. Debian Linux served as a direct testbed for compatibility, as its package ecosystem aligns closely with that of Kali Linux, which is Debian-based. This selection ensured that any packages operational on Debian would be translatable to the Kali VM environment. The transition to the Kali VM was driven by its more focused and specialized environment for cybersecurity tasks. A significant advantage of Kali lies in its extensive range of pre-installed cybersecurity tools, which provided the Information Security team with a robust platform for conducting exhaustive attack and defense simulations on the

local KU network. The utilization of Kali meant that the Information Security team had an “attack box” at the heart of the network with maximal network privileges, which enabled the attack and defense simulation tests to be conducted with maximal verbosity making sure that the chances of finding a critical vulnerability were as high as possible.

The Kali Linux Virtual Machine (VM), provided by IT’s System and Cloud Administration team and hosted within the school's local network, served as the hardware platform for this project. The VM offered an isolated and secure environment for OpenVAS implementation, thus mitigating risks associated with vulnerability assessment tasks. The advantage of using a VM lies in its capacity for resource allocation and isolation, ensuring that the project would not interfere with other operations within the same server. Also, one other major benefit of having a VM in schools’ network is that once in the KU network anyone with valid credentials could reach the OpenVAS interface, meaning that for any IT employee it was very easy to access OpenVAS be it via its webserver or SSH, and for cases in which a person wants to reach the machine from outside the network, simply using the VPN provided ensured this functionality.

No major programming languages were used in this project, but there were a few scripts created in Bash that improved the user friendliness, the language was used to create a few simple scripts that make the terminal interface of the Virtual Machine OpenVAS is installed on easier to interact and routinely update OpenVAS containers. The scripts interact with the Docker API and provide basic functionality which enables to start, stop, update, and view the live logs of the OpenVAS service daemon. My main concern was that since there are very few people in the IT team which have experience with Linux based systems, it would be hard for them to interact with the Docker API directly, therefore I added easy to use scripts for future user interaction. I also made sure that whenever a user is logged in to the Kali VM via SSH, the list of available commands gets automatically printed with relevant information enabling the user to easily update the system or view the logs. Bash scripting had to be used in this context because it is the conventional language to interact with terminal based API’s such as Docker and all of the services of OpenVAS. The main advantage bash scripting provided was that it was very easy to use in Linux, since bash scripting is universally used in all Linux based systems, and that it didn’t require any boilerplate code which made the bash scripts both easy to implement and easy to document. The scripts can be used with simple commands such as ``gvm-start``, ``gvm-stop``, ``gvm-update``, and ``gvm-logs``. There were also several small packages

mainly aimed at improving user-friendliness of the system. Some examples are: fish-shell which made the bash terminal provided by Kali easier to use due to its syntax highlighting, and auto-completion features. Nala, which is an apt wrapper, apt being the main package manager of Debian based systems. Nala made installation procedures easier to understand with its extensive CLI improvements and dependency control features.

### 3.4. *Detailed Explanation*

This section is split into two parts; the first one explains in detail how the installation procedures were conducted in a step-by-step manner, and the second part explains how the installed packages were used.

#### **Virtual Machine Setup**

The VM setup was done by the IT's Cloud and System Administration team, after providing them the .ISO image of the Kali Linux XL distribution, they installed it in a standard VM with average resources allocated to it. After the installation I was sent the required credentials to connect to the VM via SSH, the VM was also closed to access from outside of the KU network. I faced an issue with the storage aspect of the VM while using docker but was able to resolve the problem by communicating with Hümayun Haşimler. He solved the problem by increasing the disk size of the VM.

#### **Initial Workspace Setup**

After connecting to the VM with the provided credentials I began to install some of the basic packages that improved user experience, keep in mind that these packages mostly depend on the preference of the user since there are also many other alternatives. Since Kali is a Debian based Linux distribution it uses 'apt' as its main package manager, this package manager is one of the oldest of its kind, with its CLI interface being quite outdated. Due to these reasons, I have decided to install 'nala' a relatively new 'apt' wrapper, which in its cores uses 'apt' commands but improves the CLI experience vastly, 'nala' also has some dependency control features which are also quite useful to manage a distributions' packages. Nala can be installed with the following command: 'sudo apt install nala', detailed information can be found in the GitHub page of nala and also its own wiki. After configuring nala, to improve the terminal usability and efficiency, several additional packages were integrated:

**Fish Shell:** Adopted the Fish shell (fish-terminal) for its user-friendly features, including advanced autocompletion and extensive syntax highlighting, which elevated the command-line interface experience, and aided me in being more efficient.

**Exa:** Replaced the traditional 'ls' command with 'exa', a modern replacement that offers enhanced file listing capabilities and integrates seamlessly with the Fish shell environment.

**Bat:** Replaced the classic 'cat' command with 'bat', which on top of printing file contents also provides syntax highlighting, making it significantly easier to read and review files directly from the terminal.

**Command Abbreviations:** Established abbreviations for the newly adopted commands, thus streamlining their execution and minimizing the keystroke requirement for frequent operations.

Additionally, for easier access and best practice, I have added my personal SSH key to the VM.

These enhancements were important in crafting a more productive and less error-prone environment, thereby aligning with the objective to simplify the system's operability for users of varying proficiency levels in Linux systems.

### **Installation Procedure of OpenVAS**

The installation of OpenVAS began on a local Kali machine, taking advantage of the distribution's prebuilt repositories for OpenVAS. Unfortunately, this initial approach encountered problems, the installation process failed to activate the necessary daemons, rendering the OpenVAS suite inoperative. This was a crucial learning point, illustrating the necessity for alternative methods when prebuilt solutions fail.

Transitioning from Kali, I turned to my Arch Linux system. Arch is renowned for its versatility and a comprehensive AUR (Arch User Repository) repository, making it a prime candidate for custom installations, and a good test bed. I attempted to compile OpenVAS from the source. This technique proved to be more challenging than expected. Despite successful compilation and service initialization, I faced a persistent socket communication error between services. As a result, while the OpenVAS web interface was accessible, the scanner component failed to function, which was the most crucial component of the whole OpenVAS suite.

In the pursuit of a functioning OpenVAS installation, Docker emerged as the most viable solution. Docker containers encapsulate applications in a self-sufficient environment, similar to virtual machines, ensuring consistency across different systems. I discovered an official OpenVAS Docker container and proceeded with the installation on my Arch system. The Dockerized version of OpenVAS circumvented the previous complications by providing a pre-configured environment where all components could interact seamlessly, I later discovered that docker containers are apparently more well maintained than the source code of the project itself.

The installation process involved pulling the OpenVAS Docker container and executing it with the appropriate configurations. The following steps were undertaken:

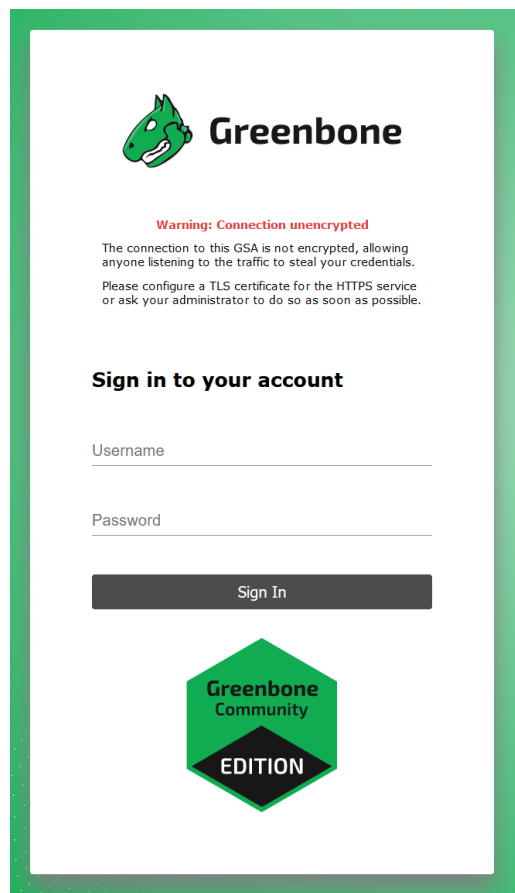
**Installing Dependencies:** OpenVAS containers and docker has some dependencies which don't always come preinstalled with Linux distributions. These dependencies are `ca-certificates`, `curl`, and `gnupg`, which can be installed with the package manager of the preferred distribution. In Arch the recommended way would be to use `pacman` in the following way: `sudo pacman -S ca-certificates, curl, gnupg, docker-ce, docker-ce-cli, containerd.io, docker-compose-plugin`. In Debian based systems, the preferred command is: `sudo apt install ca-certificates curl gnupg docker-ce docker-ce-cli containerd.io docker-compose-plugin`. The main difference being the package manager used and their corresponding syntax.

**Installing the Docker Container of OpenVAS:** The steps followed for the docker installation were followed from the Greenbone (OpenVAS) Community Editions official Documentation interchanging the commands for Debian based systems to utilize pacman, as exemplified in Installing Dependencies part. The documentation can be found at the following link: [\[Installing Docker\]](#). Through this experience, the adaptability of Docker and the importance of containerization in overcoming installation issues were highlighted. The ability to replicate a working setup across different systems without the intricacies of dependency management or service communication issues is a testament to Docker's utility in simplifying complex software deployments.



## Configuration of OpenVAS

After starting the OpenVAS daemon suite with the following custom command: ``gvm-start``, I configured the credential information and updated the password information of the administrator user, the information on how to do this in detail can be found in the following link: [[Setting up an Admin User](#)]. After all of the daemons start, the user should be able to reach the web interface at the following link: (host machines IP address):9392, port 9392 is the default port for hosting the OpenVAS web interface service.




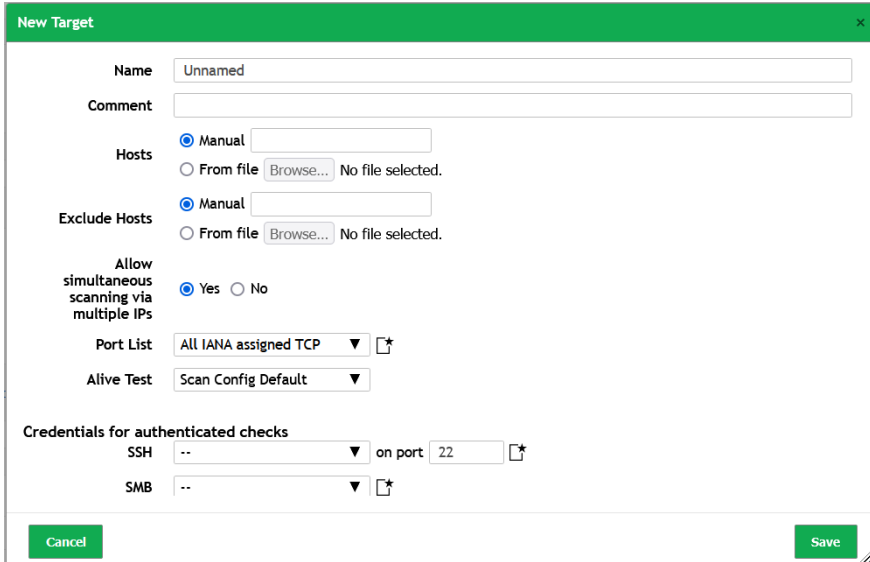
**Image 1:** The login screen of the OpenVAS web interface.

**Ensuring Scan Feature Is Working in OpenVAS:** In order to initiate scans and view their subsequent results, there are a few steps the user has to follow. Firstly, it is important to note that there is a very lengthy process which needs to complete first in order for the scanning feature to work successfully. This is the process of pulling “feed” data. Usually, this process begins right after the initialization of the OpenVAS daemons but there is no explicit warning that the process has begun or is hindering the initialization of the scans, to check for this the user should open the live logging feed of the OpenVAS. This can be achieved by running the

following command: ``docker compose -f $DOWNLOAD_DIR/docker-compose.yml -p greenbone-community-edition logs -f``. Where ``$DOWNLOAD_DIR`` is the download directory where the docker-compose.yml file is located, as it should have been downloaded by the user in “Installing the Docker Container of OpenVAS” section. Once the log interface is open the user can check if there are any indications that the process of pulling the feed is still ongoing, this part is explicitly written in the logging screen so the user should be able to easily understand whether the process has finished or not. The process can take from a few hours to a day depending on the internet connection and the location of the user. After the process is finished, we should be left with a fully working installation of OpenVAS or more specifically Greenbone OpenVAS Community Edition (GVM) Container.

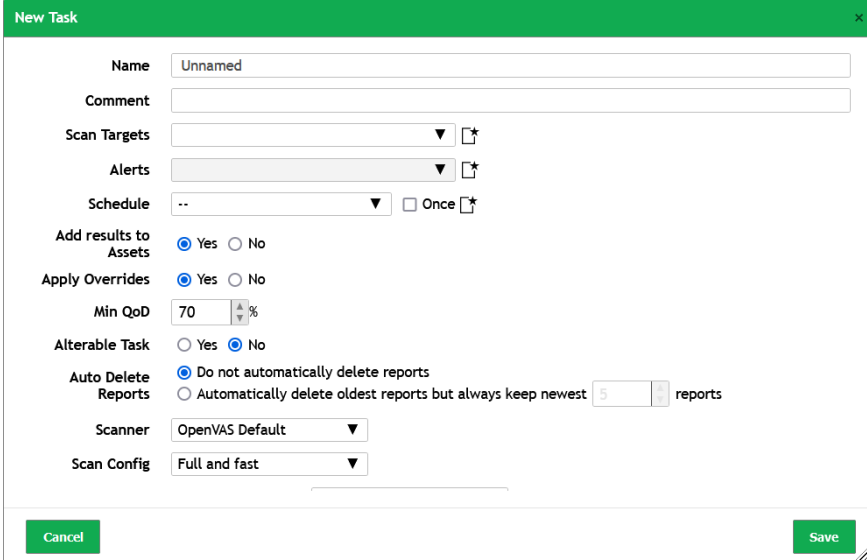
### Traversing Through the Web UI and Initiating Scans:

The Web UI of OpenVAS may seem a bit unintuitive and complicated at first, but the learning curve is actually quite shallow, and becomes easy to use quickly. In order to start the scan the user has to hover over the `configuration` panel and choose `Targets` from the list that pops up. After the `Targets` page is loaded the user has to click on the new target button at the upper left corner of the UI, which looks like the following image: . After clicking on the said button, the user can choose his desired targets and name them accordingly.



**Image 2:** The Target Panel of OpenVAS

The targets can be specified manually or from a file that contains the targets' IP addresses or their domain names. After the targets are specified and saved, the user now must initiate the scan. To do this, the user has to hover over the `Scans` section and click `Tasks` from the menu that pops up. In this page, to create a new task the button at the upper left corner should be clicked.

The image shows a 'New Task' dialog box with a green header bar. It contains several form fields: 'Name' (text input, value 'Unnamed'), 'Comment' (text input), 'Scan Targets' (dropdown menu with a star icon), 'Alerts' (dropdown menu with a star icon), 'Schedule' (dropdown menu with a star icon and a checkbox for 'Once'), 'Add results to Assets' (radio buttons for 'Yes' and 'No', 'Yes' is selected), 'Apply Overrides' (radio buttons for 'Yes' and 'No', 'Yes' is selected), 'Min QoD' (spin box, value '70', followed by a '%' sign), 'Alterable Task' (radio buttons for 'Yes' and 'No', 'No' is selected), 'Auto Delete Reports' (radio buttons for 'Do not automatically delete reports' and 'Automatically delete oldest reports but always keep newest 5 reports', the first is selected), 'Scanner' (dropdown menu, value 'OpenVAS Default'), and 'Scan Config' (dropdown menu, value 'Full and fast'). At the bottom, there are 'Cancel' and 'Save' buttons.

**Image 3:** The New Task Interface of OpenVAS

The previously saved targets can be chosen from the `Scan Targets` selection and the desired scan configuration can be chosen from the `Scan Config` selection. Upon saving the configured `New Task`, the main `gvmd` daemon will communicate with the OpenVAS scanner daemon and the scan process should begin, all of these steps can be explicitly seen in the logs section.

### Viewing and Inspecting Reports

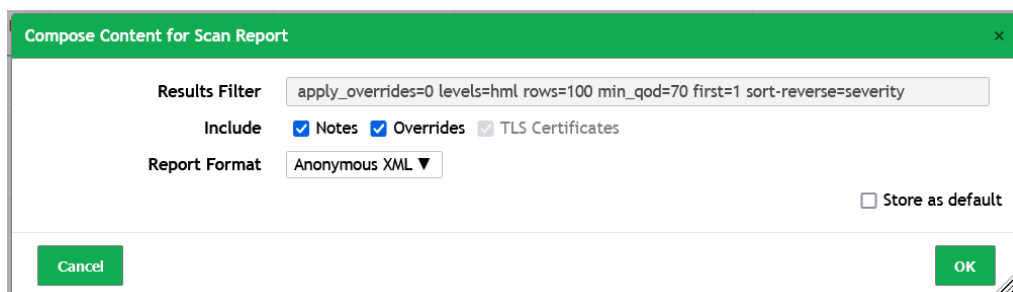
Once a scan concludes, which the user can verify from the 'Logs' and 'Dashboard' sections of the web interface, a comprehensive report on identified vulnerabilities will be available in the 'Dashboard' section. To examine this report, users should:

1. Navigate to the 'Reports' section.
2. Locate the desired report, identifiable by its timestamp.
3. Select the report to open its detailed panel.

Within the report, users will find nine different sections each offering specific information. Of particular importance is the 'Results' section. Here, users can review all discovered vulnerabilities, including:

- The name and a textual description of each vulnerability.
- The proposed type of solution for remediation.
- The severity level, categorized using the standard 'CVE' (Common Vulnerabilities and Exposures) notation.
- The Quality of Detection (QoD) rate, which reflects OpenVAS's confidence in the accuracy of each identified vulnerability, indicating the likelihood of it being a true positive.
- Host details for where the vulnerability was detected, including IP address, hostname, and the specific port involved.
- The timestamp noting when each vulnerability was found.

The reports can also be viewed in different formats after downloading them through the report's specific web page. The supported formats are: Anonymous XML, CSV, ITG, PDF, TXT, XML.



**Image 4:** UI for downloading the report

### 3.5. Results

Upon completion of the project, the results were documented and subsequently presented to the Group Leader of the Information Security team, Ertuğrul Doğan. The presentation primarily showcased the prowess of OpenVAS in identifying vulnerabilities that Nessus had overlooked.

One of the major highlights was that, with a refined scanner configuration, OpenVAS demonstrated vastly superior performance over Nessus. This superiority was not only manifested in the detection of distinct vulnerabilities but also in successfully identifying every single vulnerability that Nessus was able to detect. Such comprehensive vulnerability detection essentially rendered Nessus redundant in the context of this evaluation.

**Vulnerability Detection Comparison:** OpenVAS identified a series of very critical vulnerabilities in some of the machines used for hosting faculty websites, many of which were not detected by Nessus. Conversely, there were no vulnerabilities detected by Nessus that were missed by OpenVAS, underlining the thoroughness of OpenVAS's scanning capabilities. I was also able to find some other tests conducted by 3<sup>rd</sup> parties on purposefully vulnerable series of servers.

Category	Greenbone	Nessus
Operating System End of Life	High	Critical
Debian Vulnerabilities	High	Critical
Drupal Vulnerabilities	High	Critical
MySQL Vulnerabilities		Critical
Webmin Vulnerabilities	High	
Missing Function Level Access Control		
Apache Vulnerabilities		
Adminer Vulnerabilities	High	
SSH Key Algorithm	Medium	
Plaintext Authentication	Medium	Medium
Config File Accessible	Medium	Medium
Password Autocompletion		Medium
TLS Versions	Medium	Medium
TLS Key Algorithm	Medium	
SSL Certificate Trust	Log	Medium

**Table 1:** Vulnerabilities detected by Greenbone (OpenVAS) and Nessus on three different server configurations, without the customized scanner config for OpenVAS.

**Price and Future Flexibility Comparison:** One of the significant advantages of employing OpenVAS over Nessus is its cost-effectiveness. OpenVAS is available at no monetary expense, making it a financially viable option for organizations, especially when the licensing costs associated with Nessus seem to be unaffordable.

Beyond the evident cost savings, OpenVAS offers an array of features that enhance its value proposition:

1. **Continuous Scheduled Scans:** OpenVAS has the ability to set up recurring scans. This ensures that the system is consistently monitored for potential vulnerabilities, enhancing the security posture over time.
2. **Automatic Alert Generation:** OpenVAS features a robust alert system. Whenever a critical vulnerability is identified, the system has the ability to automatically generate and dispatch alerts. This can be in the form of emails or Track-It tickets, directed at the respective machine owners, urging them to address the detected issues.
3. **Observer Report Viewing:** This feature facilitates a collaborative approach towards system security. Other IT members can utilize this feature to analyze the vulnerability situation of their respective machines. Such transparency not only ensures accountability but also fosters a proactive security culture.
4. **Advanced Report Features:** OpenVAS stands out with its capability to generate detailed reports in CSV format. This format, being easily parsable, enables the building of advanced analytics and insights on top of the generated reports. Such functionalities empower the IT team with data-driven decision-making capabilities and enable a more nuanced understanding of the organization's security health.
5. **High Customizability:** A standout feature of OpenVAS is its open-source nature. Being open-source means that its source code is publicly accessible and can be modified, tailored, and improved upon by anyone with the necessary expertise. This provides a level of flexibility that proprietary tools like Nessus cannot match. Organizations can delve deep into the OpenVAS codebase and adjust it to cater to their specific requirements, be it integrating with other tools, enhancing existing

functionalities, or even building entirely new features from scratch. This capability ensures that OpenVAS remains relevant and adaptable to an organization's evolving needs. Moreover, the open-source community surrounding OpenVAS actively contributes to its improvement, meaning that new features, security patches, and performance enhancements are continually being added. This collaborative approach ensures that OpenVAS remains at the forefront of vulnerability scanning technology. By choosing OpenVAS, organizations are not just opting for a tool but are investing in a platform that offers boundless customizability, ensuring its relevance and efficacy in the ever-evolving cybersecurity landscape.

Considering the above, while both tools serve the primary purpose of vulnerability scanning, OpenVAS's cost-free nature combined with its advanced feature set renders it a more adaptable and forward-looking choice for organizations intent on robust and dynamic vulnerability management.

**Qualitative metrics:**

Feedback from the Group Leader: The Group Leader's response was notably positive, emphasizing the potential cost and efficiency benefits of transitioning to OpenVAS, given its adeptness in vulnerability detection. Notably, the Group Leader decided to showcase my presentation to Serkan Çil the ICO of the KU IT team.

In conclusion, OpenVAS not only met the set expectations but also showcased its potential as a more efficient and comprehensive vulnerability detection tool, surpassing Nessus in direct comparisons.

## 4. Conclusions

### **Education's Role in the Internship:**

Throughout my internship, I found that my academic training provided a solid foundation that equipped me to tackle real-world challenges. Most notably, I can wholeheartedly thank my COMP100 teacher Fatma Güney for trashing Microsoft Windows for over half the lecture in our first class and telling us to use Linux instead, I followed her advice and became, what I think, a very proficient Linux user. My Linux skills were the core of my whole internship and without them I wouldn't be able to tackle the issues I have faced with the installation of OpenVAS. The techniques I used, and my knowledge of Linux commands was heavily reinforced in various courses and COMP100 labs.

I learned quite a lot in the internship, I would say that the biggest area of improvement for me was handling corporate relationships and actually understanding how a corporation and the Teams in that corporation operate. Pitching my `product` (OpenVAS) to the IT team and having the chance to actually implement it was a very rewarding experience, which boosted my confidence and affirmed my capabilities as a valuable contributor to the Information Security team.



## Appendix

I'm not able to provide specific scan results from the test I completed on the schools local network as they contain sensitive and potentially confidential information, as they would be openly exposing the vulnerable machines and their specific vulnerabilities.

## References

[1] NSA: <https://www.nsa.gov/Press-Room/Press-Releases-Statements/Press-Release-View/Article/3481350/cisa-nsa-fbi-and-international-partners-issue-advisory-on-the-top-routinely-exp/#:~:text=In%202022%2C%20over%2025%2C000%20new,2022%2C%20according%20to%20the%20CSA>

[2] Report from Coalition: <https://info.coalitioninc.com/cyber-threat-index.html>

The figures were taken from the web interface of Greenbone Vulnerability Management Community Edition, also known as OpenVAS.

Links: <https://greenbone.github.io/docs/latest/>, <https://github.com/greenbone/>

[https://www.bleepingcomputer.com/news/security/fbi-cisa-and-nsa-reveal-top-exploited-vulnerabilities-of-](https://www.bleepingcomputer.com/news/security/fbi-cisa-and-nsa-reveal-top-exploited-vulnerabilities-of-2022/#:~:text=While%20the%20Common%20Vulnerabilities%20and,of%20the%20top%2012)

[2022/#:~:text=While%20the%20Common%20Vulnerabilities%20and,of%20the%20top%2012](https://www.bleepingcomputer.com/news/security/fbi-cisa-and-nsa-reveal-top-exploited-vulnerabilities-of-2022/#:~:text=While%20the%20Common%20Vulnerabilities%20and,of%20the%20top%2012)

<https://www.statista.com/statistics/500755/worldwide-common-vulnerabilities-and-exposures/#:~:text=Published%20by%20Ani%20Petrosyan%20%2C,the%20highest%20reported%20annual>

<https://greenbone.github.io/docs/latest/22.4/container/index.html>

<https://forum.greenbone.net/>

<https://www.greenbone.net/en/>