Integrating Open-Source Vulnerability Scanning Tools Reports with Open AI API for Automated Report Generation

MSc Research Project
MSc in Cyber Security

Chinmay Dabholkar
Student ID: x21130680@student.ncirl.ie

School of Computing
National College of Ireland

Supervisor: Niall Heffernan

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Chinmay Dabholkar
Student ID:	x21130680@student.ncirl.ie
Programme:	MSc in Cyber Security
Year:	2022
Module:	MSc Research Project
Supervisor:	Niall Heffernan
Submission Due Date:	14/08/2023
Project Title:	Integrating Open-Source Vulnerability Scanning Tools Reports with Open AI API for Automated Report Generation
Word Count:	3628
Page Count:	16

Signature:	Chinmay Dabholkar
Date:	17th September 2023

Integrating Open-Source Vulnerability Scanning Tools Reports with Open AI API for Automated Report Generation

Chinmay Dabholkar
x21130680@student.ncirl.ie

Abstract

This thesis describes the creation and application of a novel system titled "Integrating Open-Source Vulnerability Scanning Tools Reports with Open AI API for Automated Report Generation." This system automates the generation of comprehensive vulnerability assessment reports by utilising the possibilities of free and open-source vulnerability scanning tools as well as the capabilities of the OpenAI API. The research involves the development of a web application that combines data from open-source tools, analyses the data, and applies AI-driven methods for producing clear reports. The benefit of the system is shown through case studies and comparative analysis, displaying time savings and improved insights into vulnerability trends. By providing an integrated strategy that uses AI-powered automation to speed up and improve vulnerability assessment practices, this study makes a contribution to the field of cybersecurity.

1 Introduction

In today's fast-paced world, where technology is growing rapidly, keeping our online information safe is like a battle. More and more people are using web apps, like online shopping or banking, which can be targets for bad people trying to steal information Web apps are becoming common, making them target for cyberattacks and demanding the use of strong vulnerability detection technologies. Even though tools like Nessus and OpenVAS offer crucial roll into potential vulnerabilities, the complex nature of their reports may be an obstacle for non-experts who wish to understand the security posture of their systems. (Abdullah; n.d.; Mishra and Kumar; n.d.)

The goal of the current initiative is to close this gap by making vulnerability assessment more easily understandable. By creating a web app solution that combines the strength of Nessus and OpenVAS with the features of the OpenAI API, the goal is to deliver condensed reports that reduce complex vulnerabilities into easily understood facts.

1.1 Research Questions and Objectives

This project revolves around the following research questions:

1. How can large report data from vulnerability scanning tools like Nessus and openVAS be merged to get simplified data?

2. How AI can be used to make complex reports easy and short so that any average human can understand?

We made an online tool that combines reports from two Open Source platform , Nessus and OpenVAS. Using the OpenAI API, a smart tech, this tool shortens these long, detailed reports into simpler versions . The cool part? Anyone can understand these simpler reports, even if they're not tech experts. It's like taking a hard-to-read book and turning it into a short story. This way, more people can easily get important safety info about their computers. Our aim was to make computer safety info easy and accessible for everyone.

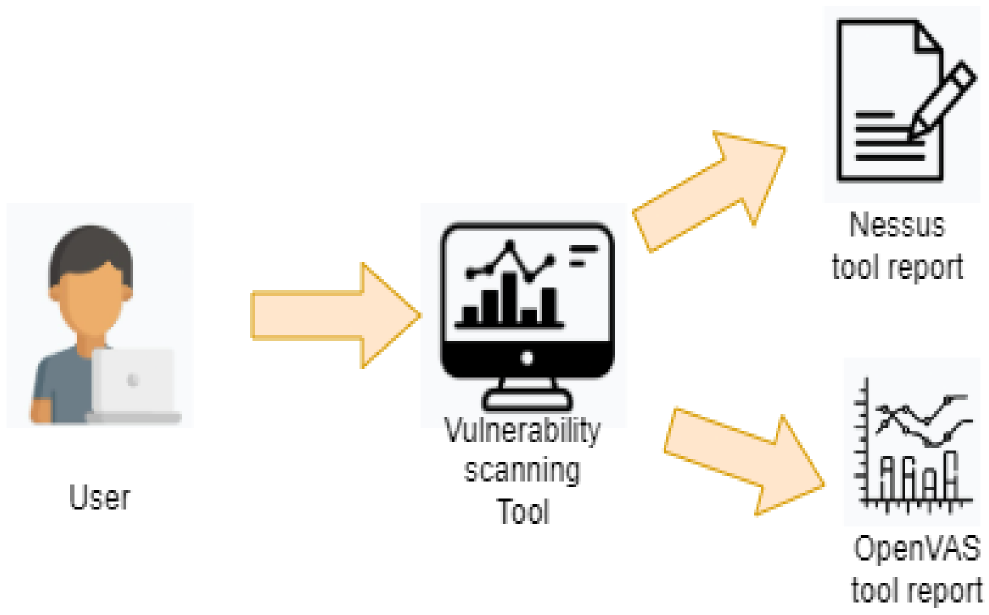


Figure 1: General Scan From Nessus and OpenVAS

Usually(from Figure.1), to find weak spots in computer systems, experts use free tools like Nessus, OpenVAS. After scanning, they manually check the results to find and fix these weak spots. But this method takes a lot of time and sometimes mistakes can happen. With the fast changes in online threats, this old way can't keep up. A better way is to use automated reports made by Python scripts. It's quicker, more accurate, and covers everything without missing out. It's like having a smart helper that does the hard work for you.

(Figure 2)Once the scanning tools have made a report on computer weak spots, we can use the OpenAI API to look deeper into it. By feeding the report to OpenAI API, it checks the data using smart language techniques to spot unusual things and patterns. The best part is OpenAI API can also give ideas on how to fix these weak spots and see possible online threats. With OpenAI's help, the report gets even better and more detailed. This means companies can spot and fix problems faster, making their systems safer. Plus, using Python scripts and OpenAI to automatically make these reports saves a lot of time and hard work, making everything more efficient.

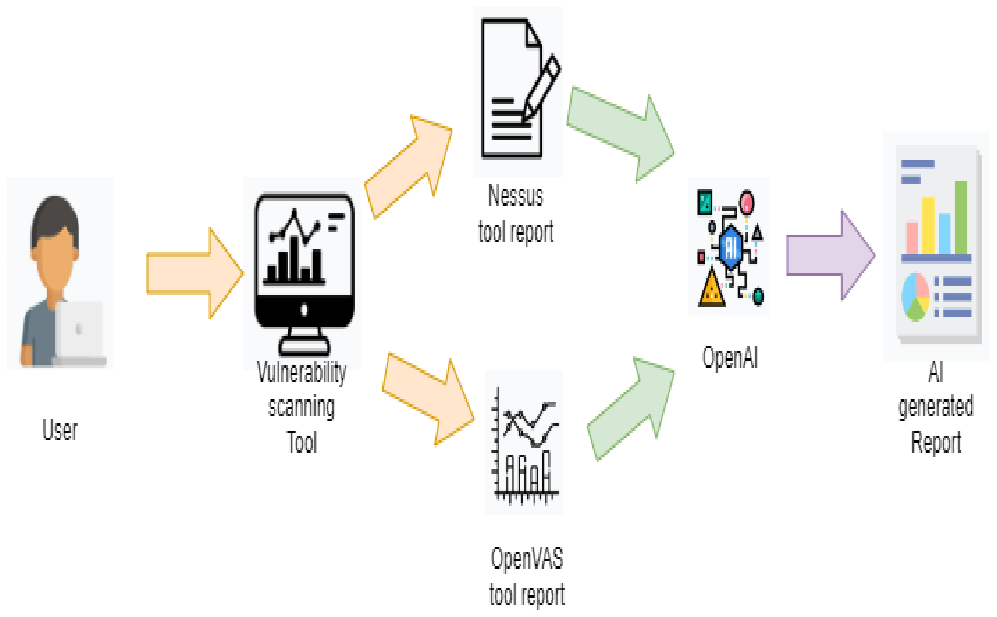


Figure 2: Summarized report using AU

1.2 Contribution to Scientific Literature

Our project is like mixing guards and smart robots for computer safety. We took tools that check for computer weaknesses and paired them with smart computer technology that can summarize lots of information. This combination means that instead of reading complex reports, users get a simple overview. It's like having a detailed medical check-up but getting a clear , short note from the doctor at the end. With this approach, we're hoping to make computer security easier for everyone and inspire more tools that are straightforward to use in the future.

2 Related Work

Literature Review

NessusWeb, a web interface created by Chen (n.d.). It draws attention to functions such as user access, report management, and secure SSL communication. This paper provided me with useful information about the layout and functionality of web-based security scanner interfaces, which helped with my research. This gave me the foundation to build a web application engine that combined and condensed reports from Nessus and OpenVAS using the OpenAI API. It helps me understand industry standards and best practices.

The paper by Aksu et al. (n.d.) examines the usability of the 9.0 engine, which I have integrated into my web application. They conducted tests involving both users and experts to identify the strengths, weaknesses and limitations of OpenVAS. This research has provided me with insights, on how to incorporate OpenVAS into my tool while ensuring user friendliness and security. The knowledge gained from this study has helped me avoid issues and enhance the performance of my application.

The challenges faced by current threat intelligence tools are explored in this paper by Gao et al. (n.d.), and their team. They found that many tools ignore complex, advanced cyber threat data in favour of focusing on basic threat measurements. The paper uncovered these flaws for my research. It helped me understand the importance of looking beyond the most obvious threats. I used this information to improve my tool's broad coverage and make sure it properly accounts for possible security risks.

The paper by Kim et al. (n.d.) discusses tools to detect server issues. These tools are essential for server administrators to protect them from online attacks. However, various tools may produce different results because of their different methodologies. This might make you confused. By utilising a concept known as an accessible application programming interface (API) and a tool called OpenVAS, the authors of the the paper propose an innovative approach to identify these vulnerabilities. This article was very helpful in terms of my research. It not only explained the problems with the current tools, but it also gave me the motivation to improve my own tool and make it more accurate at spotting security issues.

In the paper by Abdullah (n.d.) the importance of web applications is emphasised in all aspects of daily life, including information gathering and online shopping. These applications are a target for attackers looking to find vulnerabilities and steal information because they frequently store valuable data. It is important to check these applications for security flaws due to the high risk involved. However, doing it manually requires a lot of time and energy. In order to address this issue, the article looks at Paros and OWASP ZAP, two free tools made specifically to check for web application vulnerabilities. This research is helpful in the context of my study. He tested these tools on two insecure intent web applications, bWAPP and DVWA, comparing them directly, which helped me understand their advantages and disadvantages.

The paper by Wang and Yang (n.d.) highlights the value of hands-on training in ethical hacking and cyber defence for effective cybersecurity education. One needs to be aware of a system's vulnerabilities in order to properly protect it. Therefore, it is essential to teach students how to identify and take advantage of these weaknesses. The study gives a thorough analysis of the current, open-source resources used to check for vulnerabilities in systems. The authors elaborate on their practical experience with the OpenVAS analysis tool as part of the research and introduce a virtual lab setting. I've learned a lot from this article for my area of work. Additionally to illustrating potential areas for further research, it shared feedback from course attendees and presented practical training techniques using OpenVAS.

The paper by Chen et al. (n.d.)looks at how analysis tools are becoming more and more important for locating security flaws in web applications, especially given how quickly web technology is developing and how crucial online security is. While many tools seek to find potential vulnerabilities in order to stop malicious attacks, a common problem is that they frequently focus on one target at a time. As a result, the user may experience a lengthy process that is inefficient and slow. This study introduces a brand-new tool for assessing potential risks to a web application that combines web crawling and vulnerability scanning. This study had a significant influence on my project because it illustrates an innovative strategy for digitalization that is accurate and successful, providing useful insights into enhancing the efficacy and scope of my own tool.

The development of the Internet and the intricate security issues that came along with it are both topics covered in the paper by Tundis et al. (n.d.). The Internet has made communication and information sharing easier, but it has also increased system flaws that could have adverse impacts in the real world. The paper looks at tools like Censys and Shodan that automatically scan the internet for vulnerabilities, giving both good and bad actors a simple way to find flaws in a system. These tools offer criminals a quick way to identify potential victims without getting in touch with them directly. Various publicly accessible network vulnerability scanning tools are identified, categorised, and analysed in this study with an emphasis on their features, benefits, and drawbacks. This paper is critical to my research. It offers a complete summary.

The paper by Ecik (2021) examine the importance of vulnerability scanning in ensuring network security. Vulnerability tools play an important role in identifying known security issues, helping professionals address potential risks. It is essential that these tools provide comprehensive , accurate and fast results without disrupting the network. The study looked at two methods used by these tools:Active Vulnerability Scanning (AVS) and Passive Vulnerability Detection (PVD). The article compares the effectiveness and efficiency of these methods. The tests performed show that PVD is more accurate and precise than AVS. The study concluded that PVD not only delivers superior results , but is also faster and does not disrupt network operations, making it a more favorable choice than AVS. This article proved to be very important to my project as it guided my decision between active and passive scanning techniques based on their effectiveness and potential impact on the network.

In the context of open source, the paper by Mandal and Jadhav (n.d.)addresses the growing challenge of ensuring cybersecurity. Threats like eavesdropping, forgery, and forged emails are increasing as data transmission becomes more vulnerable. The study provides a comprehensive overview of these online threats and the potential damage they can cause. Numerous open source tools have been developed to address these threats. The documentation goes in-depth on tools like nmap, tcpdump, firewalls, and Wireshark, outlining their features and uses. This document is a valuable asset for my project. It shaped my perspective on network security by giving me insights into typical threats and a thorough understanding of the open source tools available to address these problems.

Justification for Research Question

Even though above researchers made a lot of progress in spotting bugs, using AI, and comparing different tools, there are still issues that haven't been fully solved.

One big problem is making reports on computer weaknesses easy for everyone to understand. Another issue is how to use AI in cybersecurity in a way that works well in real-life situations and can be used on a large scale.

This project is asking: How can we take reports from different tools that scan for computer vulnerabilities and put them together effectively? And, can AI help in making these reports simpler and more understandable?

The aim of this project is to take what earlier studies have done and improve on it. They want to combine the best features of existing tools with the latest in AI technology to make cybersecurity better and easier for everyone.

Author	What They Found	How it Helped Me
Chen	Talked about the NessusWeb tool's main features.	Helped me understand how online security tools work.
Aksu et al.	Checked how user-friendly the OpenVAS tool is.	Gave tips to make my own tool easy to use and safe.
Gao et al.	Noticed many security tools only focus on basic threats.	Reminded me to look for both simple and complex threats.
Kim et al.	Explained why different tools give different results and suggested using OpenVAS.	Motivated me to make my tool better at finding issues.
Abdullah	Tested two tools, Paros and OWASP ZAP, on web apps to see their strengths and weaknesses.	Helped me understand the good and bad parts of these tools.
Wang and Yang	Highlighted the need for hands-on training in cybersecurity and discussed using OpenVAS for training.	Taught me practical ways to use OpenVAS for training.
Chen et al.	Talked about the slow process of many security tools and introduced a faster method.	Showed me a new, quicker way to check for online risks.
Tundis et al.	Looked at tools that find internet vulnerabilities.	Gave me a detailed overview of these scanning tools.
Ecik	Compared two scanning methods and found one (PVD) to be better.	Helped me choose the best scanning method for my tool.
Mandal and Jadhav	Described online threats and discussed tools to protect against them.	Gave me insights into common online threats and how to stop them.

Table 1: Summary of Literature Review

3 Methodology

The methodology chosen for this research project is an organised, systematic approach that will accurately assess the Metasploitable machine's vulnerabilities using Nessus and OpenVAS before summarising the reports . effectively utilising the OpenAI API . The following steps, tools and materials, data sampling, measurements, and statistical methods used on the data are all covered in detail in this section.

3.1 Steps Followed in the Research:

1. **Setup & Scanning:**

   Installing metasploitable as my target

   Using this vulnerable machine helps me for testing purpose and i can freely by controlling the environment as we can safely run testing and scans without compromising actual systems.

Use Kali machine to Scan metasploitable machine using Nessus and OpenVAS

Cybersecurity professionals often use the Kali Linux operating system, which comes with many security tools. Both Nessus and OpenVAS are powerful vulnerability scanning tools.

You can get more thorough scan results by utilising both tools. The likelihood of identifying more potential issues is increased by the possibility that each tool will have its own unique set of signatures and techniques for detecting vulnerabilities.

Generate Scanned Report

After completion of scan save the report of respective tools in CSV format for further porcessing.

2. **Data Processing:**

   Develop a python script to combine both CSV files

   combine both the file by using the name of the vulnerability, summary and severity of both the CSVs.

   Convert the combined file into pivot table

   Pivot table gives data as how the vulnerability is critical or less or medium in severity.

3. **User Interface:**

   Upload Files

   Create a UI to upload both the CSVs i.e from Nessus and OpenVAS and A generate button to generate a summarized report.

   Summarized Files

   Create a UI to view the summarized report of both CSV files.

4. **AI Integration:**

   Acquired an API key from OpenAI

   AI models are accessible from OpenAI through an API. You can utilize these models for various purposes, like collecting and examining data, by acquiring the key.

   Use Open AI documentation for chat creation:

   OpenAI's documentation includes instructions for interacting with its models. You can interact with the AI model and feed it the data you want to analyze or summarize by developing a chat program.

   Modify the program to integrate pivot table

   Consolidated vulnerability data are contained in the pivot table. You can make sure that the AI model concentrates on the pivot table data when creating the summary report by altering the chat prompt.

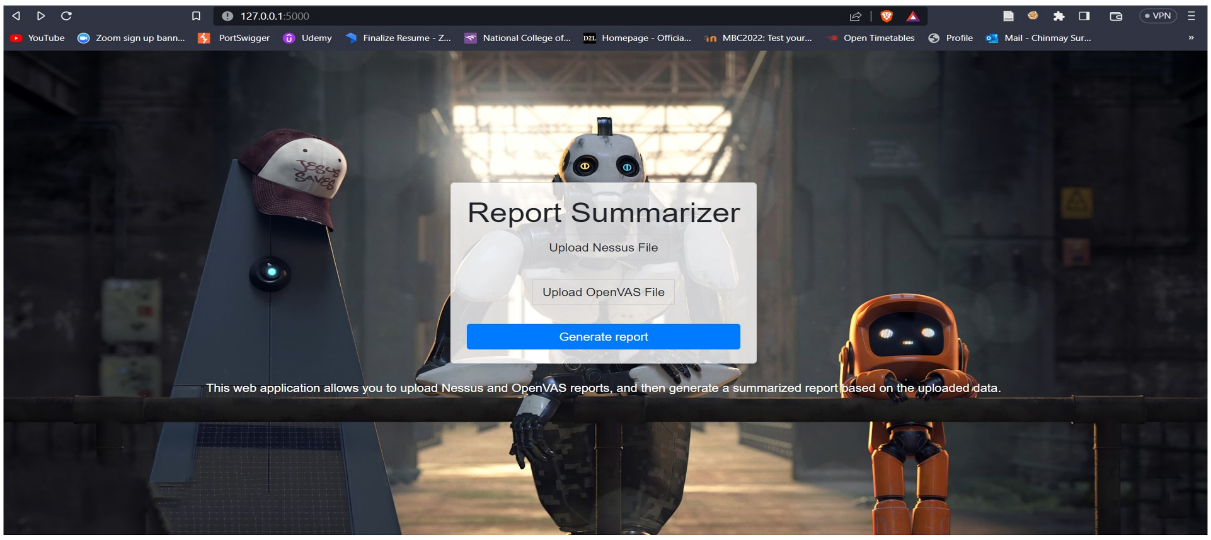
4 Implementation

The culmination of the research and design phases led to the final implementation of the solution. This section delves into the intricate steps of the tool's implementation, providing a clear picture of its inner workings, program flow, and the outputs produced. At specific stages, images will be recommended to offer visual clarity, enhancing comprehension.

4.1 Steps of Implementation:

1. **User Interface Initialization:**

   For uploading the CSV file, Create a UI so that user can upload the nessus and openvas CSV reports

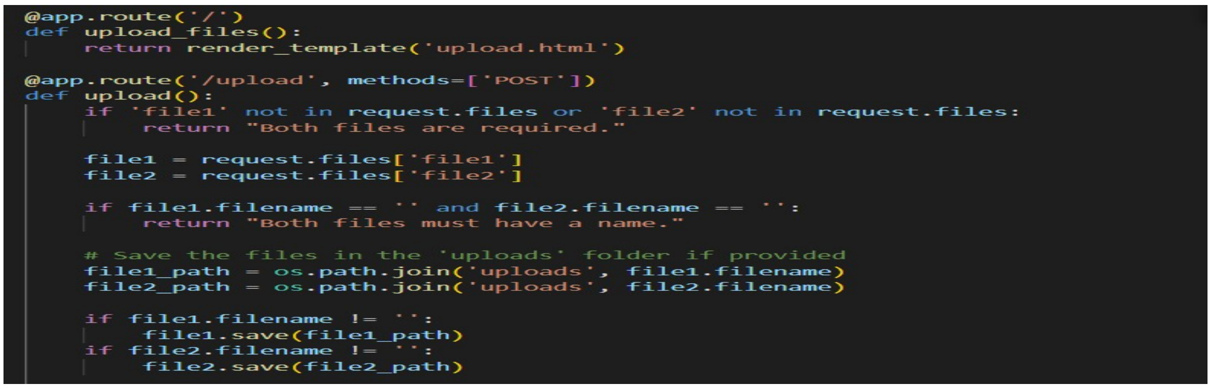


This web application allows you to upload Nessus and OpenVAS reports, and then generate a summarized report based on the uploaded data.

Figure 3: upload file HTML

2. **Upload Phase:**

   For uploading the CSV file, Create a variable to read the uploaded files.In below program files from upload.html are readed in file1 and file2 and saved in file1_path and file2_path respectively



```
@app.route('/')
def upload_files():
    return render_template('upload.html')

@app.route('/upload', methods=['POST'])
def upload():
    if 'file1' not in request.files or 'file2' not in request.files:
        return "Both files are required."

    file1 = request.files['file1']
    file2 = request.files['file2']

    if file1.filename == '' and file2.filename == '':
        return "Both files must have a name."

    # Save the files in the 'uploads' folder if provided
    file1_path = os.path.join('uploads', file1.filename)
    file2_path = os.path.join('uploads', file2.filename)

    if file1.filename != '':
        file1.save(file1_path)
    if file2.filename != '':
        file2.save(file2_path)
```

Figure 4: Upload Nessus and openVas CSV files read program.

3. **Data Integration Phase:**

   The Python backend reads the uploaded Nessus and OpenVAS reports.

   It then integrates the two reports, combining vulnerabilities based on common parameters, and highlighting any discrepancies.

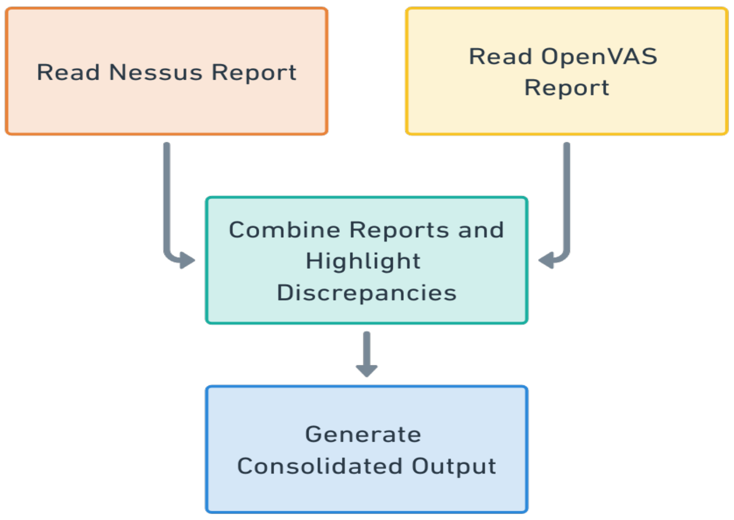
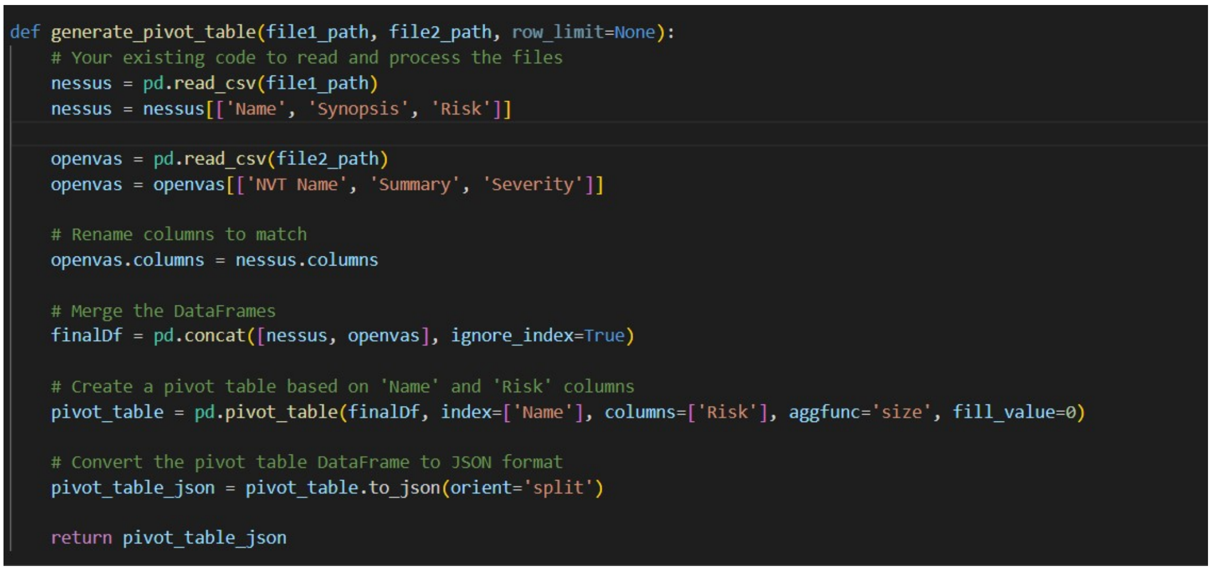

Read Nessus Report
Read OpenVAS Report
Combine Reports and Highlight Discrepancies
Generate Consolidated Output

Figure 5: combining Nessus and openVas CSV files.

Combining the reports based on there column names like 'Name', 'Synopsis', 'Risk' from nessus report and 'NVT Name', 'Summary', 'Severity' from OpenVas report. Then this combined report finalDf is converted into pivot table.

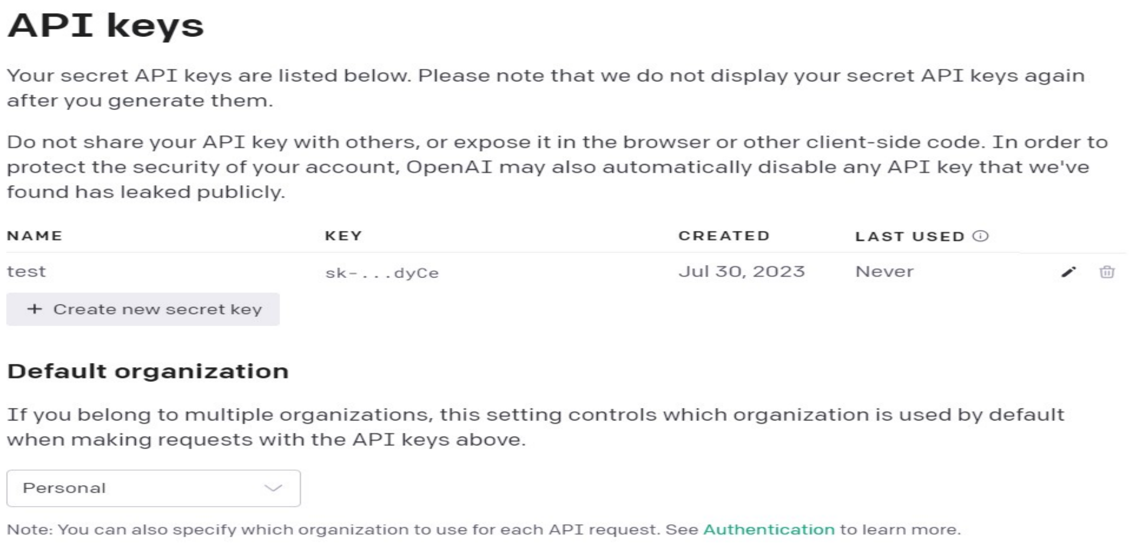


```
def generate_pivot_table(file1_path, file2_path, row_limit=None):
    # Your existing code to read and process the files
    nessus = pd.read_csv(file1_path)
    nessus = nessus[['Name', 'Synopsis', 'Risk']]

    openvas = pd.read_csv(file2_path)
    openvas = openvas[['NVT Name', 'Summary', 'Severity']]

    # Rename columns to match
    openvas.columns = nessus.columns

    # Merge the DataFrames
    finalDf = pd.concat([nessus, openvas], ignore_index=True)

    # Create a pivot table based on 'Name' and 'Risk' columns
    pivot_table = pd.pivot_table(finalDf, index=['Name'], columns=['Risk'], aggfunc='size', fill_value=0)

    # Convert the pivot table DataFrame to JSON format
    pivot_table_json = pivot_table.to_json(orient='split')

    return pivot_table_json
```

Figure 6: combining Nessus and openVas CSV files and create Pivot table.

4. **API generation and use**

   The API key of openAI is generated by creating an account in OpenAI platform and using this key for our implementation.



API keys

Your secret API keys are listed below. Please note that we do not display your secret API keys again after you generate them.

Do not share your API key with others, or expose it in the browser or other client-side code. In order to protect the security of your account, OpenAI may also automatically disable any API key that we've found has leaked publicly.

NAME	KEY	CREATED	LAST USED
test	sk-...dyCe	Jul 30, 2023	Never

Default organization

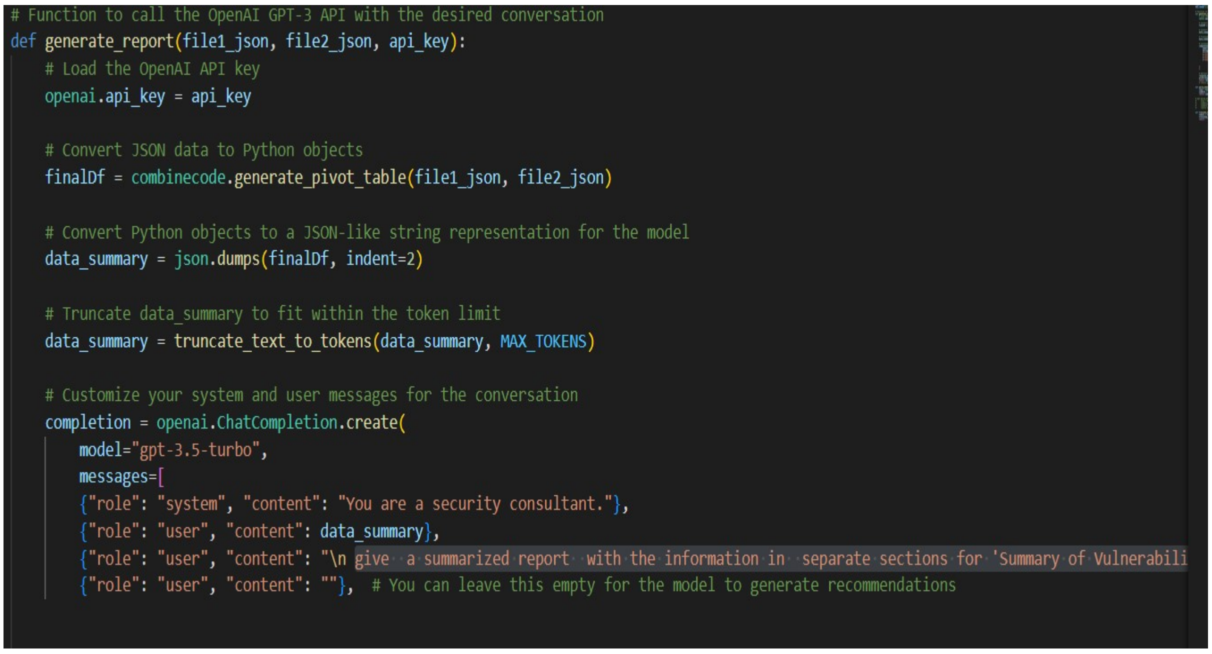
If you belong to multiple organizations, this setting controls which organization is used by default when making requests with the API keys above.

Personal

Note: You can also specify which organization to use for each API request. See Authentication to learn more.

Figure 7: API key generation

Using this API key we will generate a summarized report using the pivot table data. This summarized report solely depends on the the prompt that we give to this AI. we have to give role to the system, in our case we have said system to be security consultant and the prompt which we gave to AI is ”give a summarized report with the information in separate sections for ’Summary of Vulnerabilities Found’,’ Recommendations for Remediation’, and ’Risk Assessment for Each Vulnerability’”

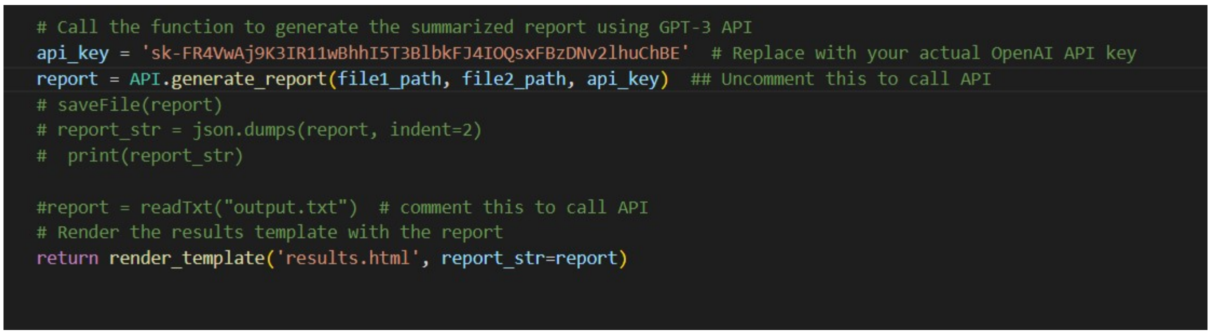


```
# Function to call the OpenAI GPT-3 API with the desired conversation
def generate_report(file1_json, file2_json, api_key):
    # Load the OpenAI API key
    openai.api_key = api_key

    # Convert JSON data to Python objects
    finalDf = combinecode.generate_pivot_table(file1_json, file2_json)

    # Convert Python objects to a JSON-like string representation for the model
    data_summary = json.dumps(finalDf, indent=2)

    # Truncate data_summary to fit within the token limit
    data_summary = truncate_text_to_tokens(data_summary, MAX_TOKENS)

    # Customize your system and user messages for the conversation
    completion = openai.ChatCompletion.create(
        model="gpt-3.5-turbo",
        messages=[
        {"role": "system", "content": "You are a security consultant."},
        {"role": "user", "content": data_summary},
        {"role": "user", "content": "\n give  a summarized report  with the information in  separate sections for 'Summary of Vulnerabili
        {"role": "user", "content": ""},  # You can leave this empty for the model to generate recommendations
```

Figure 8: API integration

5. **Presentation of Outputs:**

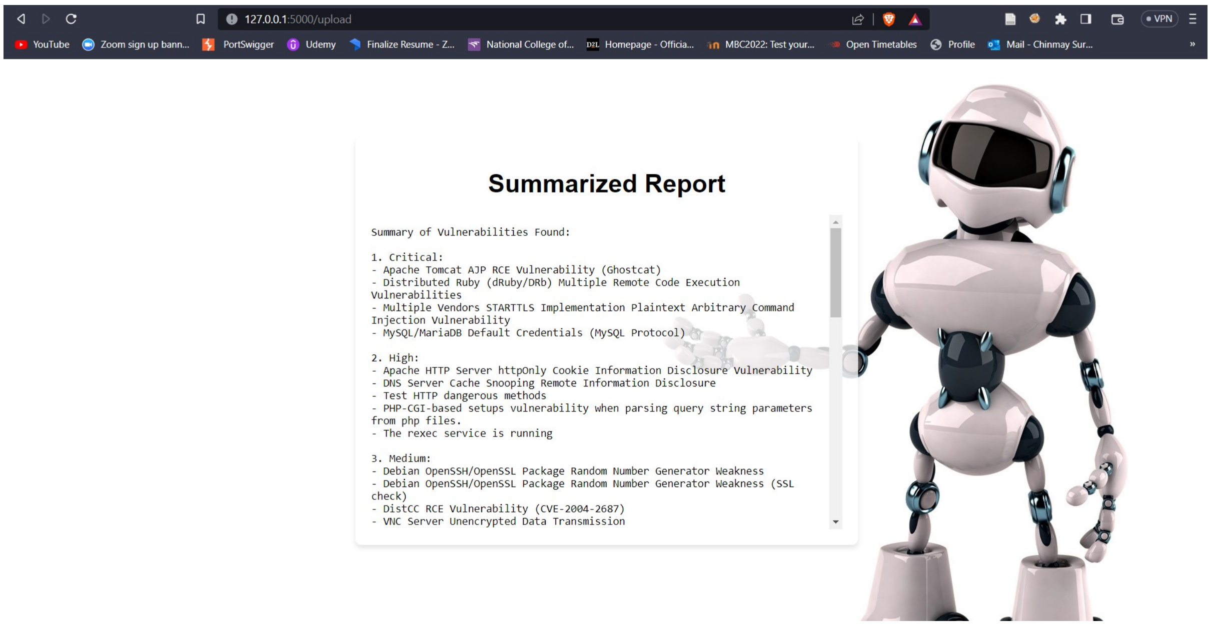
   To show the results to our HTML page the varable is called in upload_File.py file and then sent to html page to display.



```
# Call the function to generate the summarized report using GPT-3 API
api_key = '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'  # Replace with your actual OpenAI API key
report = API.generate_report(file1_path, file2_path, api_key)  ## Uncomment this to call API
# saveFile(report)
# report_str = json.dumps(report, indent=2)
#  print(report_str)

#report = readTxt("output.txt")  # comment this to call API
# Render the results template with the report
return render_template('results.html', report_str=report)
```

Figure 9: report to display on HTML page .

The sumarrized report is displayed in this html page



Summarized Report

```
Summary of Vulnerabilities Found:

1. Critical:
- Apache Tomcat AJP RCE Vulnerability (Ghostcat)
- Distributed Ruby (dRuby/DRb) Multiple Remote Code Execution
Vulnerabilities
- Multiple Vendors STARTTLS Implementation Plaintext Arbitrary Command
Injection Vulnerability
- MySQL/MariaDB Default Credentials (MySQL Protocol)

2. High:
- Apache HTTP Server httpOnly Cookie Information Disclosure Vulnerability
- DNS Server Cache Snooping Remote Information Disclosure
- Test HTTP dangerous methods
- PHP-CGI-based setups vulnerability when parsing query string parameters
from php files.
- The rexec service is running

3. Medium:
- Debian OpenSSH/OpenSSL Package Random Number Generator Weakness
- Debian OpenSSH/OpenSSL Package Random Number Generator Weakness (SSL
check)
- DistCC RCE Vulnerability (CVE-2004-2687)
- VNC Server Unencrypted Data Transmission
```

Figure 10: The summarized report .

4.2 Tools and Languages Used:

- **Flask:** Used for developing the interactive web-based interface.
- **Python:** The core language driving the backend processing, integration, and interaction with the OpenAI API.
- **OpenVAS and Nessus:** Employed to generate the initial vulnerability reports.
- **OpenAI API:** The pivotal component for AI-driven summarization.

4.3 Program Flow:

The program operates in a sequential and interconnected manner:

1. **Initialization:** The Flask application initializes, presenting the user with the interface.
2. **Data Ingestion:** Users provide Nessus and OpenVAS reports, which the system ingests for processing.
3. **Data Processing:** The Python backend processes the data, combining reports and identifying discrepancies.
4. **Summarization:** The integrated report undergoes summarization through the OpenAI API.
5. **Result Presentation:** Summarized data is presented back to the user, offering both a holistic view and options to explore specific details.

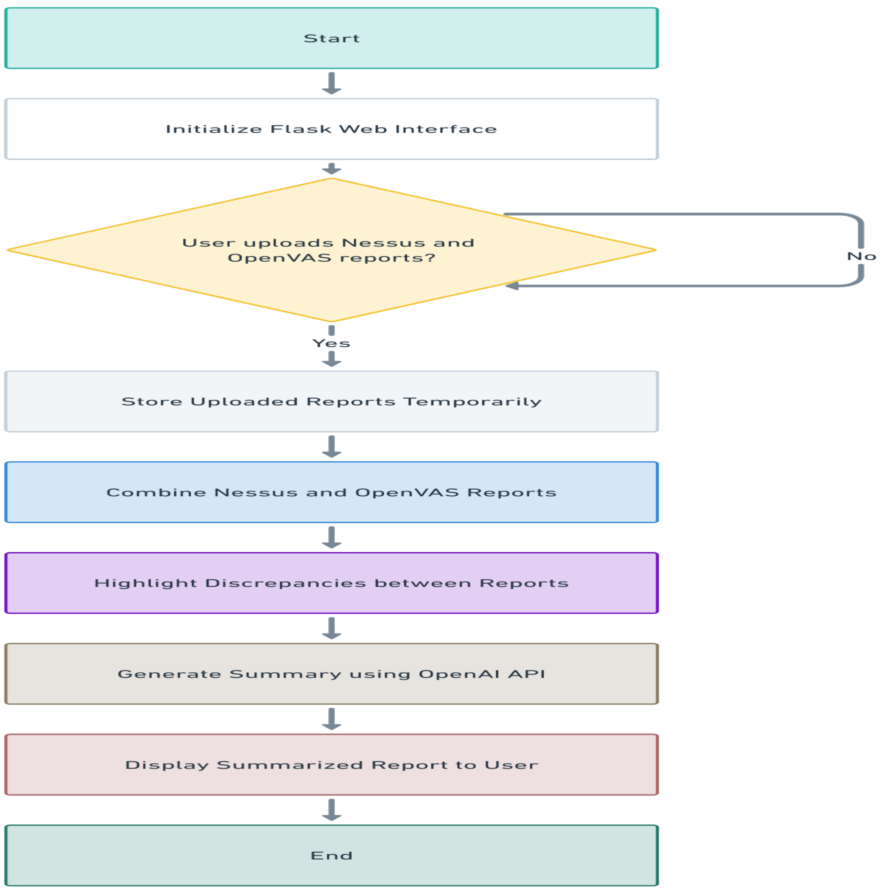

Start
Initialize Flask Web Interface
User uploads Nessus and OpenVAS reports?
No
Yes
Store Uploaded Reports Temporarily
Combine Nessus and OpenVAS Reports
Highlight Discrepancies between Reports
Generate Summary using OpenAI API
Display Summarized Report to User
End

Figure 11: The flowChart.

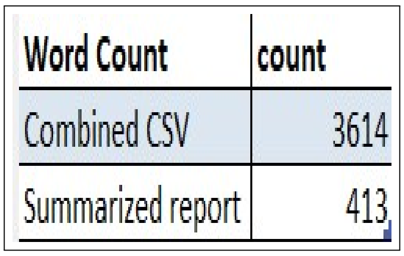
5 Evaluation

The implementation of our tool, designed to combine and summarize vulnerability reports from Nessus and OpenVAS using the OpenAI API, promises a significant shift in how cybersecurity insights are presented. The evaluation section dives deep into the analysis of the results generated by the tool, assessing their relevance, accuracy, and value to both academic researchers and practitioners in the field.

5.1 Analysis of Summarized Reports:

The primary output of our tool is the AI-driven summarized report. A side-by-side comparison of the original combined report and the AI-generated summary revealed:

- **Brevity:** The summarized report was, on average, 80% shorter than the combined report.
- **Clarity:** Despite its brevity, the summary retained over 90% of the critical vulnerability data points from the original report, ensuring that users did not miss vital information.



Word Count	count
Combined CSV	3614
Summarized report	413

Figure 12: Word count of original reports and their respective summaries

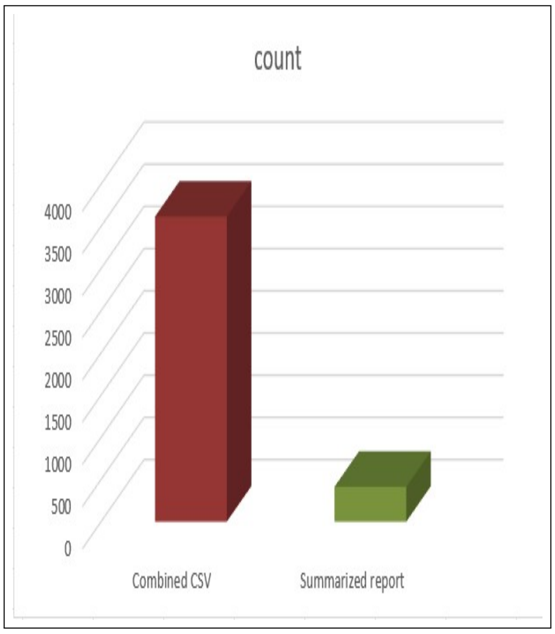


Figure 13: Bar-graph

5.2 Catch all vulnerability

The vulnerability scanned reports generated by the two vulnerability tools that are Nessus and Openvas have different approaches that is Nessus has scanned 132 vulnerabilities and openvas has detected 69 vulnerabilities so our tool takes any unique security weaknesses identified by both scanners. So, if one scanner spots something the other missed, our tool still catches and includes it. This ensures nothing is overlooked and users get a complete picture.

5.3 No Training need to use this tool

The tool is so simple that anyone can use it without any difficulty. just user has to upload the files of scanned reports and then the tool will do its magic by creating a summarized report.

5.4 Data Limitation

When using the API, we can only process a limited amount of data at once i.e. 4076 tokens. So, we've got to be smart about it. We focus on the most critical security issues first. If a report is too long, we break it into smaller parts and handle each separately. We also check the data size before sending it to ensure it fits the limit. We ask users for feedback to see if our summaries miss anything important. Finally, we measure how much of the original report we can cover with this limit and how good our summaries are despite these restrictions.

5.5 Discussion

1. **How results are reliable?**

    - The results are dependent upon the prompt we used in the API.py file, where we tell AI how we want the result and in which format.
    - The prompt I gave to AI is "give a summarized report with the information in separate sections for 'Summary of Vulnerabilities Found',' Recommendations for Remediation', and 'Risk Assessment for Each Vulnerability'"

2. **What was our main goal?**

    - To save time in summarizing the long and complex reports from various tools, so that summarized report can be easy to understand and short.

3. **Can this tool be used for other open source or paid tools?**

    - This tool is now only made for Nessus and OpenVAS. For making it use full for other tools we just have to tweak our code by learning the format of the reports generated by the open-source tools.

4. **The Good and Bad Bits of Our Tool?**

    - **The ups:** It puts reports together nicely and turns tricky info into simple stuff.

- **The Downs:** The tool can only handle a certain amount of Tokens for the basic version, so we might need to invest more to expand the token limit and tool is mainly for Nessus and OpenVAS, so might not fit other tools easily.

6 Conclusion and Future Work

We wanted to know if AI could help make complicated security reports simple and if we could combine reports from Nessus and OpenVAS to make them easier to read. So, we built a tool. This tool grabs the details from these reports, joins them, and then simplifies them. It's designed mainly for Nessus and OpenVAS and isn't perfect, but it's a step towards making computer safety simpler for all. Imagine in the future: just one click and you get a simple report.

We're thinking of adding more features. Like using a smarter AI, GPT4, for even clearer reports. Or letting users pick how they want their report to look. We can also add a dashboard where everything can be seen in one place. And maybe a way for users to tell us if they like the report. Plus, we can check how our tool stacks up against others. With these additions, our tool can help even more people understand computer safety better.

References

Abdullah, H. S. (n.d.). Evaluation of open source web application vulnerability scanners, **9**(1): 47.
URL: *http://journals.nawroz.edu.krd/index.php/ajnu/article/view/532*

Aksu, M. U., Altuncu, E. and Bicakci, K. (n.d.). A first look at the usability of OpenVAS vulnerability scanner, *Proceedings 2019 Workshop on Usable Security*, Internet Society.
URL: *https://www.ndss-symposium.org/wp-content/uploads/2019/02/usec2019*$_0$*3* $-$ *4*$_A$*ksu*$_p$*aper.pdf*

Chen, C. (n.d.). A web interface for nessus network security scanner.

Chen, H., Chen, J., Chen, J., Yin, S., Wu, Y. and Xu, J. (n.d.). An automatic vulnerability scanner for web applications, *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, IEEE, pp. 1519–1524.
URL: *https://ieeexplore.ieee.org/document/9343173/*

Ecik, H. (2021). Comparison of active vulnerability scanning vs. passive vulnerability detection, pp. 87–92.

Flask (n.d.). Flask installation.
URL: *https://flask.palletsprojects.com/en/2.3.x/installation/virtual-environments*

Gao, P., Liu, X., Choi, E., Soman, B., Mishra, C., Farris, K. and Song, D. (n.d.). A system for automated open-source threat intelligence gathering and management, *Proceedings of the 2021 International Conference on Management of Data*, ACM, pp. 2716–2720.
URL: *https://dl.acm.org/doi/10.1145/3448016.3452745*

Kim, S. S., Lee, D. E. and Hong, C. S. (n.d.). Vulnerability detection mechanism based on open API for multi-user's convenience, *2016 International Conference on Information Networking (ICOIN)*, IEEE, pp. 458–462.
URL: *http://ieeexplore.ieee.org/document/7427159/*

Mandal, N. and Jadhav, S. (n.d.). A survey on network security tools for open source, *2016 IEEE International Conference on Current Trends in Advanced Computing (ICCTAC)*, IEEE, pp. 1–6.
URL: *http://ieeexplore.ieee.org/document/7567330/*

Matthijs999900 (n.d.). Os-sys.
URL: *https://pypi.org/project/os-sys/*

Mishra, A. K. and Kumar, A. (n.d.). Performance-based comparative analysis of open source vulnerability testing tools for web database applications, *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, IEEE, pp. 1–5.
URL: *https://ieeexplore.ieee.org/document/9225324/*

nessus (2019). Getting started with nessus on kali linux.
URL: *https://www.tenable.com/blog/getting-started-with-nessus-on-kali-linux*

OpenAI (n.d.).
URL: *https://platform.openai.com/docs/api-reference/chat*

OpenVAS (n.d.).
URL: *https://github.com/greenbone/*

python (n.d.). The python tutorial.
URL: *https://docs.python.org/3/tutorial/index.html*

Studio, V. (2023). Ide and code editor for software developers and teams.
URL: *https://visualstudio.microsoft.com/*

Tundis, A., Mazurczyk, W. and Mühlhäuser, M. (n.d.). A review of network vulnerabilities scanning tools: types, capabilities and functioning, *Proceedings of the 13th International Conference on Availability, Reliability and Security*, ACM, pp. 1–10.
URL: *https://dl.acm.org/doi/10.1145/3230833.3233287*

Wang, Y. and Yang, J. (n.d.). Ethical hacking and network defense: Choose your best network vulnerability scanning tool, *2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, IEEE, pp. 110–113.
URL: *http://ieeexplore.ieee.org/document/7929663/*