Plug-and-play System for Scanning and Intelligent Diagnosis of Residential Network Vulnerabilities

Heitor Scalco Neto*, Gustavo Schwitzki Peretti*, Gabriel Moura Jappe*, Rodrigo Treichel César Pereira†, Alisson Borges Zanetti*, Danimar Veriato*, and Walter Priesnitz Filho†
*Instituto Federal Catarinense, Concórdia, Santa Catarina, Brazil
Corresponding e-mail: heitor.scalco@ifc.edu.br
†Universidade Federal de Santa Maria, Santa Maria, Rio Grande do Sul, Brazil
Corresponding e-mail: walter@redes.ufsm.br

Abstract—**Cybersecurity has become increasingly critical as the frequency of cyber attacks continues to rise. While critical infrastructures adopt measures to mitigate risks, the lack of protection in residential networks is often underestimated. This issue becomes more alarming as the number of reported vulnerabilities grows in parallel with the proliferation of Internet-connected devices and the expansion of the Internet of Things (IoT). In this context, this paper presents a plug-and-play vulnerability scanner built on a Raspberry Pi, designed to assist in identifying and addressing security flaws in residential environments. The proposed system integrates a Large Language Model (LLM) to generate user-friendly guidance for individuals without technical expertise. Its operation relies on three components: Nmap, used to detect devices and services within the network; the National Vulnerability Database (NVD) API, employed to retrieve Common Vulnerabilities and Exposures (CVE) associated with the detected services; and a Gemini client, which processes the retrieved CVE data to produce explanatory texts and practical remediation instructions. The system was validated in a scenario using a vulnerable target device running Metasploitable3. The results demonstrated that the application successfully identified multiple services and corresponding vulnerabilities, confirming its effectiveness in detecting potential threats. The significance of this implementation lies in its ability to present publicly available vulnerability information in a clear and accessible manner, contributing to the prevention of personal data exposure through compromised residential devices.**

Keywords—**Vulnerabilities; Nmap; Internet of Things.**

I. Introduction

As the number of cyber attacks rises, cybersecurity becomes increasingly important. In 2024, a total of 516,556 security incidents were reported in Brazil [1]. In this scenario, where cybersecurity appears threatened, governments and organizations take the initiative to create means to enhance data protection and regulatory compliance. Notable examples include the implementation of legal resources, as the General Data Protection Regulation (GDPR)[1] in European Union (EU), and the General Law for the Protection of Personal Data (LGPD)[2] in Brazil. These regulations required organizations to rigorously protect personal data in order to mitigate the impact of data breaches. Nevertheless, residential environments still have limited protection against cyber threats and their security tend to be underestimated [2].

This context turns more concerning as residential networks become more exposed with the advent of the Internet of Things (IoT), which introduces a large number of devices connected to the Internet with IPV4 and IPV6 addresses. If the software of these devices lacks proper support and maintenance to address new vulnerabilities, they may serve as entry points to the execution of cyber attacks. Simultaneously, the National Vulnerability Database (NVD) registered 39,974 vulnerabilities in 2024, revealing a growing exposure in digital systems that can affect a wide range of services existenting in residential environments [3]. An example of this threat is the recent incident of the *Vo1d* malware, which infected approximately 1.3 million Android TV boxes taking advantage of outdated versions of the operating system [4]. These factors highlight the need for tools capable of proactively identifying vulnerabilities in home networks to reduce exposure and potential data breaches and cyber attacks [5].

Given this context, this paper presents a solution that involves a plug-and-play device for vulnerability scanning in residential networks. The proposed system consists of a Raspberry Pi 3B running a script that automates network scanning using the Nmap[3] tool. This device connects to a remote centralized server, responsible for serving the devices with the set of vulnerabilities present in the network along with user-friendly informative texts with analyses of the vulnerabilities, providing valuable information, including risk metrics and remediation instructions. The search for the vulnerabilities is performed

[1]General Data Protection Regulation: https://gdpr-info.eu/

[2]*Lei Geral de Proteção de Dados*: https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/L13709compilado.htm

[3]https://nmap.org/

based on queries in the NVD, while the explanatory texts are generated by a client of the Gemini[4] large language model (LLM), specifically instructed for this task.

This paper is structured as follows: Section II reviews related work from the literature. Section III presents core concepts used in this study. Section IV describes the methodology used for the development of the tool. Section V and VI discuss the results of the present study.

II. Related Work

A complementary approach to the one proposed in this work explores the integration of Nmap with the Python programming language to automate network security audits [6]. The study describes how custom scripts can be used to perform periodic scans, identify known vulnerabilities, and generate automated reports. It also demonstrates the use of the Nmap Scripting Engine (NSE) to detect specific flaws, such as vulnerabilities in the SMB protocol, and highlights the potential of automation for incident response and penetration testing. Although the focus is on organizational environments, the proposal reinforces the importance of automation and customization in the vulnerability detection process, aligning with the goals of the plug-and-play system proposed in this paper.

Another relevant contribution proposes an open-source vulnerability scanner designed for system security auditing while remaining accessible to users without cybersecurity expertise [7]. The solution adopts a modular agent-server architecture, using low-cost hardware such as Raspberry Pi and using Nmap scripts, and automated reporting via e-mail. The system emphasis plug-and-play usability, reinforces the importance of automation, modularity and accessibility in the vulnerability detection process.

Similarly, an automated approach named Net-Nirikshak 1.0 [8] reinforces the relevance of accessible and modular tools for Vulnerability Assessment and Penetration Testing (VAPT), particularly in sensitive sectors such as banking. The system integrates automated modules for information gathering, service and vulnerability detection, SQL injection exploitation, and report generation, while minimizing the need for advanced technical expertise. Notably, it queries the NVD to identify known vulnerabilities and automates SQL injection exploitation, delivering results securely via e-mail. This solution also highlights the benefits of automation, modularity, and ease of use in improving security auditing practices.

Further contributions include a Raspberry Pi – Kali Linux Kit functioning as a vulnerability scanner [9]. The proposal leverages the comprehensive toolkit of Kali Linux in combination with the portability of Raspberry Pi devices. The proposed system offers a pre-configured platform with useful tools for network vulnerability scanning, such as Nmap. The framework was validated through comparative analysis with a desktop-based environment. The results highlight the flexibility and effectiveness of Kali Linux and the potential advantages of using Raspberry Pi in cybersecurity.

In another relevant study, a security vulnerability assessment was conducted on an Open Media Vault (OMV) cloud server running on a Raspberry Pi, using the PPDIOO network design model along with Nmap and Nessus tools [10]. The study identified several open ports and categorized vulnerabilities into high, medium, and informational levels. The author manually performed each step of the process – from installation and configuration to scanning and analysis – documenting it in detail. They proposed specific mitigation strategies, including software updates and configuration changes, to fix issues such as outdated Samba versions. This work highlights the importance of continuous monitoring and structured methodologies for securing lightweight NAS systems.

To better position our proposal within the context of these related works, we present Table I, which highlights the main characteristics of each approach. This comparison highlights how our solution stands out from the others by combining accessibility and automation with additional features designed for plug-and-play deployment and usability in resource-constrained environments. Table I illustrates the relevance of our contribution in addressing gaps identified in previous studies while reinforcing the practical applicability of automated and user-friendly vulnerability detection systems.

TABLE I
Comparison with other published approaches.

Work	**NVDLib**	**Plug & Play**	**AI**	**Nmap**	**Raspberry Pi**
[6]	×	×	×	✓	×
[7]	✓	✓	×	✓	✓
[8]	✓	×	×	✓	×
[9]	×	×	×	✓	✓
[10]	×	×	×	✓	✓
Our work	✓	✓	✓	✓	✓

III. Background

This section provides an overview of the main concepts among the technologies and standards used in this proposal, highlighting the features that were considered during the system design.

A. National Vulnerability Database

The National Vulnerability Database (NVD) is a software vulnerability repository maintained by the National Institute of

[4]https://ai.google.dev/gemini-api/docs

Standards and Technology (NIST). This initiative was introduced in 1999, when the Internet Category of Attack Toolkit (ICAT) was created as a database of attack scripts. In 2005, ICAT was rebranded to NVD. Furthermore, in 2007 the system was integrated to the National Checklist Program (NCP) and started relying on the Security Content Automation Protocol (SCAP), in order to manage vulnerability data automatically [11].

1) Common Platform Enumeration: The Common Platform Enumeration (CPE) is a naming system developed with the goal of standardizing the identification of classes of software, operating systems, and hardware devices. By using a structure based on the syntax of Uniform Resource Identifiers (URI), it is possible to create a common language between distinct systems, promoting consistency and security in the communication of information [12].

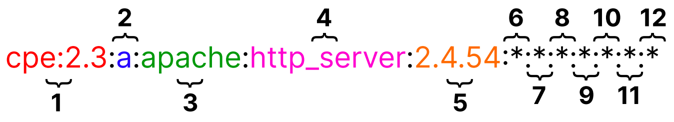

Fig. 1. Common Platform Enumeration (CPE) structure.

As shown in Figure 1, a CPE is composed of several parts between ":" that describe the asset in detail. Each field represents a specific characteristic, as the Table II shows. "*" is a character that represents all the possible values, like a wildcard.

TABLE II
FIELDS OF A CPE 2.3
FIELDS MARKED WITH * INDICATE THAT THEY CAN BE FILLED USING A WILDCARD (ASTERISK).

Figure Number	Field	Possible Values
1	CPE	2.3
2	Part	a (application), o (operating system), h (hardware)
3	Vendor	apache, microsoft, oracle, etc.
4	Product	http_server, windows_10, etc.
5	Version	2.0.1 or similar
6	Update*	alpha, beta, rc1, or similar
7	Edition*	Professional, 2021, or similar
8	Software Edition*	Enterprise, Community, or similar
9	Language*	en, pt-br, fr, or others
10	Target Software*	windows, linux, android, or similar
11	Target Hardware*	x86, arm, sparc, or similar
12	Other*	any additional qualifiers

2) Common Vulnerabilities and Exposures: The Common Vulnerabilities and Exposures (CVE) is a standardized system developed to identify and catalog publicly known cybersecurity vulnerabilities. It is managed by The MITRE Corporation[5] and sponsored by the U.S. Department of Homeland Security. Each CVE entry receives a unique identifier, facilitating scaling and consistent referencing across different security tools and platforms [13].

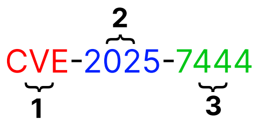

Fig. 2. CVE structure.

As illustrated in Figure 2, a CVE identifier follows a standardized format that includes the prefix "CVE", the year of disclosure, and a unique numeric identifier. This format ensures that each vulnerability can be referenced unambiguously across different systems and databases, as detailed in Table III.

TABLE III
FIELDS OF A CVE

Figure Number	Field	Possible Values
1	CVE	Always CVE
2	Year of Disclosure	2023, 2024, etc.
3	Identifier	Four number digit

Each CVE entry is associated with a detailed description of the vulnerability, including affected products, potential impacts, and references to advisories or patches. The CVEs are often used in conjunction with other standards such as the Common Vulnerability Scoring System (CVSS) to assess severity and prioritize remediation efforts [13].

B. Raspberry Pi

The Raspberry Pi is an affordable Linux-based computer, originally created to teach children computer programming, that has become a key enabler of Internet of Things (IoT) applications (Figure 3). It was developed by the Raspberry Pi Foundation, an organization dedicated to promoting computing education [14]. Although designed for educational purposes, the Raspberry Pi is now widely used as a flexible and powerful prototyping platform, particularly in automation systems and IoT scenarios [15].

C. Nmap

Network Mapper (Nmap) is a powerful scanning tool developed by Gordon Fyodor Lyon to map any kind of network. The first release of Nmap was in 1997, and over the years,

[5]https://www.mitre.org/

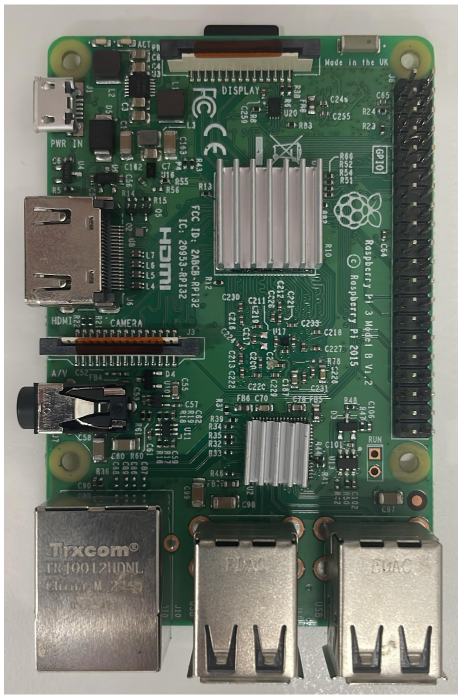
Fig. 3. Raspberry Pi 3B.

it has evolved into one of the most widely used tools in cybersecurity. The software is open-source and continuously updated to support new protocols and services, which allows the community to remaster and create their own versions [16].

The most common use of Nmap is to scan and find vulnerabilities in devices connected to a network, identify open ports[6], detect running services, and guess the host operating system [16].

D. Gemini

Gemini is a family of multimodal models trained by Google, capable of handling multiple types of data, such as audio, videos, texts, and images. This technology presents reasoning capabilities that allow for the development of generalist agents able to solve problems in a variety of domains [17]. The model was launched as “Bard” in 2023 and has been improved with the release of new models [18]. Gemini is also provided for developers through API keys, enabling them to build tools that integrate with the model [19].

IV. Development

With the aim of assisting residential network users in identifying vulnerabilities within their home networks, the proposed system acts as an automatic network vulnerability scanner. For this purpose, the implemented architecture is composed of a plug-and-play home device that identifies local software and services, together with a centralized server responsible for searching for new vulnerabilities and generating informational texts.

The selected hardware for the local scanning device was the Raspberry Pi 3B[7], which is a compact and low-cost device containing a network interface and is compatible with a wide range of software and operating systems [20]. We used the Raspberry Pi OS[8], a Debian-based system recommended for Raspberry Pi use cases.

The software stack, composed of Python[9] (version 3.11.2) scripts and a database instance, was managed via Docker[10] (version 28.3.3) containers. The database software chosen to locally store the network state was MariaDB[11] (version 12.0.2), an open-source relational database capable of representing relationships and entities, such as hosts, services, and vulnerabilities, as records in predefined tables.

There are two main programs that run in the local device: (i) a scanner and (ii) a web server. While the scanner (i) continuously updates the network state in the database, the web server (ii) is responsible for providing the scanning information in a web page. This web server was developed in Flask[12] due to its intuitive handling, as well as its wide range of tools that facilitate request processing [21].

For the scanning process, the system relies on Nmap, which is a popular open-source tool able to scan networks, detecting hosts, ports, services, and systems [16]. This application is notable for its performance in scanning large networks and can be used in vulnerability scanning by using its outputs to extract CPE identifiers to search for CVE matches [22].

With the aim of providing users with clear, useful information, and guidance on vulnerabilities, the system has an instructed LLM client used to generate the descriptive texts. Since Google provides free API keys to use Gemini, its models were used for content generation. However, configuring Gemini for personal use requires an account and key management [19], it was decided to maintain this configuration on a remote and centralized server.

The core component of the application is the scanner script, as shown in Figure 4. The scanning process runs continuously and can be divided into the following phases.

[6]Network ports act as entry points for data into a device. Open ports may expose services and potential vulnerabilities.

[7]https://www.raspberrypi.com/products/raspberry-pi-3-model-b/
[8]https://www.raspberrypi.com/documentation/computers/os.html
[9]https://www.python.org/
[10]https://www.docker.com/
[11]https://mariadb.org/
[12]A Python framework for developing web applications: https://flask.palletsprojects.com/en/stable/

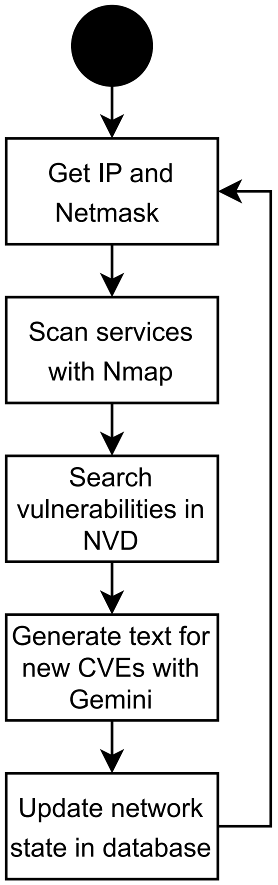
Get IP and Netmask
Scan services with Nmap
Search vulnerabilities in NVD
Generate text for new CVEs with Gemini
Update network state in database

Fig. 4. Scanning system flowchart.

1) **Get IP and Netmask**: Since Nmap works with IP packets [16], it depends on the input of a set, or a range, of IP addresses to be able to find hosts in the network. Therefore, before the scanning process, it is necessary to obtain the range of IP addresses that define the network scope to be analyzed. This information is managed by retrieving the Ethernet interface data through the operating system and inserting the network IP address and the netmask in the database. This stage needs to be executed repeatedly due to the fact that IP addresses are not permanent identifiers and can change dynamically.
2) **Scan services with Nmap**: Since the network address and mask are identified, the program executes a Nmap command to discover software services running in the range of addresses identified. An example of a Nmap command for the range of addresses 192.168.1.0/24 is presented below:

```
nmap -sV -O -T4 192.168.1.0/24 -oX -
```

A detailed explanation of the options utilized in the Nmap command is provided in Table IV.

TABLE IV
ELEMENTS OF THE NMAP COMMAND.

Element	Meaning
nmap	Refers to the Nmap executable file
-sV	Includes service detection
-O	Includes operating systems detection
-T4	Accelerates scan (uses ”agressive mode”)
192.168.1.0/24	Example of a range of addresses
-oX	Sets output format to XML
-	Sends output to the standard output instead of a file

The output of the Nmap execution is a Extensible Markup Language (XML) structure with metadata about the scan and a list of hosts, in which is possible to obtain the operating system and running services with CPE identifiers.
3) **Search vulnerabilities in NVD**: The services related to vulnerability searching and analysis are executed in the remote server. Consequently, when a local device concludes a scanning, it provides the CPEs identified in the network to the server, requesting the search for new vulnerabilities. Then, the server searches for CPE-CVE matches in the NVD API[13]. These queries are replied by the NVD with a JSON with details about the CVE matches, such as descriptions, metrics and references[14].
4) **Generate text for new CVEs with Gemini**: Instead of responding the requests directly with the information retrieved from the NVD API, the server generates an informative text[15] for every CVE record. The server uses a Gemini 2.0 Flash client instructed with a system instruction[16] designed to produce overviews of vulnerabilities containing the following components:
   - **Description**: a brief summary describing the vulnerability in clear language.
   - **Risk level**: explanation of the severity of the issue.
   - **Recommended actions**: instructions for resolving or mitigating the vulnerability.

   The input for the client is the CVE record from the API response along with the text content extracted from the web pages of some of the references of the CVE. Additionally, the Gemini client is directed to provide

[13]https://nvd.nist.gov/developers/vulnerabilities
[14]Example: https://services.nvd.nist.gov/rest/json/cves/2.0?cpeName=cpe:2.3:a:apache:http_server:2.4.54:*:*:*:*:*:*:*
[15]The text is generated in Portuguese.
[16]https://github.com/labsep/pishield-files/blob/main/system-instruction.txt

the response in Hypertext Markup Language (HTML), facilitating the integration of the generated content with the web interface available for the user.

5) **Update network state in database**: After each scanning, the server stores a snapshot of the latest network state in the database. This snapshot includes the hosts, services, and vulnerabilities identified during the most recent scan, ensuring that the web interface always reflects the latest available data.

The component of the system directly accessible for the residential users is the web server serving the interface for vulnerability information.

The criteria employed to validate the application include scan duration and performance during execution. To this end, a testing environment was set in a residential context. In order to create a vulnerable target, the Metasploitable3[17] system, which is a virtual machine designed with multiple vulnerabilities, was installed in a device in the network.

V. Results and Discussion

The benchmark results are presented in Figure 5. The total duration of the scanning process was 3,530 seconds (approximately 59 minutes). It can be seen that the majority of this time was spent on text generation on the server. In contrast, the Nmap scan, which retrieves the set of devices and services on the network, took 206 seconds (approximately 3 minutes and 26 seconds). This indicates that for the majority of the process, the Raspberry Pi remained idle, waiting for the server response.

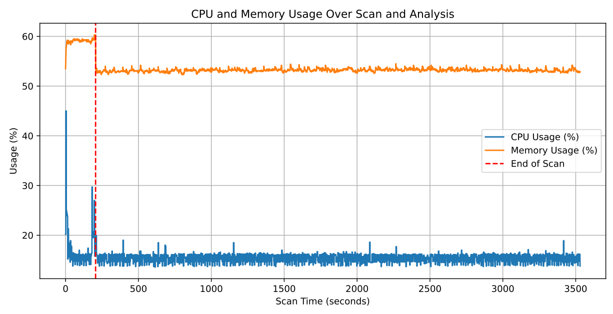

Fig. 5. System performance metrics.

The scan result is available at GitHub[18]. By combining service identification and vulnerability lookup through Nmap and the NVD API, a total of 7 services and 147 CVE records were detected in the Metasploitable3 system.

[17]https://github.com/rapid7/metasploitable3
[18]https://github.com/labsep/pishield-files/blob/main/scan.json

When the system performs the initial scan, it tends to take a considerable amount of time. Although this duration may be considered time-consuming, the system would still be capable to perform multiple scans daily if scans had constant duration. However, with storage in the server's database, subsequent updates of the network state are significantly faster, since only a few new vulnerabilites are tipically added and the text generated for each vulnerability is immutable and can be reused.

The generated texts provide information in simple language, allowing non-experts users to understand the issue and address the vulnerability. This feature is particularly important in the residential context, since it transforms technical information into user-friendly explanations. Another useful aspect is the OS prediction of Nmap. Although it is not always accurate, it may be useful in this approach by helping users to identify devices more easily, serving as additional information beyond IP and MAC addresses.

VI. Conclusions

In this scenario, where cybersecurity became more and more vulnerable, and the amount of IoT devices is increasing, people need more security, in this case, provided by the proposed software. The application successfully detected multiple services and vulnerabilities in the Metasploitable3 network environment, indicating that Nmap and NVD are capable of identifying a wide range of security issues. The integration of the detection system with LLMs potential to generate accessible explanations can benefit users without technical expertise for risk mitigation in residential environments.

This product can be useful for anticipating and preventing security compromise incidents in residential environments, enabling the effective use of public-interest information—such as CVE records—in mitigating risks and known vulnerabilities.

Our goal is to migrate the system to run in a lighter OS than Raspberry Pi OS, which allows better utilization of resources, such as RAM and processing power, optimizing the speed of the scans. As a future update, the system will search in specific IoT vulnerability databases, with the aim of covering a wider range of devices connected to the Internet. Furthermore, to ensure independence from external services, such as third-party LLMs servers, the solution will employ lightweight and locally deployed LLMs to support vulnerabilities analysis.

References

[1] CERT.br, "Incidentes notificados ao cert.br," 2025, acesso em: 8 out. 2025. [Online]. Available: https://stats.cert.br/incidentes/

[2] T. A. Coleti, R. Balancieri, A. Menolli, O. A. Mahmoud, V. H. Sotti, M. Yvano, and M. Morandini, "Handling of personal data by smart home equipment: an exploratory analysis in the context of lgpd," *Journal on Interactive Systems*, vol. 15, no. 1, pp. 311–322, 2024.

[3] National Vulnerability Database, "Search and Statistics," 2025, acesso em: 19 jul. 2025. [Online]. Available: https://nvd.nist.gov/vuln/search/statistics
[4] Dr.Web, "Void captures over a million android tv boxes," 2024, acesso em: 21 oct. 2024. [Online]. Available: https://news.drweb.com/show/?i=14900&lng=en
[5] H. Aarseth, "Identifying vulnerable services using non-intrusive techniques," Master's thesis, University of Oslo, 2023.
[6] J. Wołoszyn and M. Wołoszyn, "Using nmap and python for an automated network security audit," *Dydaktyka informatyki*, vol. 19, pp. 227–238, 2024.
[7] J. P. Seara and C. Serrão, "Automation of system security vulnerabilities detection using open-source software," *Electronics*, vol. 13, no. 5, p. 873, 2024.
[8] S. Shah and B. Mehtre, "An automated approach to vulnerability assessment and penetration testing using net-nirikshak 1.0," in *2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies*. IEEE, 2014, pp. 707–712.
[9] W. A. Alyani and A. Nasir, "Development of raspberry-pi kali linux kit as vulnerability scan," *International Journal of Synergy in Engineering and Technology*, vol. 5, no. 2, pp. 1–9, 2024.
[10] R. Ritzkal, P. P. Amalia, W. Mahmud, A. H. Hendrawan, B. A. Prakoso, I. Riawan *et al.*, "Security vulnerability analysis and recommendations for open media vault cloud server on raspberry pi," *Ingenierie des Systemes d'Information*, vol. 28, no. 3, p. 711, 2023.
[11] National Vulnerability Database, "General," 2025, acesso em: 19 jul. 2025. [Online]. Available: https://nvd.nist.gov/general
[12] K. S. Brant Cheikes, David Waltermire, "Common platform enumeration: Naming specification version 2.3," National Institute of Standards and Technology (NIST), Tech. Rep., 2011. [Online]. Available: https://csrc.nist.gov/pubs/ir/7695/final
[13] National Vulnerability Database, "CVEs and the NVD Process," 2024, acesso em: 19 jul. 2025. [Online]. Available: https://nvd.nist.gov/general/cve-process
[14] eLinux, "Rpi hub," 2023. [Online]. Available: https://elinux.org/RPi_Hub#About
[15] S. E. Mathe, H. K. Kondaveeti, S. Vappangi, S. D. Vanambathina, and N. K. Kumaravelu, "A comprehensive review on applications of raspberry pi," *Computer Science Review*, vol. 52, p. 100636, 2024.
[16] G. F. Lyon, *Nmap network scanning: The official Nmap project guide to network discovery and security scanning*. Insecure, 2009.
[17] G. Team, "Gemini: A family of highly capable multimodal models," 2025. [Online]. Available: https://arxiv.org/abs/2312.11805
[18] Gemini, "An overview of the gemini app," 2025. [Online]. Available: https://gemini.google/overview/
[19] G. A. for Developers, *Using Gemini API keys*, 2025, acesso em: 20 jul. 2025. [Online]. Available: https://ai.google.dev/gemini-api/docs/api-key
[20] D. Papakyriakou and I. S. Barbounakis, "Benchmarking and review of raspberry pi (rpi) 2b vs rpi 3b vs rpi 3b+ vs rpi 4b (8gb)," *International Journal of Computer Applications*, vol. 185, no. 3, pp. 37–52, 2023.
[21] Pallets, *Flask Documentation (3.1.x)*, 2024, acesso em: 30 maio 2025. [Online]. Available: https://flask.palletsprojects.com/en/stable/
[22] V. A. Saber, "Exploring the correlation between vulnerability scanning and nmap," *International Journal of Intelligent Computing and Information Sciences*, vol. 25, no. 1, pp. 41–50, 2025.