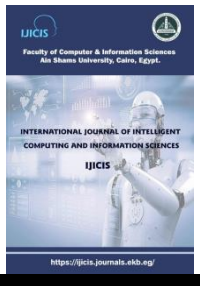
IJICIS, Vol.25, No.1, 41-50
DOI: 10.21608/ijicis.2025.370530.1381

International Journal of Intelligent Computing and Information Sciences

https://ijicis.journals.ekb.eg/



EXPLORING THE CORRELATION BETWEEN VULNERABILITY SCANNING AND NMAP

Verina Saber*

Faculty of Computer and Information Sciences, Ain Shams University,
Cairo, Egypt
ElSewedy University of Technology,
Cairo, Egypt
verina.adel@cis.asu.edu.eg

Dina ElSayad

Faculty of Computer and Information Sciences, Ain Shams University,
Cairo, Egypt
dina.elsayad@cis.asu.edu.eg

Ayman M. Bahaa-Eldin

Faculty of Engineering, Ain Shams University,
Cairo, Egypt
ayman.bahaa@eng.asu.edu.eg

Zaki T. Fayed

Faculty of Computer and Information Sciences, Ain Shams University,
Cairo, Egypt
zaki.taha@cis.asu.edu.eg

Received 2025-03-23; Revised 2025-03-23; Accepted 2025-03-29

Abstract: *Vulnerability scanning is a critical component of cybersecurity, enabling organizations to detect and mitigate security threats before they are exploited by malicious actors. This study compares the effectiveness of an Nmap-based vulnerability scanning framework with the for-profit Nessus scanner, focusing on accuracy, efficiency, and performance. The proposed framework leverages intelligent matching algorithms combined with automated scanning techniques to enhance detection capabilities. Experimental results demonstrate that the framework significantly reduces scan time, completing a comprehensive security assessment in just 3 to 4 minutes, whereas Nessus requires approximately 67 minutes for the same process. Despite faster execution, the proposed framework maintains high accuracy, achieving a recall rate of 95% by successfully identifying 37 of the 39 vulnerabilities detected by Nessus. These findings suggest that integrating advanced matching techniques with automated scanning tools not only accelerates the vulnerability assessment process but also maintains a high level of detection reliability, ultimately improving cybersecurity defenses.*

Keywords: *Python, Vulnerability Scanning, Nmap, PortScanner, CPE, CVE*

***Corresponding Author**: Verina Saber

Faculty of Computer and Information Science, Ain Shams University, Cairo, Egypt

ElSewedy University of Technology ,Cairo, Egypt

Email address: verina.adel@cis.asu.edu.eg

1. Introduction

Effective network scanning and vulnerability assessment are crucial to protecting systems against possible threats in the constantly changing field of cybersecurity. The enumeration is Nmap (Network Mapper) [1] that is one of the most well-known for this purpose. By offering comprehensive insights into network architectures, Nmap enables cybersecurity experts to identify open ports, operating services, and system configurations. Nmap is quite good at scanning and reconnaissance, but it can be made much more accurate, efficient, and scalable by automating its functions.
Network security automation gains a new level with the integration of Nmap with Python, which enables adaptive vulnerability detection, real-time analysis, and customized reporting. Because of Python's ease of use and adaptability, security experts can automate intricate scanning procedures, analyze vast amounts of data, and create unique security solutions. By offering automated control over scan operations, result filtering, and smooth integration into broader security frameworks, libraries like Python-nmap and PortScanner make it easier for Python and Nmap to interact. This combination helps security teams perform quick assessments, accelerate remediation efforts, and proactively mitigate vulnerabilities in addition to improving penetration testing operations.

The Common Platform Enumeration (CPE) output, which provides standardized naming conventions for software, hardware, and firmware, is an important component of effective vulnerability identification with Nmap. CPE enables security professionals to accurately map discovered assets to known vulnerabilities listed in the National Vulnerability Database (NVD). This correlation between detected services and vulnerability data makes risk analysis easier, allowing organizations to prioritize security threats and take preventive measures more efficiently. The use of Python to automate CPE-based vulnerability detection improves the process by allowing faster assessments and real-time security intelligence.

Furthermore, as cybersecurity threats evolve, intelligent and adaptive vulnerability assessment methods become increasingly important. Traditional scanning tools, while effective, frequently require manual intervention to interpret results and correlate vulnerabilities with evolving threat intelligence. By combining machine learning and AI-driven automation, the proposed framework improves vulnerability detection accuracy, reduces false positives, and prioritizes threats based on real-world exploitability. AI-powered analysis, combined with Nmap's scanning capabilities and Python's automation strengths, enables a more proactive security approach, reducing response times, and allowing security teams to focus on high-priority threats. This study aims to bridge the gap between traditional network scanning and intelligent cybersecurity automation by providing a solution that dynamically adapts to new vulnerabilities while improving overall network resilience.

This paper investigates the use of Nmap and Python to develop an automated vulnerability scanning framework that uses CPE-based mapping for precise vulnerability identification. The proposed approach reduces manual effort, speeds up scanning processes, and improves the detection accuracy of potential security risks. This framework aims to improve real-time security assessments by automating the correlation between CPE outputs and vulnerability databases while also providing a scalable and efficient solution for penetration testing and cybersecurity analysis.

The paper is organized as follows. In the following section, the background will be discussed. Moreover, Section III shows the related work. Section IV covers the methodology, which is divided into

four sub-sections: Study framework, dataset, CPE \& CVE, and data analysis. In addition, Section v shows the result. Finally, Section VI presents the conclusion and future work.

2. Background

In today's digital age, the prevalence of cybersecurity incidents has highlighted the critical need for proactive vulnerability management to protect sensitive information and maintain operational integrity. Vulnerability scanning has become a key component of cybersecurity practices, providing organizations with a systematic approach to identifying and addressing weaknesses in their IT infrastructure. This process not only reduces the risk of cyberattacks, but also ensures compliance with various standards and frameworks, including the NIST Cybersecurity Framework, PCI DSS, and SOC 2. The effectiveness of vulnerability scanning stems from its ability to detect a wide range of issues, from outdated software to misconfigured systems, allowing organizations to prioritize and address vulnerabilities based on risk level.

Because cybersecurity threats are constantly evolving, vulnerability scan methodologies must evolve as well. Traditional vulnerability scan methods frequently involved manual interventions that were time-consuming and prone to human error. However, as automated tools and scanning technologies have evolved, vulnerability management has changed to real-time monitoring and remediation. Automation not only speeds up the scanning process, but also improves accuracy by reducing false positives and negatives. Furthermore, the use of databases like the National Vulnerability Database (NVD) has made it easier to identify vulnerabilities by providing up-to-date information on known security issues and their potential consequences.

The role of vulnerability scan in risk management is an important aspect to consider. Effective scanning generates a detailed inventory of assets and associated risks, allowing organizations to efficiently allocate resources and develop targeted mitigation strategies. Identifying high-risk vulnerabilities in critical systems, for example, enables organizations to quickly deploy patches and configurations, reducing the likelihood of exploitation. Organizations can also build a resilient defense against emerging cyber threats by taking a cyclical approach to vulnerability management that includes continuous scanning, assessment, remediation, and verification. This proactive approach is consistent with industry best practices and fosters a culture of awareness and cybersecurity preparation.

Recent study emphasizes the need of incorporating vulnerability scanning into broader security frameworks to improve its effectiveness. According to research, combining scanning results with threat intelligence feeds and real-time monitoring systems can provide a complete picture of an organization's security posture. This integrated strategy allows for early identification of attacks and rapid reaction to vulnerabilities, decreasing the risk of data breaches and system compromises. Such improvements highlight the dynamic nature of vulnerability screening and its vital role in current cybersecurity strategy [2].

3. Related Work

Using Python, nmap's capabilities can be increased through automation and customization [2], allowing for bespoke solutions for specific organizational needs. Python integration provides advanced capabilities such as continuous monitoring, data analysis, and extensive reporting, resulting in far faster

and more accurate detection of security risks. Practical applications include automated detection of new devices, changes to service configurations, and vulnerability assessments.

The results demonstrate the usefulness of using these techniques to enhance cybersecurity operations. Examples include detecting significant vulnerabilities in real time and dynamically mapping network topology to visualize infrastructure changes. The study reveals that automated processes save time, minimize manual labor, and give detailed network security information. Advanced techniques, like as combining machine learning with Python to do intelligent scanning, improve the ability to identify and prevent attacks in advance. This integration emphasizes the importance of using new approaches to strong network security management in today's dynamic threat scenario.

M. Asaduzzaman et. al. [3] verifies the suggested framework's effectiveness in discovering vulnerabilities in Content Management Systems (CMS) using Nmap's scripting engine. It was tested on WordPress and Joomla web servers and was successful in identifying installed plugins and extensions, as well as their vulnerabilities and versions. The framework detected six plugins on a WordPress server, two of which were recognized as insecure, and four extensions on a Joomla server, one of which was susceptible. The framework was also tested on a network of nine CMS-based servers, identifying vulnerabilities in plugins and extensions and demonstrating its capacity to scan multiple hosts effectively. The scan times vary depending on the amount of extensions deployed and the network circumstances, indicating the system's adaptability and efficiency. These results highlight the framework's practicality, reliability, and accuracy as a cost-effective tool for assessing CMS vulnerabilities and enhancing web application security.

Additionally, Nmap and Nextpose can collaborate [4] for systematic scanning and analysis. Nmap was used to map the network and discover hosts, providing information on active hosts, open ports, and network services. Nexpose, on the other hand, conducted thorough vulnerability assessments, identifying and categorizing vulnerabilities, assigning risk levels, and recommending repair solutions. The study focused on three subnets at Bundelkhand University in Jhansi, India, where both technologies were utilized to detect and investigate network vulnerabilities. The findings emphasized the significance of these tools in providing a complete view of the network's security state, highlighting open ports that could be exploited and filtered ports that show the presence of security mechanisms such as firewalls.

The scan results identified a large number of vulnerabilities on multiple hosts, some of which pose a high risk of exploitation. For example, host 172.16.3.33 had the highest vulnerability count of 36, along with a high risk score, making it an important target for repair. Similarly, other hosts, including 172.16.6.35 and 172.16.3.212, have significant vulnerability numbers, emphasizing the importance of prioritizing security gaps. The study focused on the cyclical nature of vulnerability management, which includes discovery, scanning, analysis, prioritization, remediation, and verification.
To improve security, recommendations were made for continuing scans as well as the use of more powerful scanning commands and features. The authors concluded that an effective vulnerability management architecture, enabled by tools like Nmap and Nexpose, is critical for protecting university networks from emerging cyber threats. On the other hand, the authors propose that incorporation new tools or expanding the use of advanced Nexpose and Nmap features to improve vulnerability discovery and management.

Automation is necessary, as this study [5] focuses on automating the identification of cybersecurity vulnerabilities using open source software to solve the global scarcity of cybersecurity specialists and

the increasing threat of cyber assaults. The authors created a complete vulnerability scanning program that consists of three modules: network discovery, vulnerability assessment, and reporting. These modules locate network devices, evaluate vulnerabilities, and create detailed reports with remedial recommendations. The solution is designed for ease of use and does not require prior cybersecurity expertise, making it available to enterprises with modest resources. The solution is flexible, scalable, and adheres to DevSecOps principles, effortlessly integrating with modern security frameworks. The artifact uses Python, Nmap, and MongoDB to ensure low-cost deployment and high configurability.

External testers, including IT and cloud enterprises, telecoms organizations, and independent specialists, validated the system in both laboratory and real-world contexts. The results showed that the scanner is reliable, accurate in discovering vulnerabilities, and handles data securely. The solution effectively prioritized vulnerabilities and delivered actionable insights. Usability tests emphasized its efficiency and user-friendliness, and testers agreed on its potential to improve organizational cybersecurity. Future advancements included expanding automation functions and incorporating AI for predictive analytics. The tool is a promising addition to automated cybersecurity systems that prioritize accessibility and proactive threat management.

A. Aly et. al. [6] explores cyber deception methods and their usefulness in improving cybersecurity defenses. The study investigates several deception tactics, such as honeypots, decoy servers, faked websites, and deception networks, to deceive attackers and obtain information about their activity. It divides deception implementations into four layers: network, system, software (virus), and web-based deception.

The study examines Moving Target Defense (MTD) as a dynamic deception method, emphasizing its capacity to withstand persistent cyber assaults. It also looks at real-world implementations of deception frameworks as MTDCD (MTD Enhanced Cyber Deception Defense System) for network security, Moonraker for system-based deception, SODA (System for Cyber Deception Orchestration and Automation) for malware deception, and web-based deception approaches for attack detection. The study finishes with insights on future research areas, underlining the importance of adaptive and AI-driven deception methods to counter evolving cyber threats.

4. Methodology

4.1. Study Framework

Our framework Figure. 1 depicts the workflow of vulnerability scanning and reporting using Nmap and the National Vulnerability Database (NVD). The procedure begins with information collecting, which involves scanning an IP address (target). The Nmap libraries are used to determine crucial details such as the target's port number, service name and version, CPE (Common Platform Enumeration), and state status. This scanning phase captures the basic data needed for future processing, guaranteeing that system vulnerabilities may be successfully examined.

During processing, the extracted CPE is translated into an NVD database-compatible format. The translated CPE is then compared to entries in the NVD database to discover corresponding vulnerabilities. In the final reporting phase, the vulnerabilities are evaluated and assembled into a report that includes the number of vulnerabilities, descriptions, and severity rankings. This allows enterprises to prioritize remedial activities based on the severity of the vulnerabilities found.

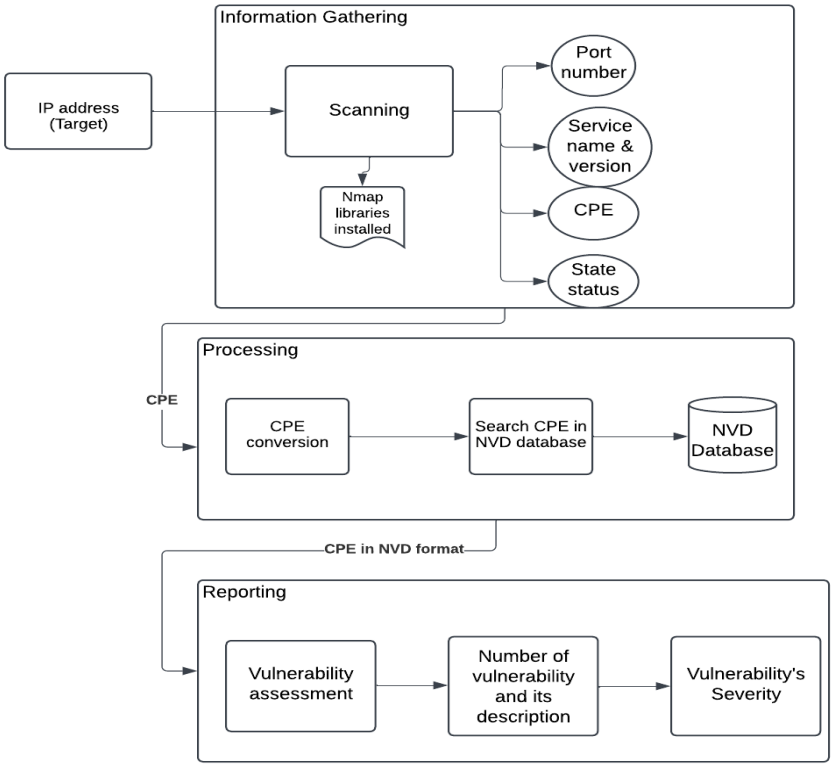


Figure. 1: Automated Vulnerability Detection and Analysis framework

4.2. Dataset

The dataset used in this study comes from the National Vulnerability Database (NVD), a comprehensive repository maintained by the National Institute of Standards and Technology (NIST). NVD gives structured information regarding software vulnerabilities, which is vital for analyzing risks in penetration testing and vulnerability scanning [7]. The dataset contains essential fields including Common Platform Enumeration (CPE), Common Vulnerabilities and Exposures (CVE) identities, CVE descriptions, and severity scores. These statistics are publicly available and provide up-to-date information about vulnerabilities in a wide range of systems, applications and platforms.

To establish a suitable dataset for this investigation, data was rigorously pulled from the NVD. The CPE entries were analyzed to determine specific software or hardware setups, and the CVE IDs were utilized to map vulnerabilities to their unique identifiers. Each CVE entry was augmented with a description that included a textual summary of the vulnerability as well as a severity score based on the Common Vulnerability Scoring System (CVSS). This organized dataset served as the foundation for vulnerability analysis, keyword matching, and risk assessment in target environments. The study uses standardized data to ensure the reliability and consistency of its conclusions.

4.3. CPE and CVE

CPE and CVE are standardized security identifiers used in cybersecurity, but they serve different purposes:

4.3.1 CPE

Common Platform Enumeration is a structured naming scheme for identifying hardware, operating systems, and software applications. It is used to uniquely define products and versions, helping security analysts and automated tools match software with known vulnerabilities. CPEs are commonly used in the National Vulnerability Database (NVD) to associate specific software and hardware products with vulnerabilities [8]. Its format is (e.g., cpe:2.3:a:oracle:mysql:8.0:*:*:*:*:*:*:*)

4.3.2 CVE

Common Vulnerabilities and Exposures is a database of publicly known security defects, each with a unique CVE ID. A CVE entry contains information about the vulnerability, its description, the affected software versions, and, in certain cases, references to patches or mitigation strategies. CVEs are crucial for tracking security risks and are frequently utilized in vulnerability assessment tools [8]. The format is (e.g., CVE-2021-21348)

4.4. Data Analysis

4.4.1 Data Pre-processing

The dataset used in this study was obtained from the National Vulnerability Database (NVD), which provides structured vulnerability information, including CPE, CVE ID, CVE descriptions, and severity scores. Before analysis, the data were pre-processed to ensure consistency and accuracy. The pre-processing steps included the following.

- Removing duplicates: Duplicate entries were eliminated to prevent redundancy in CPE-CVE mappings.
- Handling Missing Values: Any missing or incomplete data fields were either filled using alternative sources or removed if they did not contribute significantly to the analysis.
- Standardizing Formats: CPE and CVE identifiers were uniformly formatted to facilitate easier keyword-based matching.
- Extracting Key Fields: Only relevant fields (CPE-ID, CVE-ID, CVE description, severity score) were retained for analysis.

4.4.2 CPE-CVE Matching Algorithm

To identify vulnerabilities in specific software or hardware, a CPE-CVE matching process was implemented. The process involved:

- Keyword-Based Matching: CPE identifiers were extracted and converted into a format compatible with the CVE database.
- Filtering CVEs: The CVE dataset was filtered based on product names, versions, and vendor details to find relevant vulnerabilities.
- Handling Variations: Since different naming conventions exist across different sources, string normalization techniques were applied to match CPEs and CVEs accurately.
- False Positive Reduction: To minimize false positives, an additional verification step was performed, in which CVE descriptions were checked against extracted keywords to ensure relevance.

4.4.3 Used Tools and Technologies

The analysis was performed using various cybersecurity and data processing tools:

- Programming Languages: Python (for data processing and automation).
- Libraries: Pandas [9] (for data handling), NumPy [10] (for calculations), and Regular Expressions (for text matching).
- Databases: NVD (for fetching vulnerability data) and CSV datasets (for storing extracted data).
- Automation Tools: Nmap (for network scanning and CPE extraction).

5. Result

The effectiveness of a vulnerability scanning approach is measured by its efficiency (scan time) and accuracy (detection reliability). This section presents the results of our framework and compares its performance with Nessus as shown in Table 1, a widely used commercial vulnerability scanner.

5.1 Time Performance

The effectiveness of a vulnerability detection tool is critical in establishing its suitability for real-world cybersecurity applications. The suggested Nmap-based vulnerability scanning methodology performed significantly faster than the commercial Nessus scanner.

Our framework completed the scan on 3 IP addresses in approximately 3 to 4 minutes when scanning an IP address for vulnerabilities.
In contrast, Nessus took 1 hour and 7 minutes (67 minutes) to perform the same scan.
To quantify this performance improvement, we define the time efficiency equation 1:

$$T_{\text{efficiency}} = \frac{T_{\text{Nessus}}}{T_{\text{Our Framework}}} = \frac{67}{3.5} \approx 19.14 \quad (1)$$

Where:
• T_{Nessus} = 67 minutes (time taken by Nessus).
• $T_{\text{Our Framework}}$ = 3.5 minutes (average time taken by our scanner).

This means that our framework is approximately 19 times faster than Nessus in scanning vulnerabilities for a given IP address.

5.2 Recall

Recall [11] is a key metric used to evaluate the effectiveness of a vulnerability scanning framework compared to an established baseline, such as Nessus. Measures how accurately the framework detects vulnerabilities relative to a reference scanner, providing insight into its detection capability and reliability.

A high recall indicates that the framework correctly identifies most of the vulnerabilities detected by the baseline scanner, while a lower value suggests that some vulnerabilities may have been missed. This metric is particularly useful in cybersecurity assessments where accurate detection is crucial for effective risk mitigation.

Since our framework detected 37 vulnerabilities (True Positive) and Nessus detected 39 vulnerabilities (Positive), recall is calculated using the equation 2, and we get a true positive rate approximately reached 95%.

$$\text{Recall} = \frac{\text{True Positive}}{\text{Positive}} \tag{2}$$

Table 1 Comparison of Nmap-Based Framework and Nessus

Metric	Nmap-Based Framework	Nessus
Scan Time	3-4 minutes	67 minutes
Vulnerabilities Detected	37 vulnerabilities	39 vulnerabilities
Recall Rate	95 %	100%
False Positives	Low	Moderate
Efficiency	High	Low

6. Conclusion and Future Work

This study proposes a Nmap-based vulnerability scanning methodology that aims to improve the efficiency and accuracy of cybersecurity evaluations. The results show that the suggested framework greatly decreases scanning time, completing assessments in 3 to 4 minutes, whereas Nessus takes 67 minutes. This is an approximately 19x increase in efficiency, making the system more suitable for real-time vulnerability identification.
The framework accurately recognized 37 out of 39 vulnerabilities detected by Nessus, with a recall of 95%. This high true positive rate demonstrates that the suggested framework can be used as a quick and reliable replacement for standard vulnerability scanners while retaining detection efficacy. However, modest variations in the detection results indicate that more refinement is needed to increase coverage and accuracy.
In general, the findings highlight the potential of lightweight and automated vulnerability scanners in modern cybersecurity operations. By reducing scan time while maintaining high accuracy, this approach offers an effective solution for organizations that require rapid identification and mitigation of threats.

The suggested framework's future enhancements include the integration of Artificial Intelligence (AI) and Machine Learning (ML) to automate the production of test scripts for discovered vulnerabilities. By studying vulnerability descriptions, exploit databases, and historical attack patterns, the system can intelligently develop scripts that address specific security issues. This technology would allow security analysts to validate vulnerabilities more effectively, saving manual labor while enhancing accuracy. Furthermore, the framework could iteratively update the created scripts based on the execution input, increasing their effectiveness over time. By implementing AI-driven test script development, the system

would transition from a passive vulnerability scanner to an intelligent security testing solution, allowing it to handle real-time threat assessment and penetration testing.

References

1. https://nmap.org/.
2. J. W. Wołoszyn and M. Wołoszyn, “Using nmap and python for an automated network security audit,”Dydaktyka informatyki, vol. 19, 2024.
3. M. Asaduzzaman, P. P. Rawshan, N. N. Liya, M. N. Islam, and N. K. Dutta, “A vulnerability detection framework for cms using port scanning technique,” in Cyber Security and Computer Science: Second EAI International Conference, ICONCS 2020, Dhaka, Bangladesh, February 15-16, 2020, Proceedings 2, pp. 128–139, Springer, 2020.
4. K. Chhillar and S. Shrivastava, “University computer network vulnerability management using nmap and nexpose,” International Journal, vol. 10, no. 6, 2021.
5. J. P. Seara and C. Serr˜ao, “Automation of system security vulnerabilities detection using open-source software,” Electronics, vol. 13, no. 5, p. 873, 2024.
6. A. Aly, M. Fayez, M. M. Al-Qutt, and A. Hamad, “Navigating the deception stack: In-depth analysis and application of comprehensive cyber defense solutions,” International Journal of Intelligent Computing and Information Sciences, vol. 23, no. 4, pp. 50–65, 2023.
7. X. Li, S. Moreschini, Z. Zhang, F. Palomba, and D. Taibi, “The anatomy of a vulnerability database: A systematic mapping study,” Journal of Systems and Software, vol. 201, p. 111679, 2023.
8. A. Adithya, V. Vyas, M. Mohan, V. Aaswin, S. Lanka, et al., “Vulnerability scanning by cpe-cve matching.,” Grenze International Journal of Engineering & Technology (GIJET), vol. 10, 2024.
9. T. pandas development team, “pandas-dev/pandas: Pandas,” 2020.
10. C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. van Kerkwijk, M. Brett, A. Haldane, J. F. del R´ıo, M. Wiebe, P. Peterson, P. G´erard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, “Array programming with NumPy,” Nature, vol. 585, no. 7825, pp. 357–
    362, 2020.
11. P. Karageorgos, T. Richter, M.-B. Haffmans, J. Schindler, and J. Naumann, “The role of word-recognition accuracy in the development of word-recognition speed and reading comprehension in primary school:
    A longitudinal examination,” Cognitive Development, vol. 56, p. 100949, 2020.