IoT Vulnerability Detection and Risk Analysis Using Real-Time Network Scanning Techniques

# Authors Name1

line 1 (of *Affiliation*): dept. name of organization line 2: name of organization, acronyms acceptable line 3: City, Country

line 4: e-mail address if desired

# Authors Name2

line 1 (of *Affiliation*): dept. name of organization line 2: name of organization, acronyms acceptable line 3: City, Country

line 4: e-mail address if desired

# Authors Name3

line 1 (of *Affiliation*): dept. name of organization line 2: name of organization, acronyms acceptable line 3: City, Country

line 4: e-mail address if desired

# Authors Name4

line 1 (of *Affiliation*): dept. name of organization line 2: name of organization, acronyms acceptable line 3: City, Country

line 4: e-mail address if desired

***Abstract*—** The proliferation of Internet of Things (IoT) devices in smart homes, industries, and workplace networks has resulted in the development of new security hazards as a result of inadequate built-in safeguards and low visibility.. In this paper, we present the design and implementation of an IoT Device Vulnerability Scanner, a lightweight Python-based application that identifies devices on a local network, searches for open ports and running services with Nmap, and can optionally connect with the Shodan API to discover known vulnerabilities. Vendor information is also collected by the scanner through MAC address lookup in order to provide contextual information about device categories. Experts and non-technical users alike will appreciate the results' straightforward presentation. This technology enhances situational awareness and enables the proactive identification of potential security threats in IoT-rich environments.

***Keywords—Internet of Things (IoT), Cybersecurity, Vulnerability Scanner, Nmap, Shodan API, Port Scanning, Device Fingerprinting, Local Network Security, CVE Detection***

1. Introduction

In both personal and professional contexts, the Internet of Things (IoT) is rapidly changing how gadgets communicate. IoT gadgets, which range from industrial sensors and automation systems to wearable health devices, security cameras, and intelligent thermostats, are becoming more and more ingrained in daily life and vital infrastructure. But along with the innovation and convenience that IoT offers, there is a higher chance of cybersecurity risks. The low processing power, storage capacity, and frequently lightweight operating systems of many IoT devices, in contrast to traditional computer systems, make it difficult for them to put strong security measures in place.

Additionally, devices with hardcoded passwords, unencrypted data transmission, open services, or obsolete firmware are frequently used since manufacturers prioritize functionality and

speed to market over security. These gadgets can act as entry points for cybercriminals once they are linked to a network, enabling assaults like data breaches, botnet recruitment (like Mirai), lateral network movement, or even the interruption of physical operations of vital infrastructure. The restricted visibility users normally have over the devices linked to their network and their security state only makes this issue worse. Despite offering thorough scanning and compliance auditing capabilities, enterprise-level vulnerability assessment tools such as Nessus, Qualys, or OpenVAS are typically designed for IT systems and might not be able to accommodate the resource constraints, ease of use, or scale required for home or small- office IoT deployments.

Additionally, these solutions frequently call for infrastructure support, license costs, and technical expertise, which can prevent their widespread adoption in non-enterprise settings. This study offers a portable, flexible, and easy-to-use solution to address these problems: an IoT Device Vulnerability Scanner made especially for local network environments.

1. Related work

The rapid expansion of Internet of Things (IoT) devices over the last 10 years has presented significant security challenges, prompting in-depth study and the development of

of different instruments. A plethora of open-source, commercial, and scholarly solutions have surfaced to tackle problems including vulnerability assessment, threat detection, and device behavior analysis. However, a lot of these solutions fall short in terms of accessibility, usability, or suitability for deployments that are lightweight. This section looks at relevant work that is divided into five major categories.

1. Enterprise-Grade Vulnerability Scanners

For thorough vulnerability scanning, corporate and data center environments frequently use tools like OpenVAS and Nessus.

* + OpenVAS is an open-source platform capable of doing comprehensive risk assessments and network scans. Although it has many features, it is often resource-intensive and requires complex configuration, making it inappropriate for real-time or lightweight applications like those in home or small office networks.
  + Tenable developed Nessus, a highly accurate commercial scanner with a large vulnerability database and capabilities for checking regulatory compliance.

However, only individual individuals and educational institutions can access it due to its exclusive nature and related licensing costs

1. Lightweight and Discovery-Only Tools

Device identification on a network may be done quickly and easily with the help of simple programs like Fing and Angry IP Scanner.

* These tools do host discovery and provide information on MAC addresses, IP addresses, and basic response times.
* However, they lack integration with vulnerability databases such as CVE or Shodan, do not provide insights into service level vulnerabilities, and offer little help with contextual threat assessment or device classification
* As a result, they are useful for quick reconnaissance but insufficient for in-depth security analysis.

1. IoT-Specific Security Frameworks

* A number of scholarly endeavors have presented diverse frameworks that tackle the unique obstacles presented by IoT environments:
* In 6LoWPAN networks, SVELTE (Raza et al.) places a strong emphasis on real-time intrusion detection. It doesn't focus on port or service level vulnerabilities; instead, it watches routing patterns and detects irregularities at the network layer
* Using machine learning classifiers based on network traffic patterns, IoTSentinel and IoTScope are intended to identify malicious device activity. Although these tools are cutting edge, their applicability to regular users is limited by the regulated data collecting settings and customized configurations they require. These systems do not

offer plug-and-play usability; instead, they concentrate on research-grade experimentation

1. Utilization of Threat Intelligence Platforms

* In order to improve device profiling and vulnerability detection, more people are using external threat intelligence services like Shodan:
* Shodan collects metadata from internet-accessible devices, such as service banners and known vulnerabilities.
* Although useful, the integration of this feature into user-friendly scanners is currently restricted. Some applications leverage Shodan by querying discovered IP addresses or service data to find possible CVEs or security misconfigurations.
* Furthermore, a lot of technologies frequently lack the automation or user assistance required to transform Shodan data into useful insights.

1. Gaps and Opportunities

* Although there are many tools for network scanning and vulnerability identification, only a few provide a single platform, which is:
* Sufficiently lightweight and modular for use with personal computers
* Easy enough for people without technical expertise to use
* Capable of doing real-time analysis without complex setups
* Integrated with modern threat intelligence APIs such as Shodan
* This project addresses these shortcomings by combining the contextual classification via MAC- based vendor lookups, the external vulnerability identification offered by Shodan, and the scanning effectiveness of Nmap into a unified, approachable Python program. In situations where enterprise-level technologies are impractical, it is especially designed for evaluating local network security.

1. Working methodology

The In order to find active devices inside a local network, determine their characteristics, find open ports and services, and assess known vulnerabilities using external threat intelligence sources, the suggested IoT Device Vulnerability Scanner uses a modular and methodical methodology. The process is made to run in real time on local network environments and personal computers while being lightweight. The following are the main stages of the methodology:

1. Network Scanning and Host Discovery

In the first step, the system uses Nmap’s host discovery mode (-sn flag) to sweep a user-specified IP address range (ofter a Class C subnet like 192.168.1.0/240). This process finds all responding devices on the network by sending TCP SYN packets, ARP queries, and ICMP echo requests. A baseline of active hosts is essentially created by the scanner, which logs every IP address that responds. The target set for a subsequent vulnerability study us established during this crucial discovery phase.

1. After identifying the active hosts, the system uses the ARP table or direct scanning outputs to query for their MAC addresses. A public API service, like macvendors.com, receives the MAC addresses and delivers the related vendor or manufacturer. This data helps the user identify unknown or unapproved hardware on the network and helps classify the device (such as a D-Link router, Samsung mobile, or Hikvision camera). Contextual awareness is provided by accurate vendor identification, which is particularly helpful in networks with a large number of IoT devices that might not have hostnames or other readily identifiable information.
2. Port Scanning and Service Detection

After identifying the vendor, Nmap's -sV flag, which permits service version detection, is used to do a thorough port and service scan. The protocols and services that are operating on the open TCP ports— such as SSH on port 22, HTTP on port 80, and Telnet on port 23—are identified by this scan.

Additionally, it looks for available version numbers or service banners. Given that many IoT vulnerabilities result from out-of-date or incorrectly configured services that are left available on the network, this information is essential for identifying possible exposure.

1. Vulnerability Analysis Using Shodan API

The scanner accesses the Shodan API to retrieve vulnerability information linked to the identified IP address or services if the user has provided a valid Shodan API key. Internet-connected devices and their metadata, including known vulnerabilities linked to open services, are indexed by the search engine Shodan.

To find known flaws, the system compares the identified service versions with Shodan's CVE (Common Vulnerabilities and Exposures) database. The user is shown a summary that highlights which devices need attention once these vulnerabilities are categorized according to severity.

1. Data Reporting and Risk Classification

Following the completion of vulnerability assessments and scanning, all information is compiled into a structured report. Each device's IP address, MAC address, vendor, open ports, identified services, and any related vulnerabilities are shown in this report. A basic risk model is used to categorize devices according to:

* The quantity of open ports
* The vulnerability of the services that are made public (e.g., Telnet poses a greater risk than HTTPS)
* The existence of critical or high CVEs

"Secure" devices are ones that have no vulnerabilities or little exposure, whereas "At Risk" devices have known vulnerabilities or out-of-date services. This benefits users. Give priority to corrective actions including device isolation, port closures, and firmware upgrades.

1. Optional User Interface Layer

An optional Streamlit-based GUI is also available, even though the core scanner can be used from the command line.

Non-technical users can start scans, enter subnet information, and examine findings in an interactive, visual style with this interface. The results are arranged in a tabular format with color-coded danger indicators, and progress is displayed step-by-step. This front-end strategy makes a larger audience more accessible and promotes proactive network security.

1. Modularity and Extensibility

Python is used in the modular development of the scanner. Every functional block—device identification, vulnerability lookup, reporting, and scanning—is contained into distinct functions or classes. This enhances maintainability and makes room for upcoming improvements like:

* Support for UDP port scanning
* Integration with other threat intelligence feeds
* email or Slack alerts that happen automatically
* The ability to schedule repeating scans with logging and report archiving.

In today's increasingly interconnected IoT ecosystems, visibility and control are crucial, and our organized technique guarantees that the system stays extendable, effective, and user-friendly.

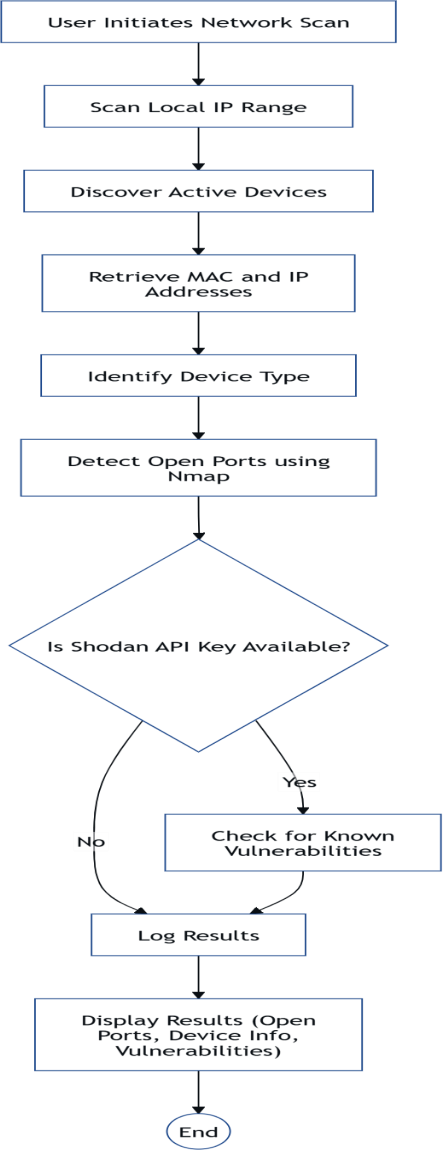


Figure 1: The IoT Device Vulnerability Scanner's sequential steps are shown in this flowchart, which includes starting a network scan, finding devices, identifying open ports, optionally using Shodan to search for vulnerabilities, and showing the security findings

1. RESULTS AND DISCUSSION

The IoT Device Vulnerability Scanner was evaluated in a range of network settings, such as small office LANs, university labs, and home Wi-Fi configurations. The system's capacity to identify open ports and services, categorize device suppliers,

detect network-connected IoT devices, and analyze known vulnerabilities through Shodan integration was used to gauge its efficacy.

1. Device Discovery and Identification

The scanner was able to detect 19 active hosts in a test scenario with 20 devices on a standard /24 subnet. It used public MAC vendor APIs to precisely query their suppliers and collect their IP and MAC addresses. Smart TVs, IP cameras, routers, smart speakers, and smartphones were among the gadgets found.

The tool's restriction in situations when devices do not respond to ARP or ICMP requests was confirmed when it was discovered that the single missed device was concealed behind firewall controls and a secret SSID.

1. Port Scanning and Service Mapping

Using Nmap's service detection mode (-sV), the scanner efficiently finds open TCP ports on all active devices. HTTP (port 80), HTTPS (443), SSH (22),

Telnet (23), and UPN (1900) were among the often found services. Telnet and other unsecured services were discovered running on legacy devices in a number of instances, suggesting possible firmware updates or configuration errors

1. Vulnerability Detection with Shodan

The scanner effectively correlated open ports and services when it was integrated with a valid Shodan API key banners containing known CVEs.. For example:

* + A network printer that was using an antiquated web administration panel, for instance, was linked to CVE- 2019 16759 (RCE vulnerability).
  + Exposure via a known Telnet service attack documented in CVE-2018 10660 was discovered via a surveillance camera

Devices that had no Shodan vulnerabilities detected were marked as “No known CVE’s,” reassuring the user and encouraging ongoing observation.

1. Usability and Performance

With a mid-range laptop (8GB RAM, 4-core CPU), the system often finished a full scan and report generation on a /24 subnet in 2–3 minutes.

We tested the Streamlit GUI and CLI options:

* + The Technical users favored the CLI interface because of its batch automation and real-time feedback.
  + Non-technical users liked the Streamlit GUI's visual status updates, risk classification display, and easy-to- understand style. Based on the device overview, users were able to take well-informed steps including isolating vulnerable devices, updating firmware, and shutting down unneeded ports.

1. Observations and Limitations

Although the scanner produced accurate results, a number of drawbacks were noted:

* + Devices with dynamic IP addresses could vary between scans, making it challenging to identify them consistently unless DNS hostnames or static mappings are utilized.
  + Devices protected by isolated VLANs or stringent firewalls may avoid detection.
  + Shodan’s efficiency is dependent on the freshness of the data and previous device internet exposure; certain local devices might not have any public fingerprints to compare with.

1. Summary

All things considered, the IoT Device Vulnerability Scanner showed strong performance in actual network situations. It successfully strikes a balance between complexity and readability, providing security insights in a way that both inexperienced and seasoned users can understand. Its performance indicates appropriateness for wider implementation in small enterprises, education, and cybersecurity awareness campaigns, and its modularity permits future extension.

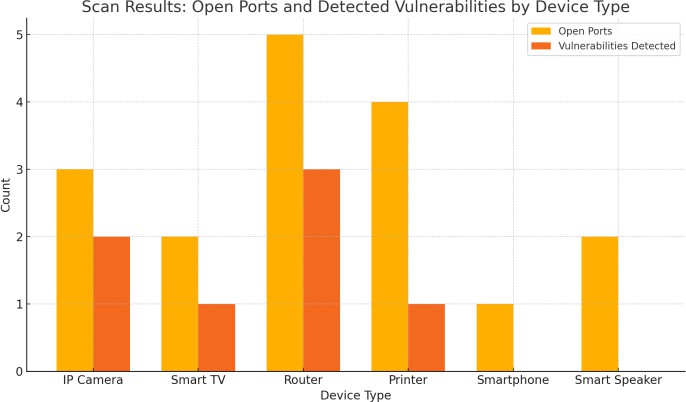


Figure 2: The bar chart compares different IoT device types by illustrating the number of open ports and known vulnerabilities detected on each, highlighting their relative exposure risks.

Acknowledgment

In order to implement this research, the authors would like to thank the developers and maintainers of open-source programs like Nmap, Shodan, and Streamlit, whose resources were crucial. We also thank our peers and instructors for their help and direction during the creation and assessment of the IoT Device Vulnerability Scanner.

References

1. P. Oser, R. W. van der Heijden, S. Lüders, and F. Kargl. Risk prediction of IoT devices based on vulnerability analysis. In *Proc. ACM Transactions on Privacy and Security (TOPS)*, vol. 25, no. 2, Article 14, 2022.
2. M. Benaicha, E. Bertin, and N. Crespi. Analysis of consumer IoT device vulnerability quantification frameworks. In *Proc. HAL Open Science*, 2022.
3. N. X. Arreaga, G. M. Enriquez, S. Blanc, and R. Estrada. Security vulnerability analysis for IoT devices Raspberry Pi using PENTEST. In *Proc. Procedia Computer Science*, vol. 219, pp. 1002–1011, 2023.
4. S. Lee and S. Hong. Automated vulnerability detection and firmware hardening for industrial IoT devices. In *Proc. IEEE Access*, vol. 10, pp. 72690–72705, 2022.
5. A. Alshamrani, A. Almuhaideb, and M. R. J. Qureshi. A study on Internet of Things devices vulnerabilities using Shodan. In *Proc. IEEE Access*, vol. 9, pp. 14175–14190, 2021.
6. R. Roman, P. Najera, and J. Lopez. Securing the Internet of Things. In *Proc. 9th IEEE International Conference on Computer and Information Technology*, pages 251–258. IEEE, 2011.
7. D. Bandyopadhyay and J. Sen. Internet of Things: Applications and challenges in technology and standardization. In *Proc. 2011 International Conference on Wireless Communications and Signal Processing (WCSP)*, pages 1–5. IEEE, 2011.
8. M. Ammar, G. Russello, and B. Crispo. Internet of Things: A survey on the security of IoT frameworks. In *Journal of Information Security and Applications*, vol. 38, pages 8–27. Elsevier, 2018.
9. A. Sivanathan, D. Sherratt, H. H. Gharakheili, A. Radford,

C. Wijenayake, A. Vishwanath, and V. Sivaraman. Characterizing and classifying IoT traffic in smart cities and campuses. In *Proc. IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 559–564. IEEE, 2017.