International Journal of Scientific Research and Engineering Development-– Volume 8 Issue 4, July Year 2025
Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

Identifying and Mitigating Device Vulnerabilities: A Statistical Approach to Upgrade Analysis

Dr. Mohammed Rafi R*, Asha N.S**, Shwetha A***
*(Professor, Dos in Computer Science and Engineering, University B.D.T College of Engineering, Davanagere, Karnataka, India, E-mail id: mdrafi2km@ubdtce.org)
** (PG Student, Computer Science, University of B.D.T College of Engineering, Davanagere
Email: ashans2819@gmail.com)
*** (PG Student, Computer Science, University of B.D.T College of Engineering, Davanagere
Email: shwethas2512@gmail.com)

---***********************---

Abstract:

The quick proliferation of Internet of Things (IoT) devices in many different fields has made security management and device maintenance extremely difficult. Cyber attacks are made more likely by outdated hardware and unpatched software flaws, and current upgrading procedures are still primarily manual and reactive. This study presents a simulation-driven, automated method for proactively assessing if IoT devices require upgrades. This study presents a simulation-driven, automated method for proactively assessing if IoT devices require upgrades. To identify security threats, the framework incorporates Common Vulnerabilities and Exposures (CVE) data and simulates various device setups using Python-based simulations. Prioritized update suggestions are produced by rule-based statistical studies that assess software vulnerabilities, hardware obsolescence, and patch compliance. While Matplotlib visualizations offer decision-makers clear, actionable insights, a user-friendly graphical interface created with Tkinter enables involvement easy. In along with enhancing security compliance and operational efficiency, the system also acts as a teaching tool for cybersecurity research. In order to further increase the robustness and manageability of extensive IoT deployments, future improvements will include real-time monitoring and machine learning-based predictive maintenance.

Keywords—**Device Security, Upgrade Analysis, Predictive Maintenance, Patch Compliance, IoT Vulnerability Assessment**

---***********************---

I. INTRODUCTION

An entirely new range of cybersecurity vulnerabilities has been brought about by the exponential development in the deployment of Internet of Things (IoT) devices across a variety of contexts, from large-scale industrial automation to residential smart home systems. These gadgets usually run on legacy operating systems or firmware that is out of date, have limited computational power, and lack the mechanisms needed for automatic security updates. Frequently, manufacturers put cost-effectiveness and utility ahead of a strong security architecture, which leaves devices vulnerable to new cyberthreats. Consequently, these systems become extremely vulnerable to assaults including malware insertion, botnet recruiting, data interception, and unauthorized access. Organizational approaches to vulnerability mitigation are still mostly reactive, even with the growing knowledge of these concerns. Manual inspections or ad hoc decisions about firmware or hardware upgrades are common in current processes, which are frequently initiated only after security events have taken place. The security and dependability of IoT ecosystems are seriously compromised by the absence of a proactive, standardized, and scalable upgrade procedure, particularly in critical infrastructure domains.

This study suggests and puts into practice a simulation-based analytical framework intended to facilitate automated and statistically supported decision-making for IoT device updates in order to solve these shortcomings. A variety of modules that mimic real-world operating settings, device lifecycles, and potential threat vectors are included in the Python-developed framework. Tkinter is used to create the graphical user interface (GUI), which allows users to engage with the system via an easy-to-use frontend. Additionally, the framework uses Matplotlib for dynamic data visualization, which enables users to view and analyze patterns in security status, device performance, and upgrade urgency through detailed plots and charts. In addition to incorporating criteria like firmware age, known vulnerability ratings (like CVSS), device utilization, and patch availability, the simulation creates synthetic datasets that replicate real-world deployment settings. The need and timing of upgrades are then assessed using statistical analysis, providing a scalable and reproducible method for ongoing security auditing. This prototype serves as a fundamental paradigm for future integration with enterprise-grade security management platforms and machine learning-based predictive systems, in addition to facilitating better informed upgrade planning.

II. LITERATURE REVIEW

In recent years, the widespread use of Internet of Things (IoT) devices has transformed a number of industries, including home automation, smart cities, healthcare, and manufacturing. But this expansion has also added another level of difficulty to the upkeep and security of these gadgets. Determining which devices need hardware and software upgrades is one of the most urgent problems, particularly in large-scale installations involving hundreds or thousands of nodes [1], [2]. Uniform management is impossible since every device may have a distinct configuration, firmware version, operating environment, and level of threat exposure.

Conventional device maintenance approaches frequently only address problems after a security breach or system failure has occurred, mostly depending on human tracking or reactive efforts [3]. The total resilience of the IoT infrastructure is decreased by this lack of proactive management, which exposes systems to known attacks. Furthermore, a lot of businesses don't keep organized databases to monitor firmware upgrades or evaluate patch compliance. In the absence of such systems, it is challenging to use established threat intelligence resources like the Common Vulnerabilities and Exposures (CVE) database to identify out-of-date hardware, assess software vulnerabilities, or ascertain the current patching status [4].

To overcome these shortcomings, this research introduces an automated simulation-based framework for evaluating IoT upgrade requirements. The framework is designed to:

- Detect outdated hardware components based on lifespan models.
- Identify exploitable software vulnerabilities using CVE data and severity ratings from the Common Vulnerability Scoring System (CVSS).
- Assess patch management compliance and recommend corrective measures.
- Visualize insights in a user-friendly format for technical and non-technical stakeholders.

This approach provides a structured and repeatable method for ensuring proactive maintenance and enhanced security posture.

A. Existing System

Current approaches in IoT device maintenance and upgrade management are characterized by several limitations:

1) ***Manual Upgrade Practices***: Most organizations continue to rely on human oversight for tracking device versions, vulnerabilities, and patch statuses. In addition to being time-consuming, this procedure is prone to errors and oversights. [5].

2) ***Reactive Security Strategies:*** Updates are frequently applied only after security incidents have occurred, leading to increased system exposure and downtime [6].

3) ***Lack of Visualization:*** Existing tools rarely incorporate data visualization or intuitive dashboards that can help system administrators quickly identify trends and prioritize actions.

4) ***No Simulation Frameworks:*** Current maintenance systems generally lack the ability to model device behavior in a simulated environment, hindering testing, prediction, and training capabilities.

These limitations significantly reduce the efficiency, scalability, and security of IoT device management systems.

B. Proposed System

To address the issues outlined above, the proposed system introduces a novel simulation and visualization-based framework. Key features of the system include:

1) ***Automated Simulation Engine:*** The framework can simulate the behavior and lifecycle of various IoT devices, each with different firmware versions, hardware specifications, and deployment environments. This allows researchers and administrators to assess upgrade needs in a controlled setting.

2) ***Rule-Based Statistical Analysis:*** The system uses a set of rules and statistical techniques to evaluate vulnerabilities and prioritize upgrades. These rules are based on hardware age, software version discrepancies, known CVEs, and update availability [7].

3) ***Graphical User Interface (GUI):*** Developed using Python's Tkinter library, the interface allows users to easily initiate simulations, analyze device health, and explore upgrade recommendations.

4) ***Matplotlib-Based Visualization:*** Analytical results are presented through charts and graphs generated using the Matplotlib library. These visuals include bar graphs of vulnerability severity, time series of patch compliance, and heatmaps showing device risk levels.

5) ***Educational and Research Applications:*** The tool is suitable not only for operational deployment but also for use in cybersecurity education and academic research, offering a practical environment to explore upgrade strategies and simulate threat scenarios.

This framework bridges the gap between reactive maintenance and intelligent upgrade planning, thereby enhancing the resilience and manageability of IoT systems.

TABLE I
COMPARATIVE SUMMARY

Feature	**Existing System**	**Proposed System**
Device Evaluation Method	Manual or static rules	Automated rule-based simulation
Upgrade Decision Support	Reactive	Proactive
Visualization	Rare or nonexistent	Integrated chart and report generation
User Interaction	Command-line or none	GUI-based (Tkinter)
Flexibility	Low	High – supports custom device counts
Risk Modeling	Basic or absent	Rule-driven vulnerability scoring

III. OBJECTIVES

The primary goal of this research is to develop a robust, simulation-driven framework that can proactively assess and recommend upgrade strategies for IoT devices based on their current hardware and software states. Given the increasing number of security breaches originating from outdated or poorly maintained IoT infrastructure, this framework aims to streamline the maintenance process through automation, statistical reasoning, and user-friendly visualization tools.

The following are the study's particular goals:

1) ***Objective 1: To Detect Outdated Hardware Configurations***

Evaluate the age, usage patterns, and performance degradation of hardware components in IoT devices to identify those at risk of failure or incompatibility with modern software updates.

2) ***Objective 2: To Identify Software Vulnerabilities Using CVE Data***

Integrate the Common Vulnerabilities and Exposures (CVE) database to recognize known security flaws in device firmware and software components. Use severity scores (e.g., CVSS) to prioritize mitigation.

3) ***Objective3: To Assess Patch Compliance and Update Effectiveness***

Determine whether devices are running the latest security patches and if patch deployment practices align with best security standards. This includes identifying patch gaps and their potential impact.

4)Objective4: To Visualize Upgrade Needs Through Interactive Dashboards

Provide clear, graphical representations of upgrade requirements using Matplotlib to assist administrators in making informed, data-driven decisions.

5) Objective 5: To Develop a Scalable and Educational Simulation Tool

Build a modular and extensible Python application that simulates varied IoT device environments and serves as a learning platform for researchers, educators, and students in cybersecurity and systems management.

IV. METHODOLOGY

To achieve the above objectives, a structured and systematic methodology is adopted, combining simulation, rule-based analytics, and visual interface development. The key steps in the methodology are outlined below:

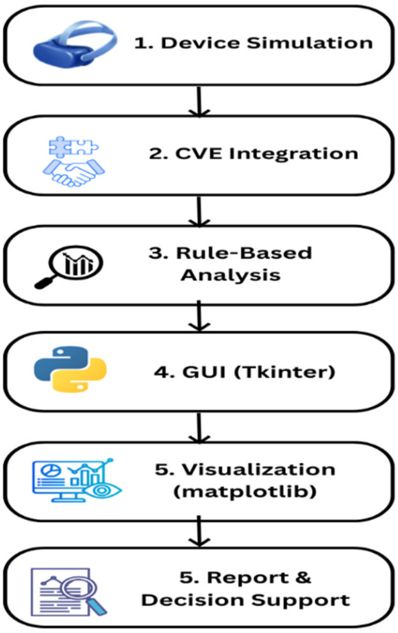

Fig: 1. Flow Diagram

A. Data Simulation and Device Modelling

A Python-based simulation engine is developed to model a wide range of IoT devices. Each device profile is generated with attributes such as:

- Hardware specifications (e.g., processor type, memory size, age)
- Software details (e.g., OS version, installed packages)
- Patch history and update timestamp
- Exposure level to internet-based threats

This simulated dataset mimics real-world IoT deployment scenarios, including variability in configurations and compliance.

B. Vulnerability Assessment Using CVE Integration

The system queries and integrates public vulnerability databases (such as NIST's NVD or MITRE CVE) to assess the security posture of each simulated device. Software versions are compared against known CVEs, and severity scores are used to classify devices as high-, medium-, or low-risk.

C. Rule-Based Statistical Analysis

A set of heuristic and statistical rules is applied to evaluate:

- Hardware end-of-life status
- Firmware/software version obsolescence
- Patch lag (i.e., how many versions behind a device is)
- Vulnerability exposure index (based on CVE scores)

The outcome of this analysis determines the urgency of upgrades, enabling prioritization based on objective risk metrics.

D. GUI Development with Tkinter

To ensure user accessibility, a GUI is implemented using Python's Tkinter library. The interface allows users to:

- Initiate simulations
- View device-specific analytics
- Configure simulation parameters
- Trigger vulnerability assessments

The GUI ensures the system can be used by both technical experts and administrators without programming knowledge.

E. Data Visualization Using Matplotlib

Visual output is generated using Matplotlib to display:

- Vulnerability distribution across devices
- Upgrade urgency heatmaps
- Patch compliance trends over time
- Summary dashboards showing critical, moderate, and low-priority devices

This visual component supports informed decision-making and improves transparency in maintenance strategies.

F. Evaluation and Use Case Testing

The framework is evaluated using simulated scenarios representative of educational, enterprise, and industrial settings. Metrics for evaluation include:

- Detection accuracy of vulnerable devices
- GUI usability (based on test-user feedback)
- Performance (execution time for large-scale simulations)

V. IMPLEMENTATION

In order to put the suggested approach into practice, a Python-based application that automates the analysis of device upgrade requirements must be designed and developed. The key steps in the process are listed below:

1) Data Simulation: A Python script creates a dataset that represents a fleet of devices. To replicate a real-world IT system, each device is randomly allocated hardware, software, and patch status.

2) Analysis and Classification: After processing the produced data, the system groups devices according to predetermined criteria, including software flaws, hardware obsolescence, and missing updates. Setting priorities for upgrade actions is made easier by this classification.

3) Visualization and Reporting: To help decision-makers see which areas need urgent attention, a bar chart depiction of upgrade statistics is made using Matplotlib.

The implementation provides a scalable and automated approach to upgrade analysis, minimizing manual effort and enhancing cybersecurity compliance. This approach can be further improved by integrating real-time monitoring systems and machine learning models for predictive maintenance.

VI. RESULTS & ANALYSIS

Devices that need hardware upgrades, those with software flaws, and those without the required updates are the three main categories into which the system successfully divides devices. The findings show that a sizable portion of devices need urgent care in order to preserve performance and security. Older hardware versions (such as HW_v1 and HW_v3) are common and contribute to the majority of upgrade demands, according to the dataset analysis. Furthermore, certain versions of the program—SW_v1.0 and SW_v2.0 in particular—have known CVE risks and are concentrated in software vulnerabilities. Nearly half of the devices had patch compliance problems, indicating a general difficulty in guaranteeing appropriate security updates. The distribution of upgrading requirements is clearly shown by the bar chart depiction, which helps decision-makers effectively prioritize interventions.This automated analysis helps organizations proactively address security risks, minimize system downtime, and optimize resource allocation for device maintenance.

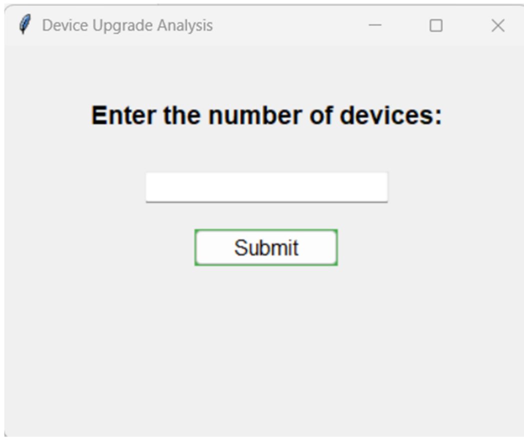

Fig. 2. input interface for entering the number of devices.

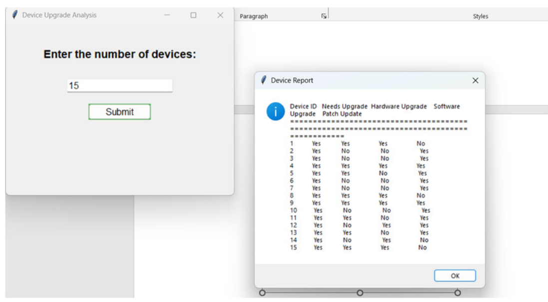

Fig. 3. Report popup displaying upgrade requirements per device.

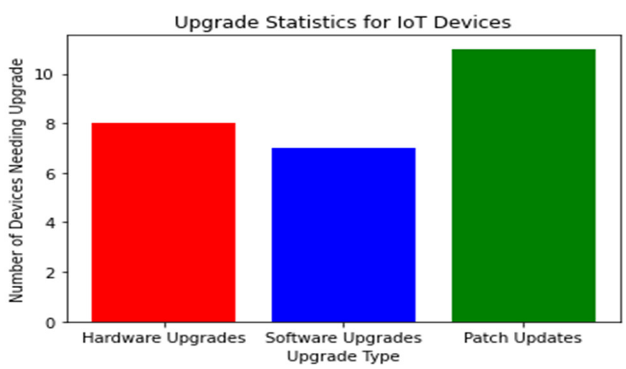

Fig. 4. Bar chart summarizing required upgrades by type.

VII. ADVANTAGES, LIMITATIONS AND FUTURE SCOPE

A. ADVANTAGES

1) ***Automated Analysis:*** Reduces manual effort in identifying upgrade requirements.
2) ***Enhanced Security:*** Addresses software vulnerabilities and outdated hardware.
3) ***Data-Driven Decision Making:*** Provides a clear visualization of upgrade needs.
4) ***Scalability:*** Can be applied to large-scale device fleets.

B. LIMITATIONS

Real-world device environments might not be accurately represented by simulated data. Does not integrate real-time sensor data for hardware monitoring. Limited software vulnerability assessment based on predefined CVE data.

C. FUTURE SCOPE

- Integration with IoT-based real-time monitoring systems.
- Implementation of machine learning models for predictive maintenance.
- Expansion to multi-platform environments (e.g., Linux, Windows, IoT devices).
- Advanced cybersecurity analytics to improve risk assessment.
- Integration of real-world CVE data.
- Dynamic threat scoring and policy updates.
- Export functionality (CSV, PDF).
- Dashboard-based GUI extensions.

VIII. CONCLUSION

This research presents a comprehensive and automated framework for analyzing and managing upgrade requirements in Internet of Things (IoT) devices. By leveraging simulated datasets, rule-based statistical analysis, and integration with known vulnerability databases such as CVE, the system effectively identifies outdated hardware and vulnerable software components across diverse device configurations. The use of a graphical user interface (GUI) built with Tkinter and data visualizations powered by Matplotlib ensures that both technical and non-technical stakeholders can easily interpret the results and make informed decisions regarding security upgrades.

Key drawbacks of current device maintenance procedures are addressed by the suggested method, such as their reliance on manual assessments, lack of organized vulnerability evaluation, and lack of visual decision assistance tools. The system functions as a scalable tool for researchers, educators, cybersecurity analysts, and IT managers by mimicking real-world device settings and mapping them to known security concerns. Additionally, its modular design allows for flexibility for a variety of use scenarios, ranging from enterprise-level security auditing to instructional presentations. Future developments will try to include real-time monitoring features, which will enable ongoing tracking of device conditions and vulnerabilities as they change. Furthermore, the system will be able to switch from reactive upgrade recommendations to predictive maintenance methods through the integration of machine learning models. These developments will improve the system's capacity to prevent downtime, protect IoT ecosystems proactively, and maintain regulatory compliance in ever-more complex digital infrastructures.

REFERENCES

[1] K. Rose, S. Eldridge, and L. Chapin, "The Internet of Things: An Overview," *Internet Society*, 2015.
[2] D. Evans, "The Internet of Things: How the Next Evolution of the Internet Is Changing Everything," *Cisco White Paper*, 2011.
[3] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, "Security, privacy and trust in Internet of Things: The road ahead," *Computer Networks*, vol. 76, pp. 146-164, 2015.
[4] National Vulnerability Database. [Online]. Available: https://nvd.nist.gov
[5] A. Mosenia and N. K. Jha, "A comprehensive study of security of Internet-of-Things," *IEEE Trans. Emerging Topics Comput.*, vol. 5, no. 4,

pp. 586–602, Oct.–Dec. 2017.
[6] H. Habibzadeh et al., "A survey of healthcare Internet of Things (HIoT): A clinical perspective," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 53-71, Jan. 2020.
[7] A. Ouaddah, H. Mousannif, A. A. Elkalam, and A. A. Ouahman, "Access control in the Internet of Things: Big challenges and new opportunities," *Computer Networks*, vol. 112, pp. 237-262, 2017.
[8] Roman, R., Najera, P., & Lopez, J. (2011). Securing the Internet of Things. Computer, 44(9), 51–58.
[9] Sicari, S., Rizzardi, A., Grieco, L. A., & Coen-Porisini, A. (2015). Security, privacy and trust in Internet of Things: The road ahead. Computer Networks, 76, 146–164.
[10] Mosenia, A., & Jha, N. K. (2017). A comprehensive study of security of Internet-of-Things. IEEE Transactions on Emerging Topics in Computing, 5(4), 586–602.
[11] Zhang, Y., Deng, R. H., & Weng, J. (2019). A lightweight attribute-based encryption scheme for smart grid. Future Generation Computer Systems, 78, 924–932.
[12] Weber, R. H. (2010). Internet of Things–New security and privacy challenges. Computer Law & Security Review, 26(1), 23–30.
[13] Hossain, M. S., Fotouhi, M., & Hasan, R. (2019). Towards an analysis of security issues, challenges, and open problems in the Internet of Things. Proceedings of the 2019 IEEE World Congress on Services (SERVICES), 264–269.
[19] Lee, I., & Lee, K. (2015). The Internet of Things (IoT): Applications, investments, and challenges for enterprises. *Business Horizons*, 58(4), 431–440.
[20] Python Software Foundation. Python Language Reference, version 3.x. Available at: https://www.python.org
[21] Tkinter Documentation: https://docs.python.org/3/library/tkinter.html
[22]Matplotlib Documentation: https://matplotlib.org/