OPEN ACCESS

Saudi Journal of Engineering and Technology

Abbreviated Key Title: Saudi J Eng Technol
ISSN 2415-6272 (Print) |ISSN 2415-6264 (Online)
Scholars Middle East Publishers, Dubai, United Arab Emirates
Journal homepage: https://saudijournals.com

Original Research Article

Introduction to Firmware Reverse Engineering for IoT Devices Using Ghidra and Binwalk

Areeba Kouser[1]*, Muhammad Siddique[2], Abiha Abbas[3]

[1]School of System & Technology, University of Management and Technology, C1 Road, sector A, LSC, Punjab, Pakistan
[2]School of System & Technology, University of Management and Technology, Lahore, Pakistan
[3]School of System & Technology, University of Management and Technology, Lahore, Pakistan

DOI: https://doi.org/10.36348/sjet.2026.v11i05.007 | **Received:** 11.02.2026 | **Accepted:** 06.04.2026 | **Published:** 14.05.2026

***Corresponding author:** Areeba Kouser
School of System & Technology, University of Management and Technology, C1 Road, sector A, LSC, Punjab, Pakistan

Abstract

The fast usage of Internet of Things (IoT) device in industrial and consumer settings has dramatically expanded on the attack surface of embedded systems. This paper explores firmware security through reverse engineering and analysis of a firmware image of an IoT style with two open-source tools: Binwalk and Ghidra. An artificial representation of the structure of typical Linux-based IoT firmware was produced by a controlled firmware image which had a SquashFS file system and compiled binaries. Embedded file systems and binaries were extracted using Binwalk and Ghidra was used to do the static analysis and decompilation of extracted executable files. The vulnerability analysis showed that there are a number of deliberately introduced security flaws such as hard-coded credentials, unsecured input handling functions and insecure configuration practices. The success of the method was shown by the successful recovery of the firmware filesystem and detection of these types of vulnerabilities with the help of the strict reverse-engineering tool. The paper shows the possible contribution of open-source tools to the analysis of firmware-level vulnerabilities and enhancing security testing of embedded IoT systems.

Keyword: Reverse Engineering Firmware, IoT, Binwalk, Ghidra, Embedded Systems.

1. INTRODUCTION

Internet of Things (IoT) has grown at an alarming rate in the last decade and is currently applied in industrial systems, smart homes, and electronics that consumers use. Nonetheless, there remain few advances in the creation of holistic security systems in the IoT environments. Several research works have assessed security risks at network and application level, whereas vulnerabilities at firmware level have been under-researched. Firmware forms the fundamental layer of software that governs behaviour of devices and any vulnerabilities at this point can enable attackers to have lasting control over devices. Thus, the main gap covered in the study is the absence of effective and systematic approaches to the analysis of security vulnerabilities in the firmware of the IoT device.

This research focuses on firmware reverse engineering using two widely adopted open-source tools; Binwalk and Ghidra. Binwalk facilitates the extraction and identification of embedded file systems and binaries within firmware images. Although Binwalk allows extracting embedded file systems and binaries, additional static analysis will be needed to determine the inner logic of compiled programs and realize possible security bugs in the extracted firmware pieces. Ghidra provides advanced static analysis capabilities for understanding binary code. Figure 1 presents a systematic approach to the study of IoT security, which begins with automated file system extraction by Binwalk and continues with vulnerability detection on the high level. The steps in the layers in Figure 1 depict the steps involved in the firmware analysis: acquisition of the firmware itself by downloading it, extraction of the file system by Binwalk, binary analysis, exploration of the vulnerabilities by means of the static reverse engineering with Ghidra. In the offered approach to investigating the embedded firmware security, every layer relates to a phase. This notes the shift in general automated scanning to more comprehensive manual testing with Ghidra, in which compiled binaries are decompiled and reformed to

reveal key vulnerabilities over the form of hardcoded credentials or bugs.

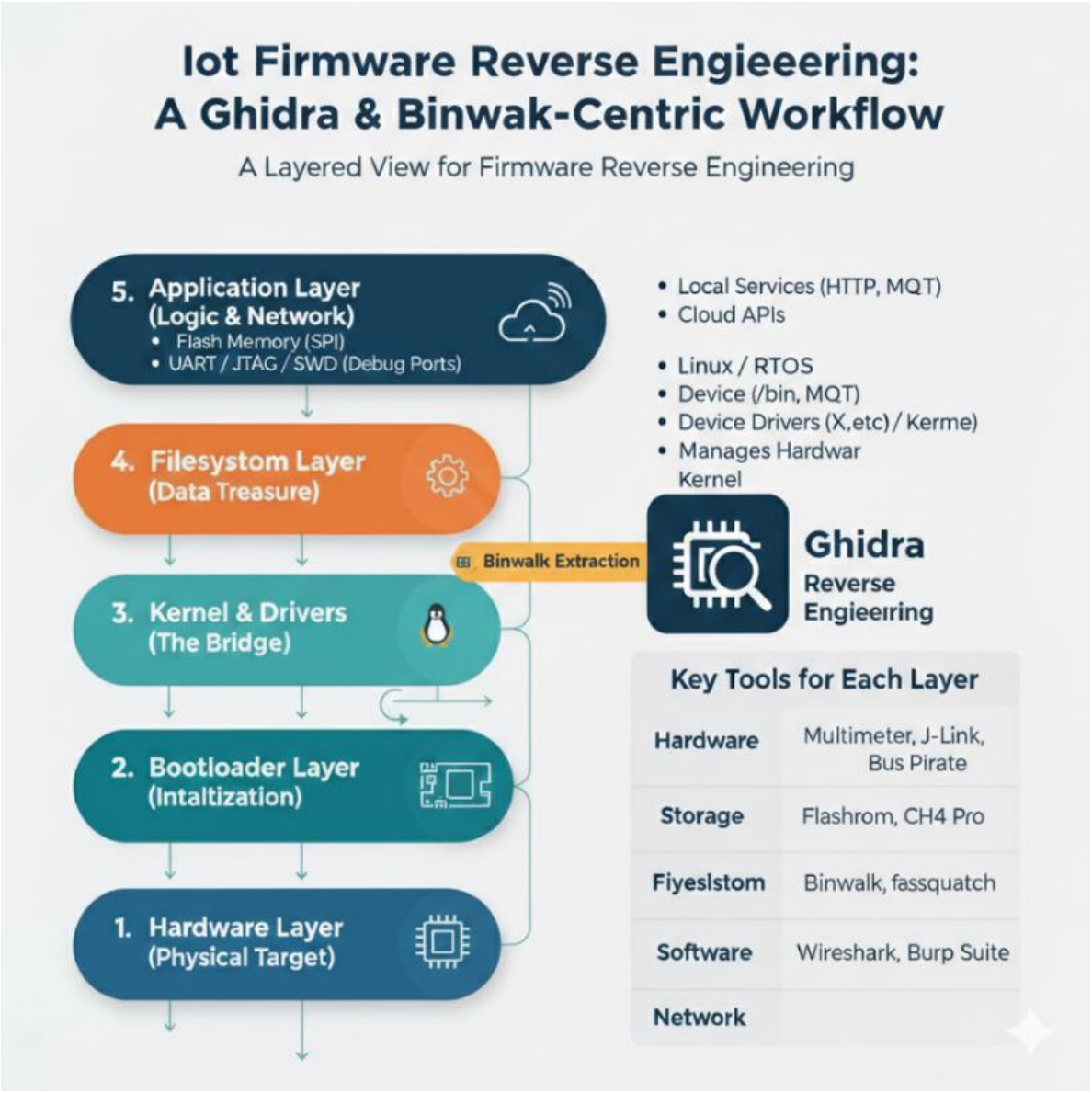

Figure 1: A Layered View for Firmware Reverse Engineering

The research paper of T. Bakhshi, B. Ghita, and I. Kuzminykh. 2 presents a taxonomy of vulnerabilities and associated them with its exploitation vectors and with the auditing tools that may aid in their identification. The goal of the study conducted by K. Kaushik et al. 6 was to introduce the techniques and methodologies of studying and reverse-engineering firmware, and threats to firmware vulnerabilities. Although these papers offer theoretical backgrounds and taxonomies to firmware vulnerabilities, the current work is different because it illustrates an experimental workflow utilizing the open-source tools to extract firmware and revert traditionally to a controlled experimental setting.

In the industrial facilities, IoT sensors and data analytics can be used to track energy consumption trends, determine areas of inefficiency and optimize energy consumption. For efficient power consumption, smart grids provide the real-time data about power requirements, cost and its availability. A use case was defined by A. R. Nygård, A. Sharma, and S. Katsikas 12. They used an IED often used in smart grid's digital substations. In a research by Y. G. Hassan et al. 16, they focused on filling the current gaps and to develop a protected and strong IoT ecosystem by providing reliable solutions and collective efforts.

To provide remote enhancement ideal for devices having limited storage capacity, essential memory resources are required for reforming application-based firmware. The paper by Z. Zheng, X. Cai, S. Wang, and Y. Yang 17 proposed a secure, efficient, and universal remote firmware upgrade solution. Y. O. D. A. Minami et al. 19 applied their

Approach to 226 real product binaries. Therefore, they discovered two running programs that were plainly transmitting IDs and passwords.

There is a lack of structured and practice-oriented approaches for analyzing such firmware in the context of digital forensics. T. Kruger 7 developed a systematic, five-phase analysis model for Linux-based open embedded firmware, addressing both technical and forensic requirements. The work of C. R. Barone IV et al. 5 presented findings of a needs analysis survey of Reverse Engineering (RE). In response to the question on whether there is a lack of sufficient candidates on RE, most 33/71 (46.48) said strongly agree and 20/71 (28.17) said agree somewhat.

The growing globalization of the Internet of Things (IoT) supply chain that has taken place in the last few decades has led to important security issues. R. A workflow was suggested by Tsang, D. Joseph, Q. Wu, S. Salehi, N. Carreon, P. Mohapatra, and H. Homayoun [3] to reverse engineer the binary, with the assistance of Ghidra and Python scripting, and offer two simples, yet novel, function matching algorithms.

2. LITERATURE REVIEW

One of the most potent tools of modern-day cybersecurity is Reverse Engineering, which allows identifying the vulnerabilities, analyzing malicious code and improving the defense measures. The work by Cisar et al. 15 focuses primarily on dynamic behavior monitoring during program execution, whereas other studies emphasize static firmware analysis and reverse engineering techniques to identify vulnerabilities directly from binary code. The study conducted by I. Srihith *et al.,* 17 discussed the methods how attackers can exploit vulnerabilities in firmware and how this will affect the security and privacy of the users. They also made the recommendations to protect against firmware attacks, such as to maintain firmware in good, and using a strong password together with checking unusual activities occurring in the network.

The absence of the full-scale survey on the security of firmware in the IoT is supposed to emphasize the significant motivations of a firmware insecurity in the IoT, counting the vulnerabilities, and conducting a detailed examination of the most significant methods of analysis. The article of C. D. Dinh [15] included some details of the device firmware initiation by making the point of the significance of the firmware security, classifying the vulnerabilities in IoT, the processes of analyzing the firmware vulnerabilities through some Open-source tools, and as illustrated by analyzing some firmware's of the popular devices. J. Liu et al. 9 applied IotSleuth and assessed it across 117 IoT firmware samples of nine vendors. The experimental findings demonstrated that IotSleuth identified 27 vulnerabilities. But these studies are mostly related to the analysis of existing dataset of firmware and automatic detection tools but they are not tests of a controlled experimental process of constructing and analysing firmware images to be educational and research.

The ecosystem of non-Linux firmware is not well known, with the majority of current literature being dedicated to Linux based firmware. A. Balgavy and M. Muench [10] studied 21,755 samples, which were acquired in earlier studies and in new ones. They also as a component of a security measurement surveyed the existence of operating systems and memory protections on a subsample of 756 non-Linux ARM samples and discovered that most do not utilize either. The study of D. Tauscher, M. Nowatkowski, J. D. Morris, and D. Baldwin 10 provided the discussion and comparison of some of the chosen medical devices that will simulate a working wireless network, transmitting between patient and host. Findings made it possible to identify and detect attack vectors or device fragilities and categorically classify them into a framework similar to the Purdue/TCP/IP models-targeting Medical Internet of Things (MIoT).

RE may be conducted at chip level or at the printed circuit board (PCB) level and different methods of analysis can be employed, such as structural, functional, and combined analysis. The research of C. Vega, P. Slpsk, and S. Bhunia 12 noted that one of the factors contributing to the success of such attacks at both chip and board levels is that an attacker is able to monitor the input/output (I/O) behavior of an operational system.O. Komolafe et al. 11 studied reverse engineering techniques, its scope in different industries, the difficulties occurring during the process and the its prospects for future.

Modern tools like IDA Pro and Ghidra are used for examining the code, revealing persistence mechanisms and controlling communications. The research work of R. Tamilkodi, V. B. Sankar, S. Madhu, L. Revathi, V. S. G. Srikar, and E. Ajay 1 highlights the need for strong security practices and active vulnerability management in IoT device design and deployment, for strengthening IoT ecosystems against threats. I. Nadir, H. Mahmood, and G. Asadullah 3 suggested several important research problems as an incentive and an enabler of research in the field of firmware security of IoT. X. A, C. Yan, Y. Wang, Q. Wei, and Y. Wang [8] suggested a lightweight vulnerability scanning method, WFinder that is specifically crafted to suit embedded firmware web services to apply vulnerability checks to binary firmware backend files.

Table 1: Comparison Table of Related Work

Ref.	Year	Focus	Techniques/Methods	Findings	Journal
15	2025	Reverse Engineering in Cybersecurity	Analysis of binary data, disassembly of code and behavior monitoring	Highlighted applications in IoT devices, traditional software, and mobile apps for securing systems	Critical Infrastructure Protection: Future Technologies in Crisis Prevention and Response.
17	2023	Firmware attacks	Analysis of attack methods	Discussed firmware vulnerabilities, impact on security/privacy, and recommended mitigation like updates, strong passwords, and network monitoring	International Journal Advance Research Science Communication and Technology (IJARSCT).
13	2025	IOT firmware security	Open-source firmware analysis tools	Emphasized firmware security importance, classified vulnerabilities, analyzed popular device firmwares	Journal of Technical Education Science
9	2025	IOT firmware analysis	IoT Sleuth tool	Evaluated 117 firmware samples; discovered 27 new vulnerabilities	Internet of Things (Elsevier)
9.A	2024	Non linux firmware analysis	Security assessment	Analyzed 21,755 samples; non-Linux firmware often lacks OS/memory protections	Workshop on Binary Analysis Research (BAR)
10	2024	Medical IOT devices	Functional wireless network modeling, attack vector identification	Compared medical devices, identified vulnerabilities, mapped to Purdue/TCP-IP models	IEEE SoutheastCon 2024
11	2023	Reverse Engineering techniques	Review of RE methods and applications	Highlighted RE techniques, industrial applications, challenges, and future opportunities	Survey / Review Article
1	2024	IOT malware and security	Advanced tools like IDA pro, Ghidra	Emphasized proactive security practices, malware tracing, and resilience recommendations for IoT ecosystems	EEE ICDICI 2024
3	2023	IOT firmware security research	Literature survey/gap analysis	Identified key research issues to promote firmware security research	Research Survey
8	2024	Embedded firmware web services	WFinder vulnerability scanner	WFinder vulnerability scanner	Applied Sciences (MDPI)

3. METHODOLOGY

In this section, the experiment methodology that will be employed to prove the reverse engineering of firmware with IoT devices will be given. The workflow aims at creating a representative firmware image, retrieving its internal file system, and finding possible vulnerabilities in the form of vulnerable binary code with the help of open-source tools.

3.1 Work Flow

The workflow includes creation of C file, compilation into binary, creation of root filesystem, squashFS image creation, binwalk scan and reverse engineering as shown in the figure. SquashFS was chosen as it is a common embedded Linux system and internet of things firmware since it is compressed and is read only form, thus the thought was it makes SquashFS a realistic example of firmware deployed in commercial devices.

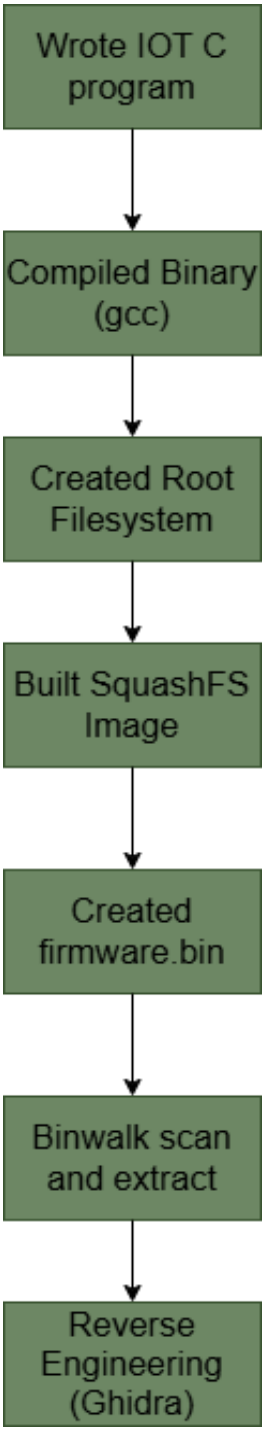

Figure 2: Flow Chart of the proposed methodology

3.2 Practical Work

The firmware image adopted in this research has been artificially created to be used in experiments and security vulnerabilities have been purposefully introduced in the sample program to show the way that can be used by reverse engineering tools to identify them.

3.2.1 Environmental Setup

All tests were performed on Kali Linux that has a broad variety of security and reverse engineering tools. The following tools were used,

- Binwalk for analyzing and pulling out firmware
- Ghidra for examining binary code without running it
- GCC for turning source code into firmware binaries
- SquashFS tools for making a file system

This setup is similar to what security experts use when they study IoT firmware.

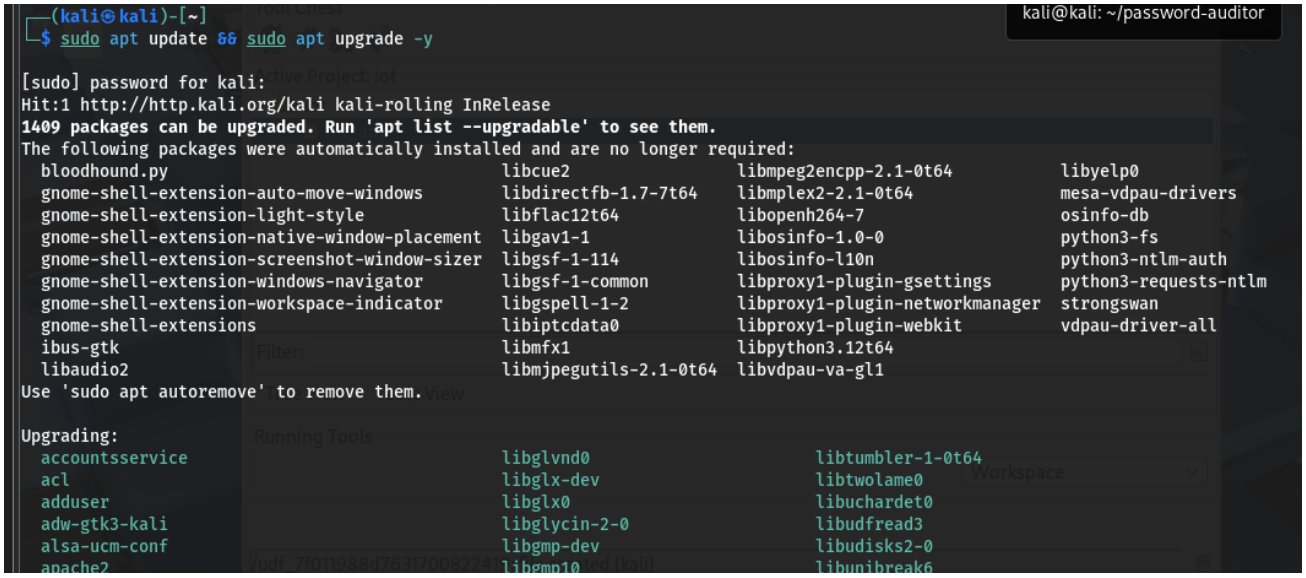

Figure 3: Environmental Setup

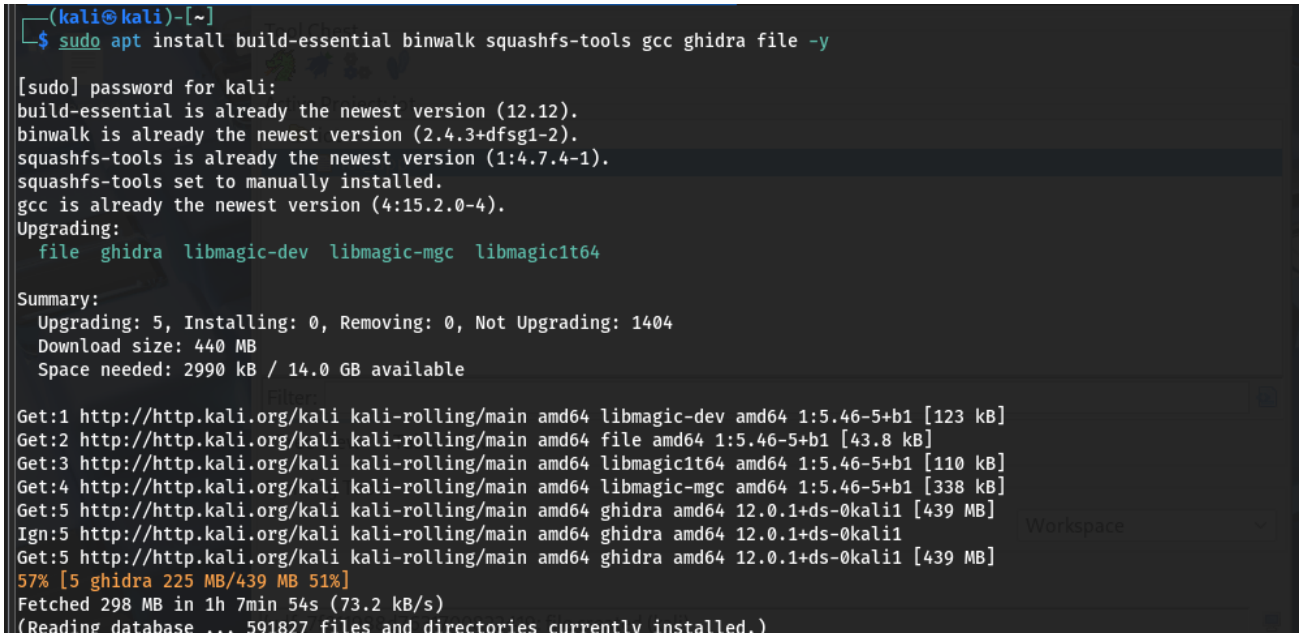

Figure 4: Installation of required tools

3.2.2 Creation of a Sample Program (Target binary)

The program deliberately contained a number of general IoT firmware attacks such as:

- Hard coded usernames and passwords.
- Unsafe input Unsafe input functions like gets and strcpy.
- Plain-text password storage.
- Lack of input validation.

Figure 5: Program file creation

Figure 6: Code written in the file.

3.2.3 Compilation of Binary (IOT-style)

GCC was used to complile the code file.

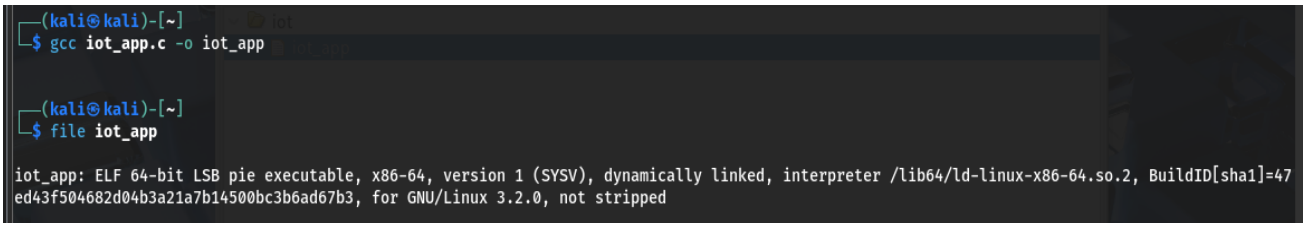

Figure 7: File compilation process

3.2.4 Creation of Root filesystem

File with root privileged is illustrated in the figure.

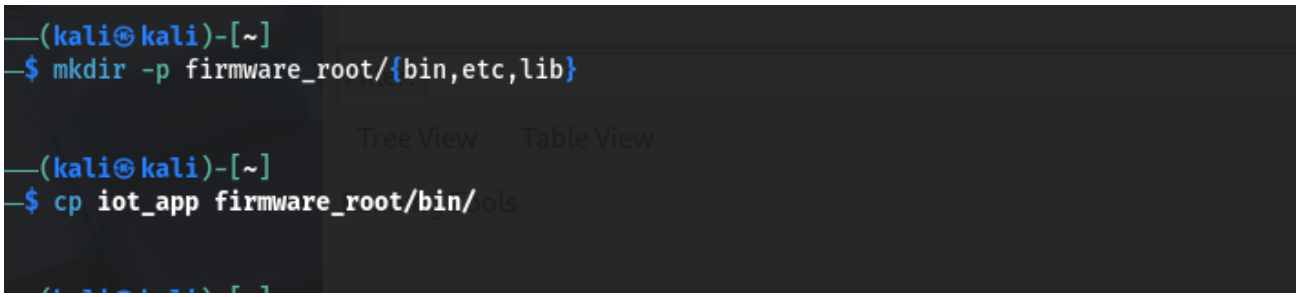

Figure 8: File with root privileged

3.2.5 Conversion of Filesystem into SquashFS

A squashFS system was created and then converted into firmware file for reverse engineering process.

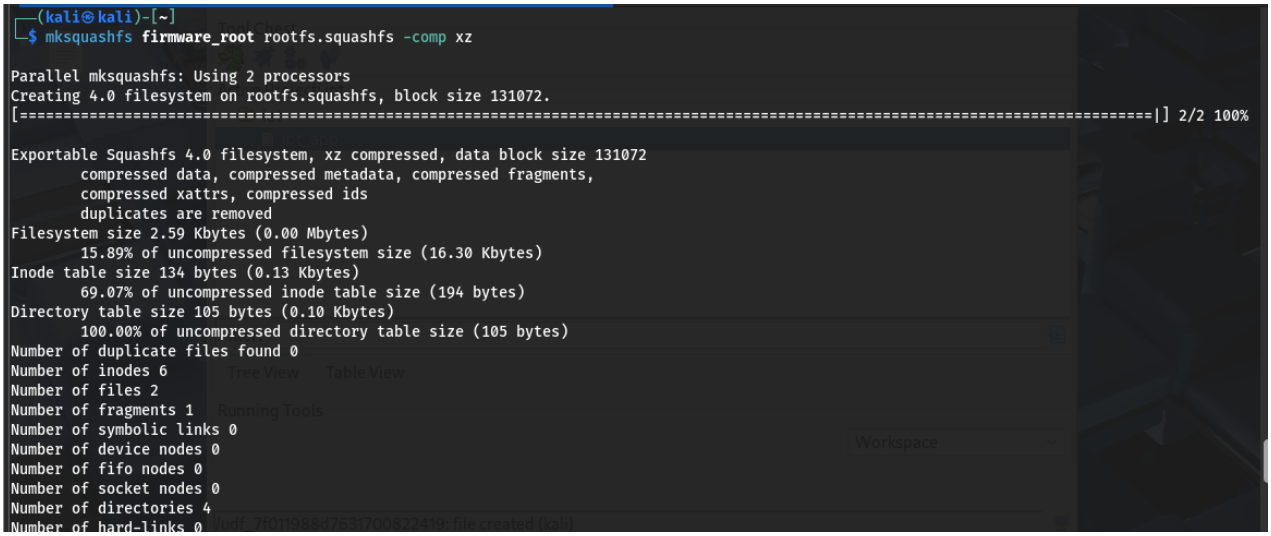

Figure 9: Squash FS format

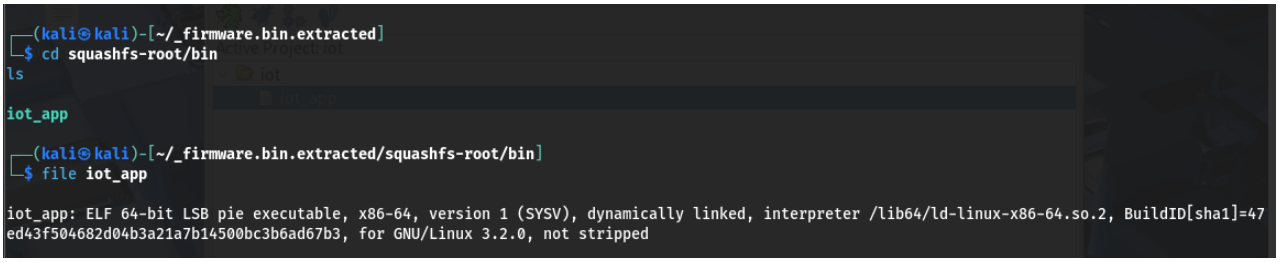

Figure 10: Changes shown in figure

3.2.6 Analysis of Firmware using Binwalk

Binwalk scanned the file for hidden file systems and different types of compression. When it found something, it automatically pulled out the contents. This process showed the layout of the internal file system and allowed access to the binaries and configuration files that are usually hidden inside IoT firmware. It must be mentioned that Binwalk might prove ineffective in the analysis of encrypted, obfuscated, or encrypted with proprietary compression systems, firmware images.

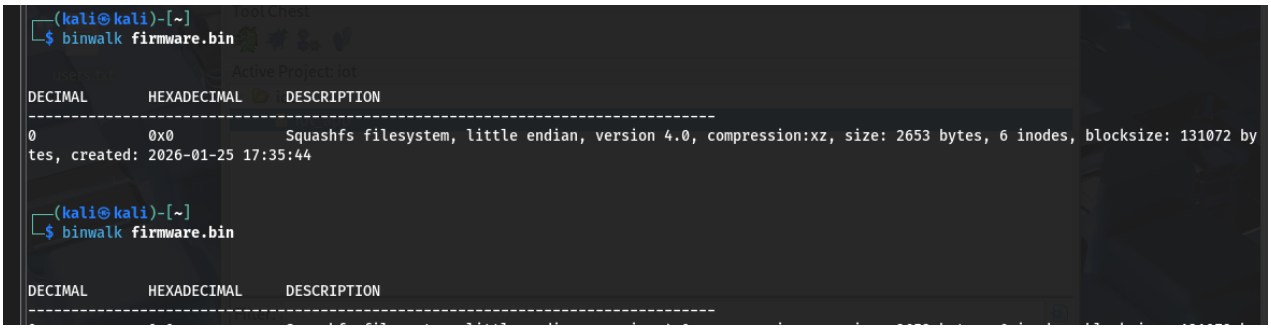

Figure 11: Analysis using binwalk

3.2.7 Reverse Engineering using Ghidra

Once the files were pulled out, the executable programs were picked for closer inspection. The files were then imported into Ghidra, which automatically identified the processor type. Ghidra's tools were used to make flowcharts, find functions, and convert machine code into a readable C-like format. The Ghidra static analysis revealed that the program had certain security vulnerabilities such as hardcoded authentication credentials, unsafe input handling functions that result in a buffer overflow vulnerability, and sensitive information is exposed in binary strings. Such weaknesses are used to illustrate generic implementation vulnerabilities that can be present in poorly secured IoT firmware.

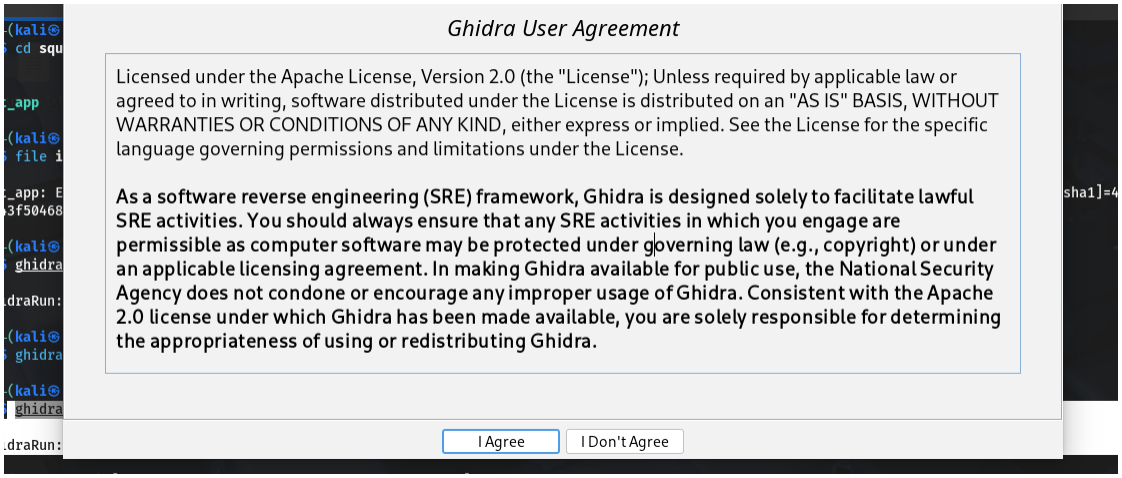

Figure 12: Ghidra setup

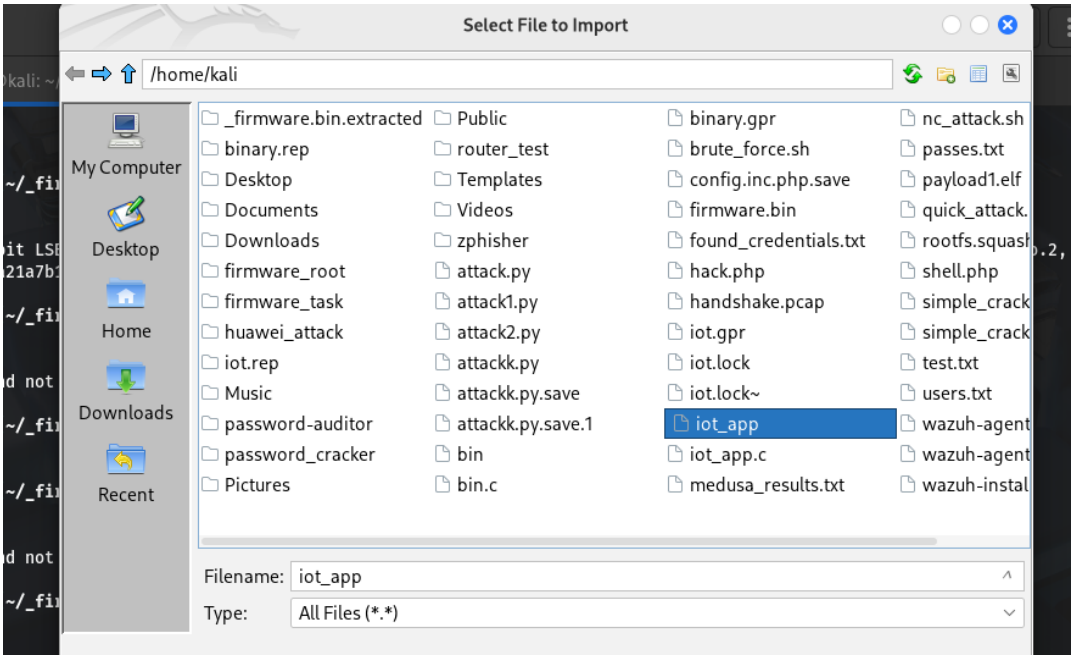

Figure 13: File selection for analysis

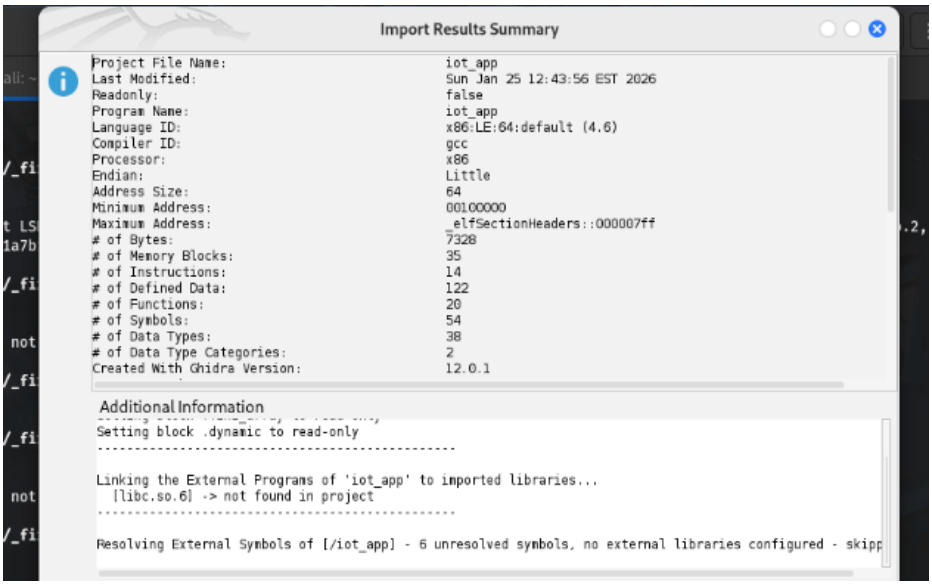

Figure 14: Summary of file

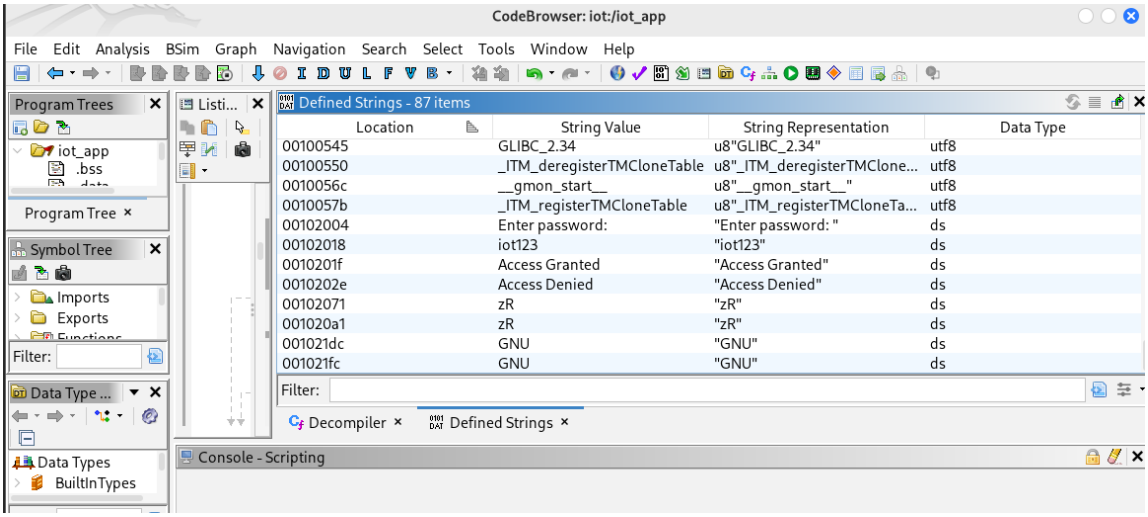

Figure 15: String view of file

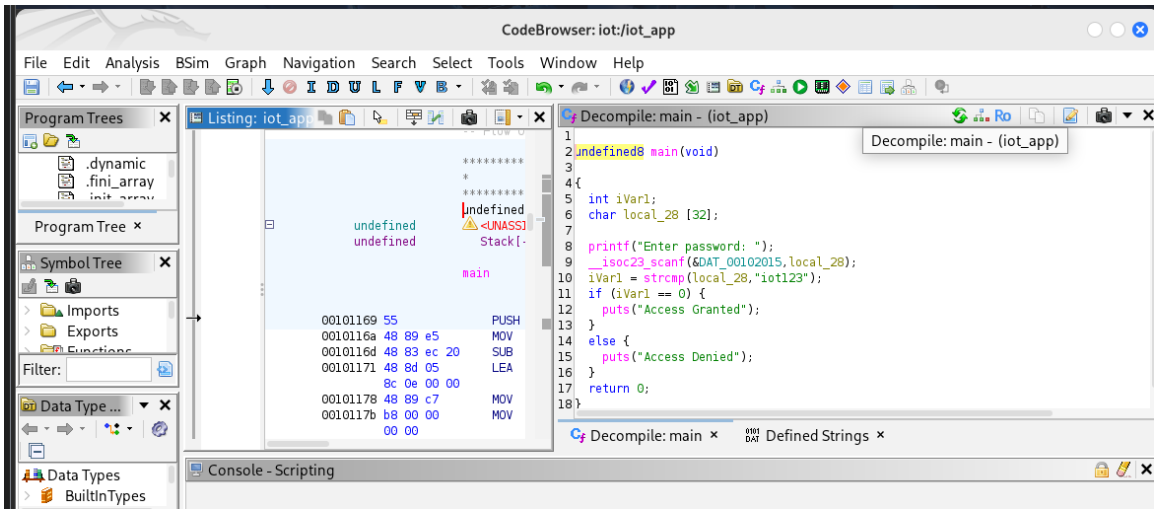

Figure 16: Code view of file

Special attention was given to:

- How the device checks if someone is allowed to access it
- Passwords that are written directly into the code
- Use of unsafe functions
- How commands are executed

This helped find possible security weaknesses in the firmware.

4. ANALYSIS AND FINDINGS

The firmware image made for this study was analyzed using Binwalk and Ghidra. Binwalk showed there was an embedded SquashFS filesystem, which was automatically extracted to show the internal structure of the firmware. We found .exe files and configuration files which were stored in common linux directories. Credential details and cause of a user login were exposed in form of plain text entries, when strings were analyzed.

The disassembler of Ghidra showed C type code which was easy to read in which some fixed credentials were shown, used by system for authentication. An input handling functions was also found which was unsafe because it turned out be vulnerable to buffer overflow. All these examinations and results highlighted some common security issues which can be easily exposed by using firmware reverse engineering.

Because the firmware image was created with a purposely made vulnerability, the existence of insecure code practices including hardcoded credentials did not come out of the blue. Nonetheless, the analysis also showed how these vulnerabilities can also be systematically located with the help of conventional reverse engineering methods.

5. DISCUSSION

The results of this study highlighted the importance of reverse engineering of firmware for IoT devices security check. Hardcoded credentials and poor coding techniques showed the common weaknesses in unsecure IoT devices. By creating a firmware instead of using an already available firmware was essential for maintain the ethical work flow.

The tools like Binwalk and Ghidra were used together, and gave a high-level view of firmware. Overall, the importance of secure coding practices and avoiding inclusion of sensitive information, were found in our work.

The results of this research complement the earlier studies regarding the analysis of the vulnerability of firmware. As an example, Liu et al. 9 have shown the automated vulnerability detection of an IoT firmware with the use of an ion Sleuth tool, whereas Dinh 13has discussed the necessity of firmware security testing with the help of open-source software. Unlike these works, the present research is based on illustrating a practical experimental process that may be implemented in the academic setting in order to learn techniques of firmware analysis.

6. CONCLUSION

All the tools and techniques used in our research work were ethically handled and implemented. Even the firmware image was created in linux system manually. Ghidra and Binwalk were proved to be efficient tools to be used in linux. The results proved that using static firmware analysis technique is a strong technique for revealing security issues of IoT devices, even if the source code is not known. Yet, this study can be limited by the fact that the firmware image was created artificially and cannot reflect the complexity of proprietary firmware in commercial IoT devices in real-life situations fully.

7. Future Work

The analysis of real-world Internet of Things firmware images should be chosen as the priority of future research since it would give a better picture of vulnerabilities that exist in devices deployed. Moreover, by introducing dynamic methods of firmware analysis, including emulation and runtime analysis, vulnerabilities that are not apparent through the application of static analysis might be detected. Lastly, the analysis of firmware can further be automated with the help of AI-assisted vulnerability detection tools, and the accuracy of detection can also be enhanced.

REFERENCES

1. R. Tamilkodi, V. B. Sankar, S. Madhu, L. Revathi, V. S. G. Srikar, and E. Ajay, "Exploring IoT device vulnerabilities through malware analysis and reverse engineering," in *2024 5th International Conference on Data Intelligence and Cognitive Informatics (ICDICI)*, Nov. 2024, pp. 279–283.
2. T. Bakhshi, B. Ghita, and I. Kuzminykh, "A review of IoT firmware vulnerabilities and auditing techniques," *Sensors*, vol. 24, no. 2, p. 708, 2024.
3. R. Tsang, D. Joseph, Q. Wu, S. Salehi, N. Carreon, P. Mohapatra, and H. Homayoun, "FANDEMIC: Firmware attack construction and deployment on power management integrated circuit and impacts on IoT applications," in *NDSS*, Feb. 2022.
4. Nadir, H. Mahmood, and G. Asadullah, "A taxonomy of IoT firmware security and principal analysis techniques," 2023.
5. C. R. Barone IV, R. Serafin, I. Shavrov, I. Baggili, A. Ali-Gombe, G. G. Richard III, and A. Case, "A reverse engineering education needs analysis survey," *arXiv preprint arXiv:2212.07531*, 2022
6. K. Kaushik, A. Bhardwaj, and S. Dahiya, "Framework to analyze and exploit the smart home IoT firmware," *Measurement: Sensors*, vol. 37, p. 101406, 2025.
7. T. Kruger, "A systematic approach to digital forensics of embedded firmware," *SSRN*, 2025, [Online]. Available: https://ssrn.com/abstract=5371918.
8. X. A, C. Yan, Y. Wang, Q. Wei, and Y. Wang, "A vulnerability scanning method for web services in embedded firmware," *Applied Sciences*, vol. 14, no. 6, Art. no. 2373, 2024.
9. J. Liu, Y. Li, R. Yao, Y. Zhang, and H. Liang, "Detecting vulnerabilities in IoT firmware via keyword identification and path optimization," *Internet of Things*, p. 101808, 2025.
A. Balgavy and M. Muench, "Firmline: a generic pipeline for large-scale analysis of non-Linux firmware," in *Proc. Workshop on Binary Analysis Research (BAR)*, 2024.
10. D. Tauscher, M. Nowatkowski, J. D. Morris, and D. Baldwin, "Medical devices: Reverse engineering for innovation and advancement in healthcare," in *Proc. SoutheastCon 2024*, Mar. 2024, pp. 860–862.
11. O. Komolafe, I. T. Adejugbe, T. I. Olorunsola, J. A. Olowonubi, O. Oluwole, J. O. Aigbovbiosa, and O. A. Oyegunwa, "Reverse engineering: Techniques, applications, challenges, opportunities."
12. C. Vega, P. Slpsk, and S. Bhunia, "IOLock: An input/output locking scheme for protection against reverse engineering attacks," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 32, no. 2, pp. 347–360, 2023.
13. R. Nygård, A. Sharma, and S. Katsikas, "Hardware reverse engineering for secure smart grids," in *The Role of Cybersecurity in the Industry 5.0 Era*, Intech Open, 2024.
14. D. Dinh, "Embedded and IoT devices firmware security," *Journal of Technical Education Science*, vol. 20, no. 03, pp. 58–67, 2025.
15. S. M. Čisar, R. Pinter, and P. Čisar, "Reverse engineering and cyber security," in *Critical Infrastructure Protection: Advanced Technologies for Crisis Prevention and Response*, p. 99, 2025.
16. Y. G. Hassan, A. Collins, G. O. Babatunde, A. A. Alabi, and S. D. Mustapha, "Automated vulnerability detection and firmware hardening for industrial IoT devices," *International Journal of Multidisciplinary Research and Growth Evaluation*, vol. 4, no. 1, pp. 697–703, 2023.
17. Z. Zheng, X. Cai, S. Wang, and Y. Yang, "Research and design of IoT device firmware upgrade system based on BLE," in *Proc. 2nd Guangdong-Hong*

Kong-Macao Greater Bay Area International Conf. on Digital Economy and Artificial Intelligence, Mar. 2025, pp. 880–886.

18. Srihith, A. Donald, T. Srinivas, D. Anjali, and A. Chandana, "Firmware attacks: The silent threat to your IoT connected devices," *Int. J. Adv. Res. Sci. Commun. Technol. (IJARSCT)*, vol. 3, pp. 145–154, 2023.
19. Y. O. D. A. Minami, S. Sakuraba, S. E. I. Yuichi, Y. Tahara, and A. Ohsuga, "Detection of plaintext login information in firmware," in *2022 IEEE International Conf. on Consumer Electronics-Taiwan*, Jul. 2022, pp. 523.