JTE
HCMUTE
ISSN: 2615-9740

JOURNAL OF TECHNICAL EDUCATION SCIENCE
Ho Chi Minh City University of Technology and Education
Website: https://jte.edu.vn
Email: jte@hcmute.edu.vn

Embedded and IoT Devices Firmware Security

Cong Doan Dinh
Ho Chi Minh City University of Technology and Education, Vietnam

*Corresponding author. *Email: doandc@hcmute.edu.vn*

ARTICLE INFO

Received: 22/02/2025
Revised: 26/03/2025
Accepted: 04/04/2025
Published: 28/08/2025

KEYWORDS

Embedded security;
Internet of Things (IoT);
IoT security;
Firmware security;
Firmware analysis.

ABSTRACT

The Internet of Things (IoT) has gone a long way since its inception. However, the standardization process in IoT systems for IoT safety solutions is still in its early stages. Lots of studies have been conducted on its quality evaluation to address threats to IoT across different layers. However, most current works have failed to take into their consideration of the security aspects of the firmware in the IoT ecosystem. The lack of a comprehensive survey on IoT firmware security aims to highlight the important reasons for a firmware insecurity in IoT, list vulnerabilities, and perform an in-depth review of the key analysis techniques. Such rapid development has made IoT and embedded systems become interesting targets for potential attackers, especially the firmware attacks. This Article hereby presents some information related to the device firmware as started by highlighting the importance 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 firmwares of the popular devices.

Doi: https://doi.org/10.54644/jte.2025.1836

1. Introduction

Embedded systems consist of the dedicated computer hardware and software, often part of a larger system with limited resources [1]. The embedded systems can be found in devices such as washing machines, air conditioners, etc. Most IoT devices are the embedded systems with network connectivity such as IP cameras, smart locks, fitness trackers. The software of the dedicated computer that controls the embedded system is often called firmware and is stored in non-volatile memory. Many manufacturers store firmware in flash memory. Firmware images are provided to update the firmware of the device, which can be done automatically or manually

The modern embedded devices exist in large numbers in the global IT environment and critical communications infrastructure. The embedded systems such as routers, switches, and firewalls make up a large part of our global network infrastructure [2]. Most devices use the Linux-based firmware. Typical components of Linux firmware include: header, bootloader, kernel, file system, Figure 1. [3] shows that modern IoT devices today are quite similar to traditional computers as well as the components of firmware.

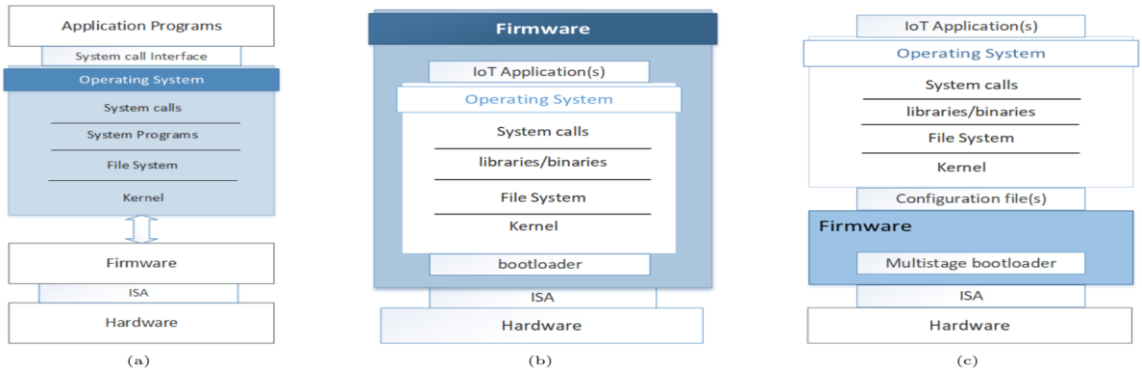

Figure 1.*(a) Traditional computers (b) Typical IoT devices (c) Full-resource IoT devices [3]*

You can obtain the device firmware through: the vendor's website, the device firmware update process, and hardware extraction. Firmware, like any other softwares, contains security vulnerabilities. Providing the security firmware is the responsibility of the device vendor [4]. There are many reports of individual device analysis indicating that the vulnerabilities have been found. However, this is a manual process and is not scalable. Today, there are a number of security analysis tools, but they focus on a single aspect: Firmware.

1.1. Researches in our country

According to [5], the survey was conducted on 1,000 leaders working in the security field of enterprises from China, Germany, Japan, the UK and the US. The survey results indicated that most security investments were for updates, vulnerability scanning and solutions to protect against information security risks. However, the information security risk from the firmware has become a notable issue when 80% of the enterprises were the victims of this attack.

The firmware is a type of computer software that provides low-level control over a specific piece of hardware on a device. It is becoming a target for hackers because it contains sensitive information such as authentication credentials and encryption keys. According to data from the US National Institute of Standards and Technology, firmware attacks have increased fivefold over the past four years. The lack of focused investments in firmware protection such as kernel data protection or memory encryption is one of the worrying realities. The report found that 36% of the enterprises invested in the hardware-based memory encryption and 46% in the hardware-based kernel protection.

In Vietnam, there are many research groups related to firmware security such as TECHPRO with "Vietnamese voice audio warning firmware on Suprema FSF2 and BS3" [6], Le Qui Don University of Technology with "Hardware safety and security: Methods, technology and applications" [7], Ministry of Information and Communications "Building standards for security camera equipment" [8].

1.2. Researches in the foreign countries

With more and more firmware vulnerabilities being discovered and detected, the governments as well as organizations and individuals around the world have conducted many studies on the firmware security, in addition to finding the vulnerabilities to warn manufacturers and deployers, they have also created frameworks as models for producing the security firmware such as OWASP, The Damn Vulnerable Router Firmware Project, Son et al. (2019), Thakur et al. (2019), Zhu et al. (2020) [9].

2. Some issues related to the embedded devices and firmware

Firmware of the embedded devices often contains serious security flaws that can be detected using analytical techniques and tools [10]. However, to apply the analytical and testing techniques, it is necessary to first identify, understand and overcome the challenges of the embedded devices.

2.1. Architecture

According to [4], firmware code is compiled for a hardware platform which refers to the CPU architecture and its instruction set. There are several architectures for the embedded devices due to the diversity of applications, users, and vendors. This is in contrast to the PC market where standards have been developed to unify hardware architectures and their interfaces. There are three most common architectures in the embedded and IoT devices today: ARM/ARM64, MIPS, x86/x86_64. Each architecture has a different organization, making the firmware security analysis become complicated.

2.2. Differences between the traditional firmware and embedded device firmware

Like the traditional computers [10], the embedded systems are controlled by software to perform specific tasks, called firmware, the firmware for the embedded systems differs from the traditional one in the following ways:

- Firmware directly interacts with the underlying hardware of the system to perform its tasks.

- Firmware is deeply integrated into the embedded system and is stored in read-only memory (ROM) or non-volatile memory such as flash memory or electrically erasable, programmable, read-only (EEPROM) chips.
- Data in the firmware can be hard-coded inside it to be used directly by the processor of the system.

Table 1 shows the differences between the traditional firmware and that of embedded devices or IoT devices

Table 1. *Difference between the traditional firmware and embedded device firmware [10]*

Feature/component	Traditional firmware	IoT firmware
Fast boot-up	Desired	Essential
Prompt response	Yes	Mandatory
Reliability	Yes	Yes
Peripherals health	Yes	Not needed
Application and OS	No	Yes
Granular user control	No	Yes
Device attachments	Yes	Not needed
Network communicaton	Not essential	Mandatory
Middleware platform	No	Yes
Filesystem management	No	Yes
Cryptographic algorithm	Desired	Essential
Handle system updates	No	Yes
Binding with cloud/edge APIs	No	Yes
Small footprint	No	Yes

2.3. Some firmware issues of the embedded devices

The embedded devices pose serious security problems for the Internet because [11]: First, the hardware resources of embedded devices are limited (CPU capabilities, memory, power), so the firmware for the embedded devices is often tailored to the hardware. As a result, modern security mechanisms (such as ASLR, DEP) are often missing in the embedded systems. Second, the embedded devices lack security updates, many manufacturers do not provide security updates for their low-cost products for economic reasons. Third, the number of embedded devices is very large, so it is very accessible. Security analysis helps detect vulnerabilities in the embedded systems and helps manufacturers fix them before releasing products to the market or through updates. Therefore, research on security analysis for the embedded systems is very useful for Internet security but is very complicated also. There are many issues to be addressed due to the diversity of goals such as (1) considering a variety of hardware and operating systems; (2) using different types of analysis techniques, including code inspection, fuzzing, symbolic execution, emulation, protocol analysis, etc. The firmware analysis [12] can lead to: exposing sensitive data such as passwords, API keys, private certificates, compromising devices and tampering with data, understanding how the firmware works.

There are several questions that are always of interest to the firmware security for the embedded devices in general: Is there any information being leaked, is the device accepting unauthenticated commands, is the device susceptible to forwarding attacks, is the firmware image digitally signed, is the device running unnecessary services, is there any backdoors in the firmware, is the device running outdated software containing vulnerabilities?

Figure 2 [11] shows various security risks in the embedded systems.

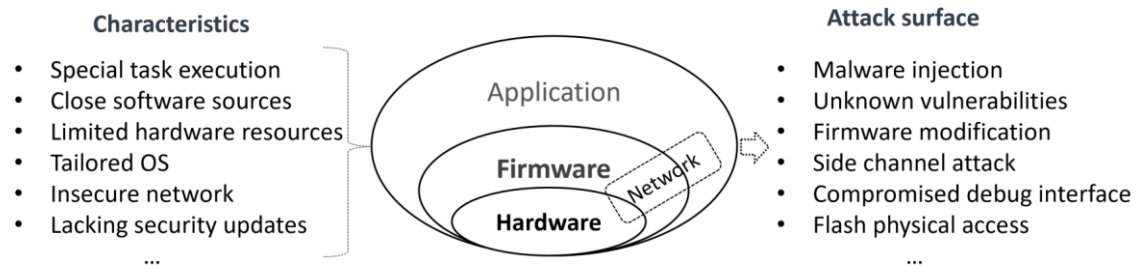

Figure 2. *Security risks in the embedded systems [11]*

2.4. Vulnerabilities in the embedded system firmware

Firmware can be a proprietary software that involves binary reverse engineering, especially for ARM and MIPS architectures [13]. Reverse engineering can reveal sensitive information such as authentication credentials, secret encryption/decryption keys. Figure 3 below shows potential vulnerabilities in the firmware.

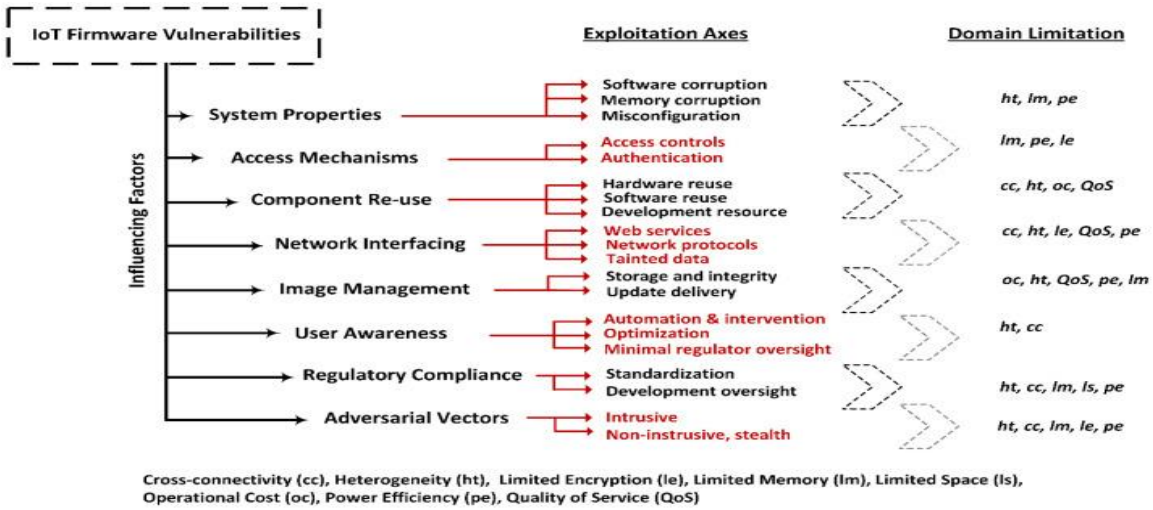

Figure 3. *Vulnerabilities, factors affecting firmware security [14]*

2.5. Some methods of the firmware protection

Manufacturers can take a number of measures to protect against the firmware attacks and enhance the security of their embedded systems. Here are some strategies [10]: Secure Boot, Firmware Integrity Verification, Code Review and Testing, Secure Development Practices, Patch Management and Updates, Secure Configuration, Hardware-Based Security, Encryption and Secure Communication, Authentication and Authorization, Access Controls, Secure Firmware Update Mechanism, Monitoring and Logging, Security Training and Awareness, Third-Party Component Security. By implementing these security measures, the manufacturers can significantly reduce the risk of the firmware attacks and enhance the overall security of their embedded systems. It is important to realize that security is a continuous process, and they need to continuously monitor and update their firmware.

2.6. Entropy

Entropy is a concept introduced by Shannon (1948) to measure the uncertainty about the occurrence of a given event, providing a piece of information about the system. The Entropy is the average rate at which information is generated by a random data source. The larger the entropy, the more information is provided by a new value in the process. For a signal, the Entropy is calculated as follows [15], [16]:

$$H = -\sum_{i=1}^{n} p(x_i) log_2(p(x_i)) \quad (1)$$

where $p(x_i)$ is the probability of the value x_i occurring.

Meaning of the Entropy in the Firmware Analysis:

– The Entropy is the information density or randomness of the file contents, or simply a measure of the number of bits required to represent each character that appears in the file. It is found that encrypted and compressed data have much higher Entropy than others. By calculating the Entropy of each file block, we can identify areas in the file that can be compressed or encrypted.

– In the systems with relatively severe hardware constraints such as embedded systems, firmware updates are often distributed in compressed form to save space. To analyze the firmware, it is first necessary to determine whether it is encrypted or compressed. One way to determine this is to perform an entropy analysis of the file. If the Entropy is very high, it is a good indication that the file is actually compressed or encrypted. To analyze the actual firmware, it must first be decompressed/decrypted. If we have a block of data with very high Entropy (i.e. near random), it would make no sense to try to treat it as code and parse it, as the result would be meaningless.

3. Firmware analysis

3.1. Analysis methodology

OWASP (Open Web Application Security Project) [17]: Being an open community with the goal of helping other organizations form, develop, collect, manage, and maintain aspects of application and device security. The OWASP-FSTM (OWASP Firmware Security Testing Methodology) Guide refers to the OWASP Firmware Security Testing Methodology. The FSTM methodology is divided into nine phases to ensure that, when complied, the investigator will conduct a comprehensive security analysis of an embedded or IoT device. The stages in the methodology are presented in Table 2 [17], [18], [19]:

Table 2. *Steps in the firmware analysis methodology*

Stage	Description
1. Information gathering and reconnaissance	Acquire all relative technical and documentation details pertaining to the target device's firmware.
2. Obtaining firmware	Attain firmware using one or more of the proposed methods listed.
3. Analyzing firmware	Examine the target firmware's characteristics.
4. Extracting the filesystem	Carve filesystem contents from the target firmware.
5. Analyzing filesystem	Statically analyze extracted filesystem configuration files and binaries for vulnerabilities.
6. Emulating firmware	Emulate firmware files and components.
7. Dynamic analysis	Perform dynamic security testing against firmware and application interfaces.
8. Runtime analysis	Analyze compiled binaries during device runtime.
9. Binary exploitation	Exploit identified vulnerabilities discovered in previous stages to attain root and/or code execution.

3.2. Some information achieved through the firmware analysis

Usually, upon analysis, the following information will be obtained [3]:

- Software identification (the operating systems to be used, programs to be present, versions, services, vulnerabilities).
- CPU architecture.

- File system contents: Files containing passwords, encryption keys, public key certificates, executable files, configuration files, sensitive keywords. Usually, system accounts can be found in the /etc/passwd file, and hashed passwords are stored in the /etc/shadow file. Traditionally, the passwords are hashed based on the DES algorithm, and can be easily decrypted with the jonh tool.
- Multiple devices that can use the same cryptographic key, even devices from different vendors.
- Datasets of the private keys as found in the embedded devices
- Public key certificates: People can manipulate the private keys, public keys, and X.509 certificates with the openssl program. For example, people can view the contents of an X.509 certificate in PEM format using a command-line tool and estimate the number of devices connected to the Internet using the same public key certificate by searching through a search engine.
- The executable files that can be inspected to display the information in the header or reversed using tools such as radare 2.

Obtaining this information is crucial to the security of the device, as attackers can use it to gain access to the system. For example, default passwords can be found in files in /etc/passwd or even decrypted password files (/etc/shadow) to obtain information about user accounts and their corresponding passwords…

4. Analysis practice

4.1. EMBA analysis tool

EMBA (Embedded Analysis Toolkit) was born out of the necessity to address the growing concerns about software security. It was developed by a group of security researchers including Michael Messner and Pascal Echmann. EMBA is a powerful firmware analysis tool compared to those of the same field such as CHIPSEC, UEFITool, IDA Pro, Firmadyn, FACT.

EMBA has many features, making it an indispensable tool in software security analysis, some of which include [20], [21], [22], [23]:

– Analysis of the firmware images extracted from various devices such as laptops, servers and IoT devices, including encrypted firmware.

– Vulnerability detection: It scans the firmware for known vulnerabilities and issues, providing detailed reports. EMBA also uses version detection techniques through emulation with Qemu, based on the obtained binary versions, it will list out the related vulnerabilities through CVE.

– Device configuration audit: EMBA can extract configuration data from the firmware, allowing for in-depth auditing of device settings, searching for passwords and keys, if a hashed password is found, it will automatically perform a password dictionary attack.

– Reverse engineering support: Disassemble and analyze firmware code to provide details about its internal workings.

4.2. Computer configuration for running the EMBA analysis tool

The EMBA analysis tool is run on VirtualBox 7.0 which runs Ubuntu 20.04 operating system, 20GB memory, 100GB hard disk, 8 core CPU.

4.3. Analyzed firmwares

Information about the analyzed firmwares is summarized in Table 3.

Table 3. *Information of the analyzed firmwares*

No.	Devices	Vendors	Firmware Images	Sizes
1	IP-Camera	Dlink	DCS-5000l_v1.03.05.zip	4.4MB
2	Wireless Router	NETGEAR	N300-V1.1.0.54_1.0.1.zip	3.8MB
3	Wireless Router	TP-link	TL-WDR3500_V1.zip	6.22MB

4	Webcam	D-link	DCS-930-squashfs.bin	4MB

4.4. Analysis results

4.4.1. *The analysis results of the device 1 are summarized in Table 4.*

4.4.2. *The analysis results of the device 2 are summarized in Table 4.*

Table 4. *Summary of the analysis results of the devices 1&2*

	Firmware analysis in 4.4.1	Firmware analysis in 4.4.2
Iterm	**Explaination**	**Explaination**
Architecture	Detected architecture and endianness (verified): MIPS / EL	Detected architecture and endianness (verified): MIPS / EL
OS	Operating system detected (verified): Linux / v2.6.21	Operating system detected (verified): Linux / v2.6.21
File System	288 files and 63 directories detected	251 files and 63 directories detected
Shell scripts	Found 143 issues in 27 shell scripts	Found 166 issues in 28 shell scripts
Configuration issues	Found 5 password related details via STACS. Found 1 outdated certificates and 1 expiring certificates in 3 certificate files and in a total of 4 certificates. Found 0 interesting files and 1 files that could be useful for post-exploitation.	Found 5 password related details via STACS. Found 1 outdated certificates and 1 expiring certificates in 3 certificate files and in a total of 4 certificates. Found 1 kernel modules with 0 licensing issues. Found 0 interesting files and 1 files that could be useful for post-exploitation
SBOM	Identified a SBOM including 9 software components with version details	Identified a SBOM including 7 software components with version details.
CVE analysis	Identified 2771 CVE entries. Identified 747 High rated CVE entries / Exploits: 61 Identified 1740 Medium rated CVE entries / Exploits: 90 Identified 284 Low rated CVE entries /Exploits: 19 170 possible exploits available (13 Metasploit modules). Remote exploits: 3 / Local exploits: 31 / DoS exploits: 4 / Github PoCs: 0 / Known exploited vulnerabilities: 6 / Verified Exploits: 0	Identified 2817 CVE entries. Identified 746 High rated CVE entries / Exploits: 61 Identified 1783 Medium rated CVE entries / Exploits: 93 Identified 288 Low rated CVE entries /Exploits: 20 174 possible exploits available (17 Metasploit modules). Remote exploits: 3 / Local exploits: 31 / DoS exploits: 5 / Github PoCs: 0 / Known exploited vulnerabilities: 6 / Verified Exploits: 0
Password information	Found 5 password related details via STACS	Found 5 password related details via STACS
Certification	Found 1 outdated certificates and 1 expiring certificates in 3 certificate files and in a total of 4 certificates	Found 1 outdated certificates and 1 expiring certificates in 3 certificate files and in a total of 4 certificates
Configuration files	Found 0 interesting files and 1 files that could be useful for post-exploitation**.**	Found 3 possible configuration files: fstab; upnpd.conf; lld2d.conf

Vulnerabilities	Found 6708 possible vulnerabilities (via semgrep in Ghidra decompiled code) in 21 tested binaries. Found 387 usages of strcpy in 67 binaries.	Found 6928 possible vulnerabilities (via semgrep in Ghidra decompiled code) in 21 tested binaries. Found 281 usages of strcpy in 59 binaries.

4.4.3. *The analysis results of the device 3 are summarized in Table 5.*

4.4.4. *The analysis results of the device 4 are summarized in Table 5.*

Table 5. *Summary of the analysis results of the devices 3 & 4*

	Firmware analysis in 4.4.3	Firmware analysis in 4.4.4
Iterm	**Explaination**	**Explaination**
Architecture	Detected architecture and endianness (verified): MIPS / EL	Detected architecture and endianness (verified): MIPS / EB
OS	Operating system detected (verified): Linux / v2.6.36	Operating system detected (verified): Linux / v2.6.31
File System	866 files and 71 directories detected.	526 files and 53 directories detected.
Shell scripts	Found 79 issues in 28 shell scripts	Found 1 issues in 4 shell scripts.
Configuration issues	Found 1 areas with weak permissions. Found 3 password related details via STACS. Found 1 outdated certificates and 0 expiring certificates in 1 certificate files and in a total of 1 certificates. Found 56 kernel modules with 0 licensing issues. Found 0 interesting files and 1 files that could be useful for post-exploitation.	Found 3 areas with weak permissions. Found 1 authentication issues. Found 1 password related details. Found 2 password related details via STACS. Found 110 kernel modules with 0 licensing issues. Found 0 interesting files and 3 files that could be useful for post-exploitation.
SBOM	Identified a SBOM including 8 software components with version details.	Identified a SBOM including 12 software components with version details.
CVE analysis	Identified 2573 CVE entries. Identified 682 High rated CVE entries / Exploits: 44 Identified 1637 Medium rated CVE entries / Exploits: 64 Identified 254 Low rated CVE entries /Exploits: 16 124 possible exploits available (11 Metasploit modules). Remote exploits: 0 / Local exploits: 29 / DoS exploits: 4 / Github PoCs: 0 / Known exploited vulnerabilities: 8 / Verified Exploits: 0	Identified 2752 CVE entries. Identified 743 High rated CVE entries / Exploits: 56 Identified 1727 Medium rated CVE entries / Exploits: 80 Identified 282 Low rated CVE entries /Exploits: 18 154 possible exploits available (17 Metasploit modules). Remote exploits: 1 / Local exploits: 32 / DoS exploits: 5 / Github PoCs: 0 / Known exploited vulnerabilities: 9 / Verified Exploits: 0
Password information	Found 3 password related details via STACS.	Found 1 password related details. Found 2 password related details via STACS.

Certification	Found 1 outdated certificates and 0 expiring certificates in 1 certificate files and in a total of 1 certificate.	
Configuration files		
Vulnerabilities	Found 9691 possible vulnerabilities (via semgrep in Ghidra decompiled code) in 21 tested binaries. Found 1059 usages of strcpy in 138 binaries.	Found 11168 possible vulnerabilities (via semgrep in Ghidra decompiled code) in 21 tested binaries. Found 966 usages of strcpy in 232 binaries.

Tables 4 and 5 contain only summary information from the EMBA-based firmware analysis results. For example, the file system information shows the directory tree, which can be browsed to find sensitive files: configuration files, user information files, script files, etc. These files may contain information that makes attacks easier. VCE vulnerabilities are publicly available vulnerabilities, and attackers can also rely on them to carry out attacks.

4.5. Interpretation and comments

From the above analysis results, it can be seen that:

- Most of the current embedded and IoT devices use MIP architecture, integrating Linux operating system, in which the components such as kernel, file system contain configuration files, weaknesses by which they can be attacked (For example, password information, script files containing risks).
- Most firmware contains many CVEs (Common Vulnerabilities and Exposures - a list of identified security vulnerabilities, publicly announced CVEs), security vulnerabilities by which hackers can use them for attack purposes.
- The embedded and IoT devices are becoming more and more popular, each device having firmware contains many risks that can be attacked. This is probably a huge challenge for manufacturers, deployers and security.

4.6. Recommendations

- To be able to partly prevent security risks from the firmware perspective, it requires the cooperation of users and manufacturers.
- On the user side: regularly update firmware from the manufacturer's main website, change default passwords, set strong passwords, configure device access using strong security standards.
- On the manufacturer side: regularly upgrade firmware to patch VCE errors, use strong authentication mechanisms, strong cryptographic methods.

5. Conclusion

In this Article, the Author has presented the security analysis for the embedded devices and IoT. It is to present firstly the information related to the hidden risks in the embedded systems which will, once deployed, become the attack surface of the adversary, in which the firmware of the device contains vulnerabilities - which, if exposed, can be used to attack the system if analyzed; secondly, the methodology in the firmware analysis; and finally, the EMBA open source software is used to test the analysis on some firmwares of the device.

- **Advantages:** Systematization of some core knowledge of the embedded system security, which results in raising an awareness, practicing compliance to limit risks in the actual deployment. The analysis process is automated withouting running the firmware.
- **Disadvantages:** Limitation to the analysis only to recognize the vulnerabilities that are at risk of being attacked in a limited number of the firmware and lack of such on-a-larger-scale deployment for statistics.

- **Development orientation:** Deployment of the powerful systems with a large-scale analysis for statistics.

Conflict of interest

The author duly declares no conflicts of interest in this Article.

REFERENCES

[1] D. G. Akestoridis, *Firmware Analysis of Embedded Systems*, Carnegie Mellon University, 2018.
[2] A. Cui, M. Costello, and S. J. Stolfo, "When Firmware Modifications Attack: A Case Study of Embedded Exploitation," Department of Computer Science, Columbia University, New York, USA, 2013.
[3] I. Nadir, H. Mahmood, and G. Asadullah, "A Taxonomy of IoT Firmware Security and Principal Analysis Techniques," *Elsevier*, Feb. 15, 2021.
[4] F. Boland, *Automated Security Analysis of Firmware*, KTH Royal Institute of Technology, Stockholm, Sweden, 2022.
[5] Vina Aspire, "More than 80% of Global Businesses are Victims of Firmware Attacks," [Online]. Available: https://vina-aspire.com/hon-80-doanh-nghiep-toan-cau-la-nan-nhan-cua-cuoc-tan-cong-firmware/
[6] Techpro, "Vietnamese Voice Warning Firmware on Suprema FSF2 and BS3," [Online]. Available: https://techpro.vn/vi/tin-tuc/firmware-canh-bao-am-thanh-giong-noi-tieng-viet-tren-suprema-fsf2-va-bs3.html
[7] L. Phuong, "Vietnamese Scientists Gradually Master Hardware Security Solutions," *Journal of Information and Communications*, 2019.
[8] H. Minh, "Security Camera Equipment Standards to Be Developed," [Online]. Available: https://baochinhphu.vn/se-xay-dung-cac-tieu-chuan-thiet-bi-camera-an-ninh-102230728164637256.htm, 2023.
[9] A. Bhardwaj, K. Kaushik, S. Bharany, and S. Kim, "Forensic Analysis and Security Assessment of IoT Camera Firmware for Smart Homes," *Egyptian Informatics Journal*, vol. 24, no. 4, p. 100409, Dec. 2023.
[10] M. Muench, *Dynamic Binary Firmware Analysis: Challenges & Solutions*, Sorbonne Université, 2021.
[11] X. Zhou, P. Wang, L. Zhou, P. Xun, and K. Lu, "A Survey of the Security Analysis of Embedded Devices," *Sensors*, 2023.
[12] Prabhankar, "Firmware Analysis Part-1,2," 2021. [Online]. Available: https://prabhankar.medium.com/firmware-analysis-part-1-cd43a1ad3f38
[13] NSE Hacking Lab, "PenTest Process Planning," [Online]. Available: https://nse.digital/pages/guides/pentest-process-planning.html
[14] T. Bakhshi, B. Ghita, and I. Kuzminykh, "A Review of IoT Firmware Vulnerabilities and Auditing Techniques," *PMCID: PMC10821153*, 2024.
[15] R. Lyda, J. Sparta, and B. Hamrock, "Using Entropy Analysis to Find Encrypted and Packed Malware," *IEEE Computer Society*, 2007.
[16] Whiteheart0, "Entropy Analysis: A Critical Test for Malware," [Online]. Available: https://whiteheart0.medium.com/entropy-analysis-a-critical-test-for-malwares-69939f5b8b1
[17] F. Á. Wic, J. M. G. Moreno, and A. V. Blanco, "IoT and Embedded Devices Security Analysis Following OWASP," [Online]. Available: https://www.tarlogic.com/blog/security-analysis-on-iot-owasp, 2022.
[18] T. C. Clancy, R. Gerdes, Y. Yang, J. Black, P. Schaumont, and D. A. Brown, "Analysis of Firmware Security in Embedded ARM Environments," Arlington, Virginia, 2019.
[19] S. U. Haq, Y. Singh, A. Sharma, R. Gupta, and D. Gupta, "A Survey on IoT & Embedded Device Firmware Security: Architecture, Extraction Techniques, and Vulnerability Analysis Frameworks," *Springer*, 2023.
[20] P. Bourmeau, "A Brief Introduction to Firmware Extraction," 2020.
[21] S. Vasile, D. Oswald, and T. Chothia, "Breaking All the Things – A Systematic Survey of Firmware Extraction Techniques for IoT Devices," University of Birmingham, 2018.
[22] J. M. Smith, "Case Analysis of Firmware Vulnerabilities and Exploitation," 2016.
[23] R. Sharma, "Unveiling Vulnerabilities: A Deep Dive into Firmware Penetration Testing – Part 1," Sep. 19, 2023. [Online]. Available: https://ravi73079.medium.com/unveiling-vulnerabilities-a-deep-dive-into-firmware-penetration-testing-part-1-904599cd79be
[24] D. de Ruck, V. Goeman, and J. Lapon, "Hands-on Workshop: Hacking and Protecting Embedded Devices," June 2022.

Cong Doan Dinh graduated from the Faculty of Electronics and Telecommunication in 1999 and then was granted with a Master's degree in Computer Science in 2002 by the HCMC University of Technology. He is currently working at the Faculty of Information Technology, the HCMC University of Technology and Education. His research interests include Network Security, Data Communications, Logic Design, IoT, Embedded Systems related technologies. Email: doandc@hcmute.edu.vn. ORCID: https://orcid.org/0009-0004-2138-5939