An Investigation into Router Firmware Security and the Embedded Device Challenge

P. Raghu Vamsi, Stuti Sharma, Aarushee Krishna, and Devika Gupta

***Abstract*—In the present IoT world the need for robust network security is more important than ever as the internet-connected gadgets are everywhere and our dependence on networked systems is increasing. Ensuring the safe transfer of data and protecting the connected devices, primarily through routers that serve as gateways to the internet, has become a critical priority. The security of router firmware is of paramount importance because any weaknesses in these vital components could lead to extensive network breaches and data compromises. Due to the growing prominence of embedded devices and many of which are surprisingly lacking in robust security measures, making them attractive targets for potential attacks. Among them, routers which function as bridges connecting local and global networks have become prime targets for Cyber attacks. To this end, this paper focuses on analyzing the firmware of TP-Link Archer AX55 AX3000 dual band Gigabit Wi-Fi 6 Router crafted by TP-Link technologies. The objective is to conduct a through security analysis to fortify the security of router firmware and then providing countermeasures for strengthening the safety and resilience of the Internet's infrastructure. The insights and recommendations resulting from this work stand to benefit router manufacturers, network administrators, and end-users in their ongoing efforts to combat ever-evolving Cyber threats.**

***Keywords*—Cyber security, Embedded devices, IoT, Network security, Router firmware, TP-Link Archer AX55 Router.**

I. Introduction

In today's highly interconnected world, routers serve as the backbone of our digital infrastructure, facilitating the seamless exchange of data that links businesses and individuals across the globe. Given their critical role, it is imperative that routers are equipped with robust security measures, as they are prime targets for online threats. The repercussions of router security vulnerabilities are profound, ranging from data breaches to corporate espionage and network intrusions. At the heart of these devices, router firmware plays a pivotal role in ensuring their proper functioning and security. It acts as both the brains and the heart, orchestrating operations and safeguarding against potential dangers. Consequently, scrutinizing router firmware is a fundamental task for security researchers as it enables the identification and evaluation of vulnerabilities. By doing so, they significantly enhance the overall security of these devices assuring the reliability and safety of our digital networks. Nevertheless, conducting security assessment of router firmware comes with its share of challenges. This environment is marked by complexity, vendor-specific methods, and a lack of standardized firmware images, making it a formidable task to assess and bolster the security of these vital devices, particularly due to their vendor-specific nature. Adding to the complexity is the restricted access to firmware source code, a rarity among vendors. Firmware binary distribution packages, typically provided in binary image form, are occasionally encrypted and obfuscated even when access is granted. While these security measures aim to thwart reverse engineering and exploitation attempts, they may not fully ad- dress zero-day vulnerabilities or known CVEs in the software components, introducing an added layer of complexity. The understanding of how firmware can be manipulated to execute unauthorized activities, potentially jeopardizing the security of the entire ecosystem, is of paramount importance [1], [2].

Routers serve as the linchpin of digital infrastructure by regulating the flow of data and communication between de- vices and the internet. However, the software that powers these critical devices, known as router firmware, grapples with variety of security challenges and vulnerabilities that demand rigorous analysis and mitigation [3]. This study aims to undertake a comprehensive security analysis of router firmware to identify vulnerabilities, assess potential threats, and propose effective mitigation strategies. This work presents analysis of firmware code base, protocol implementations, and configuration management to uncover weaknesses and potential entry points for attackers. Open Web Application Security Project (OWASP) methodology is followed to systematically address the complex issues and vulnerabilities. Also, this approach is chosen because it will facilitate the identification of vulnerabilities, the evaluation of potential risks, and the formulation of mitigation strategies.

To this end, the primary objectives of this work encompass vulnerability assessment, threat modeling, penetration testing, and recommending mitigation Strategies. In the rest of the paper after discussing the background literature in Section II, we present the methodology of the proposed study such as systematic identification, assessment, and re-mediate vulnerabilities within router firmware, thus safeguarding the integrity, confidentiality, and availability of network data in Section III. Section IV presents the observations followed by conclusion in Section V.

II. Background

Software vulnerabilities have been on a steady increase each year, with a rise in both the total number of vulnerabilities discovered and the speed at which they are identified. For instance, in 2010, the Common Vulnerabilities and Exposures (CVE) database received approximately 4,600 new vulnerability reports. This number grew to 6,500 in 2016 and skyrocketed to over 14,700 reports in 2017. It's common practice for clients and companies to rely on commercial off-the-shelf binaries for their products and operations. These external products often undergo a rigorous process, including software security validation, such as black-box penetration testing. In the context of mobile and Internet of Things (IoT) vendors, open source code is often re-purposed by penetration testers and tailored to their specific devices [4], [5].

The firmware, possessing significant authority, plays a pivotal role in bridging upper-level applications and underlying hardware components. If vulnerabilities within embedded sys- tem firmware code are not promptly addressed and controlled, potential attackers can exploit these weaknesses, resulting in disruptive actions such as denial of service or even the self- destruction of embedded devices. This can lead to widespread equipment paralysis, substantial economic losses, and severe security crises. The earlier vulnerabilities are detected, the more effectively the security of embedded devices can be bolstered [6], [7].

Some of the testing methods mentioned in the literature include [6], [8]–[12]:

1) Fuzzy Testing: This technique involves injecting a large volume of semi-valid data into the application to uncover defects.
2) Behavioral Analysis: It assesses code behavior and analyzes code results to identify potential malicious behavior or vulnerabilities.
3) Homology Analysis: This method considers the possibility that different brands of devices may run similar firmware and share common third-party libraries, potentially leading to vulnerabilities. For instance, the study found that in the case of D-Link routers, 63.15% of them, including the D-Link DIR-645 router, had common vulnerabilities that could impact the entire D- Link router family.

Costin et al. [3] delved into automated dynamic firmware analysis at scale with a focus on embedded web interfaces. Authors explored the challenges and complexities of firmware analysis and present their research on identifying vulnerabilities within embedded systems. They provided insights into the methods and tools used for such analysis and share their findings on vulnerabilities detected in embedded web interfaces. Hemram et al. [12] discussed the critical issue of firmware vulnerability detection in embedded systems and the Internet of Things (IoT). Authors proposed methods for identifying vulnerabilities within firmware. Hou et al. [11] addressed the topic of vulnerability detection in embedded system firmware. This work highlighted the significance of identifying and mitigating vulnerabilities within embedded systems. Brezolin et al. [2] presented a novel approach to vulnerability detection using IoT network traffic analytics. Nadir et al. [6] introduced a comprehensive taxonomy of IoT firmware security and principal firmware analysis techniques.

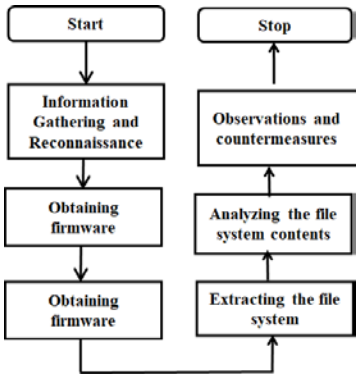


Fig. 1: Methodology of the proposed study

III. Methodology

The current study is primarily focused on conducting a comprehensive assessment of router firmware. This comprehensive examination involves a multi-step process that encompasses reverse engineering the binary image to gain insights into its functionality, detecting vulnerabilities such as buffer overflows and injection attacks, proposing effective countermeasures to bolster security, unveiling concealed features, and meticulously documenting all findings in comprehensive reports. The overarching objective of this work is to make a significant contribution to the safety and dependability of router firmware thereby fortifying protection against potential security threats. To facilitate this analysis, current study leverages the well- established framework provided by the Open Web Application Security Project (OWASP). OWASP delivers a comprehensive methodology that encompasses a suite of community tools and techniques tailored for firmware analysis during penetration testing. The Firmware Security Testing Methodology (FSTM), consisting of nine distinct steps as shown in Figure 1. Each of these steps serves as a guidance tool for analysts in the process of identifying and confirming vulnerabilities. The description of nine steps outlined in the OWASP methodology are as follows [6], [8], [10]:

1) Information gathering and reconnaissance: This is a critical initial phase which involves collecting relevant information about the firmware.
2) Obtaining firmware: The step entails procuring or obtaining the firmware to move forward with the analysis.
3) Analyzing firmware: This phase is pivotal as it involves dissecting the firmware to understand its inner workings.
4) Extracting the filesystem: By extracting the filesystem, we gain access to the core contents of the firmware, allowing for deeper inspection.

SOFTWARE	
Protocols	IPv4 IPv6
Service Kits	HomeShield Learn More>
OneMesh™	OneMesh™ Supported Without replacing your existing devices or buying a whole new WiFi ecosystem, OneMesh™ helps you create a more flexible network that covers your entire home with TP-Link OneMesh™ products. Learn More> All OneMesh Products>

Fig. 2: Software configurations of Archer AX-55

HARDWARE		SECURITY	
Processor	Dual-Core CPU	WiFi Encryption	WPA WPA2 WPA3 WPA/WPA2-Enterprise (802.1x)
Ethernet Ports	1× Gigabit WAN Port 4× Gigabit LAN Ports	Network Security	SPI Firewall Access Control IP & MAC Binding Application Layer Gateway HomeShield Security Real-Time IoT Protection Malicious Site Blocker Intrusion Prevention System DDoS Attack Prevention Home Network Scanner
USB Support	1× USB 3.0 Port Supported Partition Formats: NTFS, exFAT, HFS+, FAT32 Supported Functions: Apple Time Machine FTP Server Media Server Samba Server		
Buttons	Wi-Fi/WPS Button Power On/Off Button Reset Button		
Power	12 V ⎓ 2 A	Guest Network	1× 5 GHz Guest Network 1× 2.4 GHz Guest Network
		VPN Server	OpenVPN PPTP

Fig. 3: Hardware and security configurations of Archer AX-55

Fig. 4: Shodan reports for TP-Link

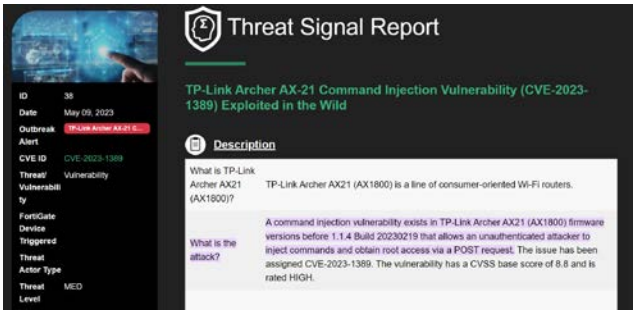


Fig. 5: Known vulnerabilities for TP-Link Archer AX Family

5) Analyzing the filesystem contents: The analysis continues by scrutinizing the contents of the filesystem in detail.
6) Observations and countermeasures: The final stage involves drawing observations from the analysis and proposing countermeasures to enhance security.

These steps provide a comprehensive structure for the current work and facilitate a systematic approach to firmware security analysis in the context of router firmware. This study focuses on analyzing the firmware of TP-Link Archer AX55 AX3000 Dual Band Gigabit Wi-Fi 6 Router crafted by TP-Link Technologies. Software and hardware configurations of the considered router are provided in the Figure 2 and 3 respectively. The following subsections describes these steps in detail.

To follow the methodology of firmware reverse engineering various tools are utilized. We have performed the experiments on Kali Linux operating system [13]. It is a renowned penetration testing and ethical hacking platform. It provides a robust environment for conducting various security assessments and analysis tasks. This operating system installed in virtual environment in Oracle Virtual Box. We use the tools such as 1) Binwalk [14]: this tool stands as a fundamental tool in our arsenal, crucial for the identification and extraction of file system components within firmware. It plays a key role in deciphering the internal structure of firmware binaries. 2) Firmwalker [15]: It is an automation tool we leverage for the examination of filesystem contents and uncompiled code. It aids in the detection of vital clues, such as keywords, URLs, and email addresses, within the firmware. 3) Binvis.io [16]: This is another valuable resource that we utilize to generate critical visual representations of firmware data. It offers insights through images such as entropy, detail, byte class, and magnitude images, contributing to a more comprehensive understanding of the data. Other supporting tools such as hashdump is utilized to observe the hashes.

A. Information gathering and reconnaissance

In this step, we gathered all pertinent technical and documentation details related to the firmware of the target devices. To accomplish this, we utilized the Shodan search engine. Figure 4 illustrates the process of information collection through the Shodan search engine. Our specific focus here is on reports related to TP-Link router to obtain technical parameters about these devices. After successful gathering of information, the next step is to conduct a systematic search for known vulnerabilities within the framework of a Threat Signal report, as depicted in Figure 5. This search enables us

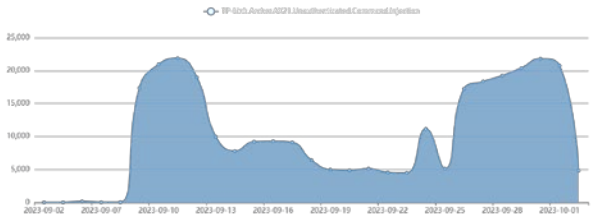


Fig. 6: Command injection attacks on number of devices vs. date

to identify potential vulnerabilities in TP-Link Archer AX21 routers. Notably, our findings reveal a heightened susceptibility of TP-Link Archer AX21 routers to command injection vulnerabilities. Furthermore, the collected information also sheds light on the historical context of command injection attacks on these routers effectively highlighting the evolving threat landscape in Figure 6. It is observed that TP-Link Archer AX21 is more prone to command injections and the information obtained regarding command injection attacks on the router across various dates.

B. Obtaining firmware

To obtain the necessary firmware, we have established a direct channel for acquisition from the vendor. We obtained the firmware from the TP-Link vendor's official website (https://www.tp-link.com/in/home-networking/wifi-router/archer-ax55/). With this link we have accessed the most reliable and up-to-date firmware version originating from the trusted vendor's platform. By obtaining the firmware directly from the vendor's designated source, we ensure the integrity and authenticity of the software, setting a solid foundation for our subsequent security analysis and testing procedures.

C. Analyzing firmware

In this phase, a multi-step process is followed to thoroughly examine the firmware and to gain deeper insights into its composition and functionality. The steps followed are as follows

- Step 1: we employed the command-line tool known as "binwalk". This versatile tool scans binary files and instrumental in detecting concealed or encoded data within the firmware. Furthermore, it aids in extracting information related to the firmware's filesystem. Binwalk's functionality allows us to uncover hidden elements within the firmware. Based on our analysis using the binwalk tool, we have determined several key characteristics of the Archer firmware. It is ascertained that the Archer firmware employs the squashfs filesystem which is a file compression system frequently used in embedded devices. Additionally, our analysis revealed that the firmware utilizes the little-endian byte order data format used in computing. Figure 7 presents the binwalking the firmware file.
- Step 2: This step is pivotal in our examination of the firmware. We utilized the "hexdump" command, as demonstrated below:
  $ hexdump -C <bin file name> > hexdump.out
  It entails byte by byte examination of the firmware's binary content. The output of this examination is stored in an output file named "hexdump.out". This command allows us to inspect and display the contents of binary files, offering various formats such as hexadecimal, decimal, octal, or ASCII representations. It enables a comprehensive understanding of the firmware's binary data. Figure 8 presents the use of hexdump on firmware file.

Fig. 7: Binwalking the firmware file

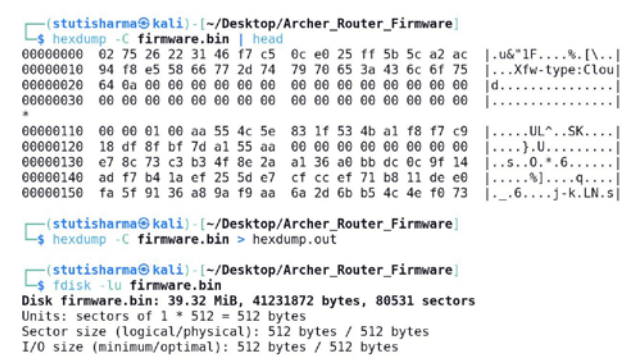
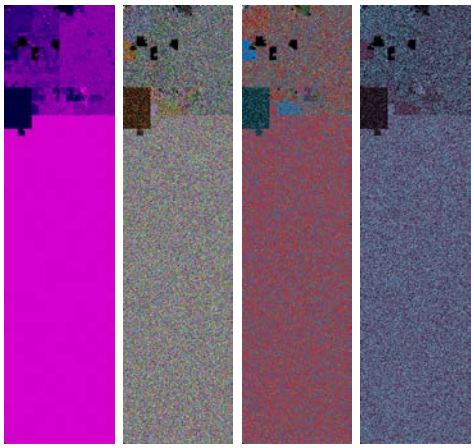


Fig. 8: Analyzing firmware with hexdump

- Step 3: The next step involves the utilization of the "fdisk" command in combination with the "-lu" flag, applied as follows:
  $ fdisk -lu <bin>
  This command is essential for listing the partitions and filesystems of the firmware. It provides an overview of the firmware's drive structure to comprehend how the firmware is organized and structured. The "fdisk" command has played a crucial role in our analysis. Beyond its fundamental role in specifying the type of filesystem used on each partition, "fdisk" also facilitates the creation and removal of partitions, as well as the allocation of partition sizes. This command provides essential insights into the organization of the firmware, and it has been an invaluable asset in our exploration of the firmware's structure and layout.

Figure 9 presents a set of images that were acquired through the binvis.io platform. These images include entropy image, detail image, byte class image, and magnitude image. Entropy (Figure 9(a)) is a measure of randomness or disorder within data. Entropy image reveals areas within the firmware where data is highly random or exhibits unusual patterns. High entropy regions may suggest the presence of encryption or compressed data. The detail image (Figure 9(b)) provides a visual representation of specific patterns or structures within the firmware binary. This can help analyzing sections of interest, potential vulnerabilities, or unexpected data configurations. Byte class image (Figure 9(c)) helps categorizing data within the firmware into different classes based on their



(a) (b) (c) (d)

Fig. 9: Images obtained from binvis.io (a) Entropy (b) Details (c) Byte class (d) Magnitude

characteristics. Byte class analysis can assist in identifying code, data, or other components within the firmware, aiding in its understanding. Magnitude image (Figure 9(d)) highlight the significance of certain features within the binary. It can help pinpoint sections that are particularly important for analysis or that may require further scrutiny.

D. Extracting the filesystem

This phase is an important phase in the entire anlysis as it requires a combination of automated and manual techniques for a thorough review of the firmware's filesystem contents. Automated extraction is facilitated through the "binwalk -ev <bin>" command. This command automatically extracts contents in the designated format "_binaryfile/filesystem/," simplifying the process and providing immediate access to known file types. In cases where the magic byte of the filesystem is not present in binwalk's signatures then the manual extraction is initiated. The following steps guide this manual extraction process:

- Use the "dd" command, as shown below, to extract the Squashfs filesystem contents and create a file named "dir.squashfs":
  $ dd if=Firmware.bin bs=1 skip=10667554 of=dir.squashfs
- The next step involves utilizing the command "unsquashfs dir.squashfs" to decompress, extract, and list the contents of the Squashfs filesystem. This approach allows for a comprehensive review of the firmware's filesystem contents. The automated extraction process, with "binwalk -ev," efficiently handles known file types, generating a new folder for each with the "_fileName.extracted" naming convention. In cases where the magic byte is notdetected, the manual method involves using binwalk to identify the filesystem's offset within the binary, followed by the extraction of the compressed filesystem. Subsequently, the filesystem is manually decompressed and extracted based on its type. This combined approach ensures that all filesystem contents are accurately extracted and available for in-depth examination and analysis, a critical aspect of our research.

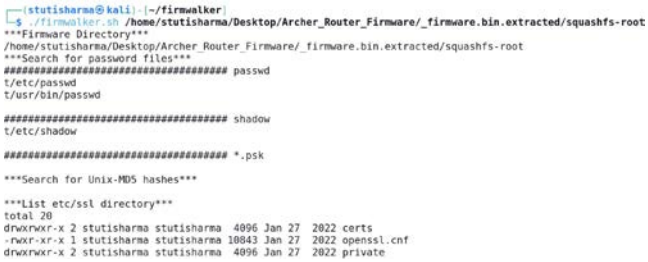


Fig. 10: Extracting password files from firmware

E. Analyzing the filesystem contents

In this phase we delve into the examination of filesystem contents aiming to gather essential clues for dynamic and run-time analysis. This multifaceted analysis encompasses both statically reviewing uncompiled code and investigating the filesystem's components. These are all facilitated through the utilization of automation tools like "firmwalker". This step commences with the deployment of automation tools. The "firmwalker tool is employed to parse shadow files, password files, SSL-related files, configuration files, script files, and much more within the filesystem. To execute this, the following command is utilized:

$ ./firmwalker.sh /home/Desktop/Archer_Router_Firmware/ firmware.bin.extracted/squashfs-root/

This command helps extraction of crucial insights, such as identifying URLs, email addresses, IP addresses, and the detection of significant keywords, including but not limited to 'admin,' 'password,' 'remote,' 'API keys,' and numerous others. Figure 10 shows the extraction of password files. Figure 11 (a) to (p) respectively shows searching for (a) SSL related files, (b) configuration files, (c) other binary files, (d) admin files, (e) root files, (f) password files, (g) dropbear files, (h) ssl files, (i) private keys, (j) telnet details, (k) secret keys, (l) pgp files, (m) gpg files, (n) tokens, (o) api keys. By statically reviewing uncompiled code and filesystem components with the aid of automation tools, we obtain valuable information that lays the foundation for dynamic and run-time analysis. This meticulous approach empowers us to detect potential vulnerabilities, uncover hidden configurations, and pinpoint areas of interest that might require further investigation.

F. Observations and countermeasures

With the study made about the router firmware, the following observations are made and countermeasures are suggested.

1) Observations:

1) Firmware Analysis Complexity: The firmware analysis process is to be multifaceted comprising automated and manual steps. This complexity underscores the need for a structured approach that combines both methods,

Fig. 11: Snapshot of results obtained from firmwalker tool to search for (a) SSL related files, (b) configuration files, (c) other binary files, (d) admin files, (e) root files, (f) password files, (g) dropbear files, (h) ssl files, (i) private keys, (j) telnet details, (k) secret keys, (l) pgp files, (m) gpg files, (n) tokens, (o) api keys

ensuring that all file system contents are accurately extracted and available for examination.

2) Filesystem Clues: The utilization of automation tools like "firmwalker" enables the parsing of critical files such as shadow files, password files, SSL-related files, and configuration files (Figure 11). This reveals valuable clues that can be pivotal for dynamic and runtime analysis.

3) Keyword Detection: The ability to detect keywords, including 'admin,' 'password,' 'remote,' and 'API keys,' among others, within the filesystem is instrumental in identifying potential vulnerabilities and security weaknesses.

2) Countermeasures:

1) Structured Approach: Employing a well-defined structured methodology for firmware analysis is essential. Combining automated tools with manual techniques ensures a comprehensive review of the firmware, minimizing the risk of overlooking critical components.
2) Regular Updates and Patch Management: To address vulnerabilities and enhance firmware security, regular updates and patch management practices should be implemented. This ensures that known vulnerabilities are addressed promptly.
3) Access Control: Restricting access to sensitive files and configuration information within the firmware can mitigate the risk of unauthorized access and data exposure. Implementing stringent access controls can bolster security.
4) Code Review: Statically analyzing uncompiled code is crucial. This can be complemented by regular code reviews to identify and address security issues and vulnerabilities at the source code level.
5) Keyword Blacklisting: Implementing keyword blacklisting mechanisms can enhance security. When sensitive keywords are detected within the firmware then automatic alerts or actions can be triggered to prevent unauthorized access or modifications.
6) User Education: Educating users and administrators about best practices for firmware management and security is essential. This includes strong password policies and the importance of keeping firmware up to date.

IV. Conclusion

In this paper, we have undertaken a comprehensive exploration of firmware analysis and security assessment of embedded devices. The process involves an intricate series of steps ranging from firmware acquisition to detailed examination of firmware components. We have employed a range of tools and technologies to facilitate our analysis and enhance its efficiency. The observations made throughout this work have revealed that the significance of a structured approach to firmware analysis was required. It is observed that automation tool such as "firmwalker" played a crucial role of in parsing critical files and identifying keywords, email addresses, and URLs within the firmware. In this analysis, several key observations have emerged that shed light on the multifaceted nature of this process. Such a method ensures that all file system contents are accurately extracted, and this comprehensive examination is pivotal in uncovering potential vulnerabilities. It is observed that the considered TP-Link Archer AX55 AX3000 router firmware have several security flaws which ultimately leaking sensitive information related to keyword. The study recommends regular updates, access control, code review, keywords blacklisting, and user education as countermeasures while using this router. In the future work, we attempt to further verify the detected vulnerabilities using automated pen-testing frameworks like Metasploit and Cobalt Strike. Moreover, we aim to enhance our methodology by exploring multiple alternatives for extraction and analysis.

References

[1] F. Bolandi, "Automated security analysis of firmware," 2022.
[2] U. Brezolin, A. Verg ütz, and M. Nogueira, "A method for vulnerability detection by iot network traffic analytics," *Ad Hoc Networks*, vol. 149, p. 103247, 2023.
[3] A. Costin, A. Zarras, and A. Francillon, "Automated dynamic firmware analysis at scale: A case study on embedded web interfaces," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, 2016, pp. 437–448.
[4] U. Ravindran and R. V. Potukuchi, "A review on web application vulnerability assessment and penetration testing," *Review of Computer Engineering Studies*, vol. 9, no. 1, 2022.
[5] P. R. Vamsi and A. Jain, "Practical security testing of electronic commerce web applications," *International Journal of Advanced Networking and Applications*, vol. 13, no. 1, pp. 4861–4873, 2021.
[6] I. Nadir, H. Mahmood, and G. Asadullah, "A taxonomy of iot firmware security and principal firmware analysis techniques," *International Journal of Critical Infrastructure Protection*, p. 100552, 2022.
[7] M. Ibrahim, A. Continella, and A. Bianchi, "Aot-attack on things: A security analysis of iot firmware updates," in *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*, 2023.
[8] P. Sun, L. Garcia, G. Salles-Loustau, and S. Zonouz, "Hybrid firmware analysis for known mobile and iot security vulnerabilities," in *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2020, pp. 373–384.
[9] J. Selsøyvold and I. H. Trosdahl, "A security assessment of an embedded iot device," B.S. thesis, NTNU, 2022.
[10] A. Qasem, P. Shirani, M. Debbabi, L. Wang, B. Lebel, and B. L. Agba, "Automatic vulnerability detection in embedded devices and firmware: Survey and layered taxonomies," *ACM Computing Surveys (CSUR)*, vol. 54, no. 2, pp. 1–42, 2021.
[11] J.-b. Hou, T. Li, and C. Chang, "Research for vulnerability detection of embedded system firmware," *Procedia Computer Science*, vol. 107, pp. 814–818, 2017.
[12] S. Hemram, G. J. W. Kathrine, G. M. Palmer, and S. V. Ewards, "Firmware vulnerability detection in embedded systems and internet of things," in *2022 International Conference on Augmented Intelligence and Sustainable Systems (ICAISS)*. IEEE, 2022, pp. 1161–1167.
[13] S.. Kali Linux. (2023, "Penetration testing and ethical hacking linux distribution." [Online]. Available: https://www.kali.org/
[14] ReFirmLabs, "Refirmlabs/binwalk: Firmware analysis tool," GitHub, binwalk Repository. [Online]. Available: https://github.com/ReFirmLabs/binwalk
[15] F. Repository, "Firmwalker repository analysis tool," GitHub, firmwalker Repository. [Online]. Available: https://github.com/craigz28/firmwalker
[16] "binvis/binvis.io: The binvis.io site," GitHub (2023, September 28). [Online]. Available: https://github.com/binvis/binvis.io

P. Raghu Vamsi - Department of Computer Science Engineering and IT, Jaypee Institute of Information Technology, A-10, Sector 62, Noida, India – 201309 (email: prvonline@yahoo.co.in)
Stuti Sharma - Department of Computer Science Engineering and IT, Jaypee Institute of Information Technology, A-10, Sector 62, Noida, India – 201309 (email: stutisharma2703@gmail.com)

Aarushee Krishna - Department of Computer Science Engineering and IT, Jaypee Institute of Information Technology, A-10, Sector 62, Noida, India – 201309 (email: aarusheekrishna@gmail.com)
Devika Gupta - Department of Computer Science Engineering and IT, Jaypee Institute of Information Technology, A-10, Sector 62, Noida, India – 201309 (email: devikagupta62@gmail.com)